# INTELIGENCIA ARTIFICIAL

## Aplicada a Robótica y Automatización

Humberto Sossa Azuela
Fernando Reyes Cortés

Marcombo
Alfaomega

*Inteligencia artificial aplicada a Robótica y Automatización*

Humberto Sossa Azuela y Fernando Reyes Cortés



Primera edición: septiembre 2020
ISBN: 978-607-538-666-9

Primera edición: MARCOMBO, S.L. 2021



ISBN: 978-84-267-3316-0
D.L.: B 7322-2021

Impreso en Servicepoint
*Printed in Spain*

# Acerca de los autores

**Dr. Juan Humberto Sossa Azuela:** Doctor por el Instituto Politécnico de Grenoble, Francia en 1992. Profesor-investigador del Instituto Politécnico Nacional y Jefe del Laboratorio de Robótica y Mecatrónica del CIC-IPN. Miembro del SNI-3. Academia Mexicana de Ciencias. Con 37 años de experiencia en docencia en la Universidad de Guadalajara, Tecnológico de Monterrey, CINVESTAV-IPN y el CIC-IPN. Autor de 4 libros y más de 450 artículos en revistas, capítulos de libros, congresos de carácter nacional e internacional. Tiene 2 patentes concedidas, cuatro derechos de autor, ha co-editado 20 libros; ha impartido más de 320 conferencias. Entre los reconocimientos más importantes destacan los siguientes galardones: Presea Lázaro Cárdenas en su categoría de investigador en 2001 y el Galardón Honorífico Universitario "Enrique Díaz de León" de la Universidad de Guadalajara en 2008, Cátedras Patrimoniales "Juan Humberto Sossa Azuela" en la Universidad Autónoma de Ciudad Juárez en 2017 y la Universidad del Valle de Atemajac en 2019, primer Premio IMPI como inventor mexicano en la categoría "Patente" en 2020. Sus líneas de investigación son: IA, aprendizaje para máquinas con aplicaciones en control de robots y manejo de información para toma de decisiones.

**Dr. Fernando Reyes Cortés**: Desde 1980 está en la Benemérita Universidad Autónoma de Puebla (BUAP). Es Profesor-Investigador de la Facultad de Ciencias de la Electrónica, BUAP. En 1984 obtuvo la Licenciatura en Ciencias de la Electrónica por la Facultad de Ciencias Físico-Matemáticas, BUAP; en 1990 obtuvo la Maestría en Ciencias en el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) y en 1997 el grado de Doctor en Ciencias en Electrónica y Telecomunicaciones por el Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE). Pertenece al SNI-2. Premio Estatal Puebla y Mérito Civil en Ingeniería por el Ayuntamiento de la Ciudad de Puebla. Autor de múltiples artículos científicos nacionales e internacionales; tiene más de 250 alumnos graduados de licenciatura, maestría y doctorado; ha diseñado y puesto en marcha a más de 50 prototipos robóticos; cuenta con 6 títulos de patentes y más de 300 conferencias. Líneas de investigación: control de robots manipuladores, modelado dinámico, visual-servoing e identificación paramétrica.

# Dedicatoria

## Juan Humberto Sossa Azuela

A mi esposa Rocío, por su amor y comprensión; a la memoria de mis padres Juan Humberto (†) y Carmen (†), por el apoyo incondicional que siempre me dieron para ser la persona que ahora soy; a mis hermanos Julia Martha, Francisco Javier (†) y Miguel Ángel, con los que siempre pasé momentos muy bellos.

## Fernando Reyes Cortés

A mi familia, por todo el amor, apoyo, comprensión y paciencia recibida: a mi esposa Silvia, mis hijos LuisFer, Leo y a mi madre Alicia (†).

A mi Alma Máter:

Benemérita Universidad Autónoma de Puebla

# Agradecimientos

Los autores deseamos agradecer al CONACYT, Instituto Politécnico Nacional (IPN) por el apoyo brindado para concretar esta obra, particularmente al Dr. Mario Alberto Rodríguez Casas, Director General; al Dr. Juan Silvestre Aranda Barradas, Secretario de Investigación y Posgrado y al Dr. Marco Antonio Moreno Ibarra, Director del Centro de Investigación en Computación. Asimismo, a la Benemérita Universidad Autónoma de Puebla por todo el apoyo proporcionado, particularmente reconocemos al Dr. Alfonso Esparza Ortiz, Rector de la Institución, al Dr. Ygnacio Martínez Laguna Vicerrector de Investigación y Estudios de Posgrado; al Dr. José Ramón Eguibar Cuenca, Director General de Investigación de la VIEP y a la Dra. Luz del Carmen Gómez Pavón, Directora de la FCE-BUAP.

Por otro lado, también es importante agradecer al Director Editorial de Alfaomega Grupo Editor, Marcelo Grillo; a Damián Fernández, Editor de esta obra; a Ivonne Nicthe Guzmán Pacheco por el diseño de la portada; a la LDG Rocío Lizeth Escobar Arriaga por el diseño de portadas en capítulos interiores y a Valentina Tolentino por sus valiosas sugerencias en la revisión de estilo.

Juan Humberto Sossa Azuela
Centro de Investigación en Computación
Instituto Politécnico Nacional

Fernando Reyes Cortés
Facultad de Ciencias de la Electrónica
Benemérita Universidad Autónoma de Puebla

Puebla, Puebla. México, a 20 de agosto de 2020.

# Mensaje del Editor

Una de las convicciones fundamentales de Alfaomega y Marcombo es que los conocimientos son esenciales en el desempeño profesional, ya que sin ellos es imposible adquirir las habilidades para competir laboralmente. El avance de la ciencia y de la técnica hace necesario actualizar continuamente esos conocimientos; de acuerdo con esto, Alfaomega Grupo Editor y Marcombo publican obras actualizadas, con alto rigor científico y técnico, y escritas por los especialistas del área respectiva más destacados.

Consciente del alto nivel competitivo que debe de adquirir el estudiante durante su formación profesional, Alfaomega y Marcombo aportan un fondo editorial que se destaca por sus lineamientos pedag´ogicos que coadyuvan a desarrollar las competencias requeridas en cada profesión específica.

De acuerdo con esta misión, con el fin de facilitar la comprensión del contenido de esta obra, cada capítulo inicia con el planteamiento de los objetivos del mismo y con una introducción en la que se plantean los antecedentes y una descripción de la estructura lógica de los temas expuestos, asimismo a lo largo de la exposición se presentan ejemplos desarrollados con todo detalle y cada capítulo concluye con un resumen y una serie de ejercicios propuestos.

Además de la estructura pedagógica con que están diseñados nuestros libros, Alfaomega y Marcombo hacen uso de los medios impresos tradicionales en combinación con las Tecnologías de la Información y las Comunicaciones (TIC) para facilitar el aprendizaje. Correspondiente a este concepto de edición, todas nuestras obras tienen su complemento en una página Web en donde el alumno y el profesor encontrarán lecturas complementarias así como programas desarrollados en relación con temas específicos de la obra.

Los libros de Alfaomega y Marcombo están diseñados para ser utilizados en los procesos de enseñanza-aprendizaje, y pueden ser usados como textos en diversos cursos o como apoyo para reforzar el desarrollo profesional, de esta forma Alfaomega y Marcombo esperan contribuir así a la formación y al desarrollo de profesionales exitosos para beneficio de la sociedad, y esperan ser sus compañeras profesionales en este viaje de por vida por el mundo del conocimiento.

# Contenido

## Capítulo 5
## Reconocimiento de objetos 161

# Plataforma de contenidos interactivos

Para tener acceso al material de la plataforma de contenidos interactivos de la obra siga los siguientes pasos:

**Procedimiento del registro**

1) Ir a la página

   www.**marcombo.info**

2) Acceder con el código promocional

   **ARTIFICIAL2**

# Página Web del libro

**Recursos disponibles en el sitio Web de esta obra**

Código fuente en **MATLAB** (versión 2020b) de todos los ejemplos desarrollados en la obra.

Código fuente en **MATLAB** para ejemplos adicionales y complementarios.

Ejemplos analíticos adicionales resueltos a detalle con sus respectivas simulaciones ilustrativas.

Descargar la información complementaria y adicional de cada capítulo y respaldar en algún directorio o carpeta previamente definida por el usuario en su equipo de cómputo.

# Prólogo

La Inteligencia Artificial (IA) es la ciencia y la ingeniería de las máquinas que actúan de manera inteligente, de las máquinas capaces de: 1) Tomar decisiones apropiadas en circunstancias inciertas y 2) Mejorar su comportamiento con base en sus experiencias. La IA es una disciplina con un nivel de madurez muy alto. No se puede poner en tela de duda su potencial, aplicabilidad e impacto en nuestra sociedad. Prácticamente todos en nuestras casas, lugares de trabajo o públicos, etcétera, sin darnos cuenta utilizamos dispositivos inteligentes que hacen la vida más fácil, amplían nuestras capacidades, nos liberan de tareas peligrosas, engorrosas y cansadas; en pocas palabras, brindan una mejor calidad de vida.

Al revisar la literatura especializada, periódicos y diferentes sitios de Internet, nos damos cuenta de que los avances logrados hasta el momento en este campo del saber permiten a la gente utilizar sistemas capaces de tomar decisiones atinadas en situaciones que antes hubiera sido muy difícil, incluso imposible lograrlo. Como ejemplo podemos referirnos a los llamados chatbots, un tipo particular de robots de servicio, no necesariamente electromecánicos, que permite seleccionar, por ejemplo, el mejor lugar para comer o cenar, la película a ver la próxima semana, qué libro comprar; todo esto conforme a nuestros gustos, o simplemente para completar un trámite en alguna entidad del gobierno.

En el caso particular de robots asistenciales, sabemos que se trata de máquinas con capacidades específicas para interactuar con los seres humanos, atender algunas de nuestras necesidades, para auxiliar, por ejemplo, a una persona con discapacidad en su alimentación, baño, etcétera. En las fábricas, los robots cooperantes o cobots auxilian a los seres humanos a realizar tareas difíciles con menor riesgo y con una eficiencia y eficacia que no se lograrían con un solo grupo de personas. En materia de vigilancia y seguridad, grupos de robots aéreos, submarinos y terrestres ya monitorizan los accesos, las acciones y demás actividades que pudieran ocurrir en espacios cerrados y abiertos. Un uso inmediato de los robots inteligentes voladores puede ser el robo de combustible, la monitorización de temperaturas en personas por posible contracción de COVID-19, entre otros.

En muchos países, la IA es un campo de investigación y aplicación con avances notorios. Esta tecnología horizontal coadyuva al desarrollo sustentable de tales países. Constituye un espacio de grandes oportunidades para el desarrollo científico, pero sobre todo tecnológico e innovador para la apertura de nuevos mercados y negocios.

La IA ha transitado por diferentes concepciones e intereses. A principios de 1950, la investigación en IA pretendía reproducir procesos de inteligencia humana con el objetivo de dar soluciones a problemas formulados en ambientes controlados, por ejemplo, en la prueba de teoremas y la solución de juegos de estrategia con reglas previamente definidas, como las damas y el ajedrez. En los últimos años, la IA se ha expandido y diversificado hasta ser considerada como una rama de las ciencias orientada a la creación de máquinas inteligentes, con habilidades para aprender, adaptarse y actuar con autonomía, como los vehículos autónomos.

El nivel de madurez en IA que se ha adquirido, confiere a esta disciplina la calidad de una ingeniería, como sucedió en su momento con la electrónica. Al principio, esta disciplina era un campo de investigación para los físicos. Con el tiempo, como se sabe, se transformó en una ingeniería, coadyuvando en la creación de carreras tan solicitadas como: Ingeniería electrónica o la Ingeniero en comunicaciones y electrónica. Los problemas que actualmente la IA aborda se relacionan con el manejo de grandes volúmenes de datos para derivar información útil en la generación automática de algoritmos para la solución de problemas complejos asociados al razonamiento, la percepción, la planificación, el aprendizaje y la habilidad de manipular objetos.

Debido a que la IA, como ya se dijo, impacta en nuestros días prácticamente en cualquier área del saber y sector: salud, seguridad, manejo energético, medio ambiente, educación, etcétera, actualmente se relaciona también con muchas carreras, en particular aquellas que la incorporan como una materia opcional u obligatoria; sobre todo en las instituciones que se han dado a la tarea de diseñar y echar a andar profesiones como la de ingeniero en IA, ingeniero en sistemas computacionales, etcétera.

En carreras profesionales como la Ingeniería en sistemas computacionales, Licenciatura en informática y Licenciatura en ciencias de la computación, la Inteligencia Artificial como curso se puede estudiar en los últimos semestres o cuatrimestres como opcional u obligatoria. Sin embargo, en carreras como Ingeniería en Inteligencia Artificial, que actualmente se imparte en la Universidad Panamericana y el Instituto Politécnico Nacional, la IA es una materia de carácter obligatorio. En todos los casos, la Inteligencia Artificial se está volviendo una materia indispensable para profesores y alumnos, sobre todo por sus potenciales aplicaciones. Los egresados de carreras en donde la Inteligencia Artificial y cursos complementarios forma parte del currículo, compiten fuertemente con egresados de otras disciplinas, obteniendo salarios bastante competitivos y en muchos casos muy superiores.

# Organización del libro

De manera general, el libro cubre cuatro grandes partes. Primeramente, una introducción al tema de la IA. Enseguida, una introducción al Aprendizaje para Máquinas como principal motor de la IA. Después, una descripción de un conjunto de técnicas representativas para el tratamiento y análisis digital de imágenes y el reconocimiento de patrones. Finalmente, una introducción a las poderosas redes neuronales artificiales para la solución de problemas complejos. La presente obra se organiza en siete capítulos:

En el **primer capítulo** se exponen las diferencias entre el cómputo inteligente y el cómputo tradicional. También se ofrece un estado del arte relativo al campo de la IA, se define el término IA y se enumeran sus diferentes categorías. Más adelante, se presenta una breve historia de la IA, así como de los principales hechos que a lo largo de la historia le han dado la forma que ahora tiene.

Después, se habla de algunas de sus aplicaciones en diversas áreas, así como del conjunto de líneas de generación de conocimiento y de su aplicación. Enseguida, se abordan los retos que todavía enfrenta la IA y de por qué, a pesar de estos retos, se puede asegurar que ya es su momento. Más adelante se dan unas palabras acerca de los dos grandes enfoques para la construcción de sistemas inteligentes y de la ética en su uso. Finalmente, se presentan algunas estadísticas relacionadas.

En el **segundo capítulo** se da la definición de Aprendizaje para Máquinas (AM), como principal motor de la IA. Enseguida, se describe el proceso de cómo el AM opera. Después, se describe cada uno de los cuatro tipos de aprendizaje que se pueden dar en una máquina, a saber: aprendizaje supervisado, aprendizaje semi-supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Más adelante, se presenta un conjunto de definiciones y conceptos provenientes del álgebra lineal. Después, se describe el tema central de este capítulo: regresión lineal. Se expone cómo esta técnica puede ser utilizada para resolver problemas de predicción, en particular en el campo de la robótica.

En el **tercer capítulo** se estudia primeramente, el problema de la clasificación de patrones a través del método de discriminación lineal, el cual contempla la probabilidad del número de clases. La construcción de la función de probabilidad se lleva a cabo de manera indirecta, a través del conocido teorema de Bayes. Después, se modela la función de probabilidad

a posteriori mediante distribuciones gaussianas multivariables. Enseguida, se construyen las correspondientes funciones de discriminación lineal. Finalmente, dichas funciones de discriminación son estimadas y utilizadas.

Después, se presenta el problema de regresión logística buscando elegir la clase con la probabilidad más alta. En lugar de estimar la probabilidad de manera indirecta como en el caso de discriminación lineal, ahora se determina de manera directa a través del cálculo del logaritmo del cociente de probabilidades de una clase y su complemento, lo cual permite obtener la llamada función de verosimilitud a través del método de Newton-Raphson multivariable.

En el **cuarto capítulo** se presentan los conceptos fundamentales, así como las técnicas básicas de filtrado relacionadas con el tratamiento y procesamiento digital de imágenes. Primeramente, se ofrecen los conceptos de imagen, imagen digital, imagen binaria e histograma de una imagen. En todos los casos se muestran ejemplos alusivos para que el lector asimile rápidamente y de manera sencilla dichos conceptos. Enseguida, se dan los detalles sobre el acondicionado de una imagen. Primeramente, se expone el concepto de ruido en una imagen y se dan ejemplos ilustrativos. Después se habla del proceso de filtrado como herramienta computacional para reducir el efecto del ruido causado en una imagen. Se muestra la operación de tres filtros muy usados por la comunidad en el acondicionado de una imagen: promedio aritmético, gaussiano y mediano.

En el **capítulo quinto** se estudia cómo una vez acondicionada una imagen, ésta puede ser segmentada en sus diferentes componentes. Se expone cómo este problema es resuelto mediante el paradigma del umbralado de imágenes en forma manual y de manera automática. Se describe la operación de los métodos de Otsu y el de mínimo error. Después se estudian varios métodos para el post-tratamiento de imágenes segmentadas para reducción del ruido residual debido a un pobre umbralado, así como varias técnicas para etiquetado de regiones conectadas en imágenes binarias.

Posteriormente, se revisan varios rasgos útiles para describir propiedades geométricas y topológicas de los objetos segmentados. Más adelante, se describe un conjunto de técnicas para detectar la presencia de objetos y para su identificación en escenas a partir de imágenes o secuencias de imágenes. Entre otros los basados en el cálculo de distancias y los estadísticos bayesianos.

En el **capítulo seis** se estudia uno de los paradigmas más utilizados actualmente en la solución de problemas de clasificación y reconocimiento de patrones, la predicción de series de tiempo, el modelado de sistemas y la optimización, el relacionado con las llamadas redes neuronales artificiales, que intentan simular la operación de las neuronas biológicas. Después se presentan los pormenores de las arquitecturas básicas de redes neuronales más utilizadas en la literatura actual. Se presenta la unidad de umbralado lógico (UUL) y la forma que puede ser entrenada a través de la regla del perceptrón para resolver problemas de diversa índole. Enseguida, se dan los detalles de la operación y funcionalidad del conocido perceptrón como extensión de la UUL. Posteriormente, se habla de la neurona lineal ADALINE y cómo ésta puede ser entrenada mediante la llamada regla Delta, basada en el principio del descenso del gradiente. Finalmente, se habla de la neurona sigmoidal y la manera en cómo es entrenada para la solución de diversos problemas.

Finalmente, en el **capítulo siete**, se continúa con el estudio del perceptrón como máquina de procesamiento de información, pero ahora acomodado en capas. Se presentan los detalles sobre la derivación de la regla Delta generalizada, útil para el entrenamiento de arreglos de perceptrones en capas para la solución de problemas no lineales. Enseguida, se explica la función de la capa intermedia de transformar un problema no lineal de clasificación en uno lineal para que el último o últimos perceptrones, que son máquinas lineales, puedan resolver el problema. Después, se detalla cómo al agregar el término de momento en la ecuación de ajuste de pesos de la red neuronales, se consigue acelerar dicho proceso de ajuste. Por último, se ofrecen las bases de operación de las llamadas redes neuronales convolucionales. Esto se hace a través de ejemplos, lo cual permite que el lector aprecie de manera sencilla las capacidades de esta poderosa máquina de procesamiento.

## Ejemplos

Los ejemplos ilustrativos que se encuentran resueltos en la obra están clasificados con respecto a su grado de complejidad en cuatro formas posibles, como a continuación se describen:

### Ejemplo 1.1

Es un ejemplo sencillo o básico que ilustra un concepto o definición. No se requiere hacer operaciones complejas ni tiene asociada programación. Contiene un contador de ejemplo para hacer referencia posteriormente.

**• Ejemplo 1.2**

Este es un ejemplo simple, la problemática a resolver se encuentra en un recuadro con fondo gris. Como parte de la simbología se utiliza una línea negra horizontal para indicar el inicio de la solución propuesta.

**Solución**

Los ejemplos simples describen un planteamiento básico que se resuelve con la formulación adecuada, descripción de conceptos; puede llevar código fuente MATLAB en programas documentados para ilustrar los resultados. Posee un contador de ejemplo para hacer la referencia correspondiente.

**•• Ejemplo 1.3**

Representa un ejemplo regular, con esta notación indica que el grado de complejidad se incrementa con respecto al ejemplo simple.

**Solución**

Esta categoría corresponde a un ejemplo regular, la solución propuesta se desarrolla en forma analítica, explicando paso a paso los detalles matemáticos y también incluye programación en MATLAB para respaldar e ilustrar la interpretación del problema por medio de simulación. Como en los casos anteriores tiene un contador específico del ejemplo.

**••• Ejemplo 1.4**

Ejemplo complejo: Es notablemente mucho más complicado que los anteriores, representa el nivel más alto de abstracción de los ejemplos resueltos en la obra.

**Solución**

Se plantea una problemática cuya solución no es trivial; requiere asimilar los conocimientos expuestos. El desarrollo de la solución es presentada con respaldo analítico en forma clara y detallada; la documentación se realiza en forma completa. Se puede hace referencia a este ejemplo por medio de su contador asociado.

## Programas

Los programas de este libro se encuentran en cuadros de código con número de referencia, cabecera con descripción, nombre del programa, número de línea de programación dentro del código que facilita la explicación y documentación, como el que se muestra a continuación.

**Código MATLAB 4.1 cap4_Histograma.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 4. Tratamiento de imágenes.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap4_Histograma.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; %remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
f=imread( 'cap4_imaprueba1.png' ); % lee imagen de prueba f(x,y).
figure(1)
imshow(f) % muestra imagen de prueba (ver figura 4.4).
title( 'Imagen original' )
[n, m]=size(f); % dimensiones de la imagen f(x,y).
c=zeros(256,1); % inicializa en ceros el registro del contador c(i).
niveles=1:256; % niveles de gris.
for x=1:n  % barrido en renglones de la imagen f(x,y).
    for y=1:m % barrido en columnas de la imagen f(x,y).
        i=f(x,y); % obtiene el nivel de gris de f(x,y).
        c(i)=c(i)+1;  % contador de píxeles a niveles de gris.
    end
end
figure(2) % (ver figura 4.4).
fill(niveles,c, 'b' ); % dibuja histograma h(r_i) de la imagen f(x,y).
title( 'Histograma original' ); figure(3)
[Contador_matlab,nivel_gris] = imhist(f);  % hist., con función MATLAB.
fill(nivel_gris,Contador_matlab, 'x' , niveles, Contador, 'O' );
title( 'Histograma con función de Matlab' );
```

Todos los programas de esta obra se encuentran disponibles en: www.alfaomega.com.mx

## Créditos de programación y paquetes de cómputo

En la presente obra se han tomado en cuenta los aspectos de calidad científica y académica docente, así como los aspectos pedagógicos de la exposición de conceptos, secuencia y desarrollo del contenido, solución de ejemplos ilustrados y con el fortalecimiento de los conocimientos a través de programación y simulación para una mejor interpretación y comprensión del lector. Por tal motivo, la selección de herramientas y paquetes de cómputo de calidad es fundamental. En consecuencia:

### Créditos de programas y herramientas de cómputo utilizados

En este libro se ha priorizado la calidad de presentación no solo en la exposición de los conceptos, también la estética y estilo de objetos pedagógicos y gráficos con la finalidad de captar y motivar la atención de alumnos y profesores. Por tal motivo, la presente obra fue editada, formada y compilada en lenguaje científico LaTeX empleando programación y macros diseñados y desarrollados por los autores, a través de MiKTeX versión 2.9 y TeX*studio* -2.12.22, **MATLAB**® versión 2020b, así como las versiones 2020 de AUTOCAD® y **SOLID**Works®.

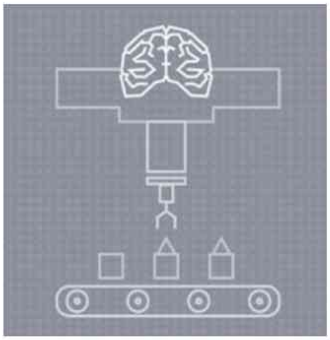
*Dr. Juan Humberto Sossa Azuela*
Centro de Investigación en Computación
Instituto Politécnico Nacional

*Dr. Fernando Reyes Cortés*
Facultad de Ciencias de la Electrónica
Benemérita Universidad Autónoma de Puebla

Ciudad de México a 20 de agosto de 2020.

Capítulo 1

# Introducción a la Inteligencia Artificial



# Descripción del capítulo

Es este capítulo primeramente se darán las principales diferencias entre el cómputo basado en la Inteligencia Artificial (IA) y el cómputo tradicional. Después se dará un estado del arte relativo al campo de la IA. Enseguida se definirá el término IA y se enumerarán sus diferentes categorías. Más adelante, se presentará una breve historia de la IA, así como de los principales hechos que a lo largo de la historia le han dado la forma que ahora tiene. Después, se mencionará algunas de sus aplicaciones en diversas áreas, así como el conjunto de líneas de generación de conocimiento y de su aplicación. Enseguida se hablará acerca de los retos que todavía enfrenta la IA y de por qué, a pesar de ellos, se puede asegurar que ya es su momento. Más adelante se hará mención acerca de los dos grandes enfoques para la construcción de sistemas inteligentes y de la ética en su uso. Después se presentarán algunas estadísticas relacionadas, y se dará una serie de conclusiones. Al final, el lector encontrará un conjunto de problemas propuestos.

Los siguientes temas serán abordados:

Cómputo tradicional *versus* cómputo IA.

Historia de la IA.

El estado del arte en IA.

Aplicaciones comerciales de la IA.

Reglas de ética para el uso de la IA.

Estadísticas de la IA.

# 1.1 Introducción

La Inteligencia Artificial (IA) es hoy en día, una disciplina con un nivel de madurez muy alto. No se puede poner en tela de juicio su potencial, aplicabilidad e impacto en nuestra sociedad. Prácticamente todos en nuestras casas, oficinas, negocios, escuelas, etcétera, sin darnos cuenta, utilizamos algún dispositivo "inteligente" que nos hace la vida más fácil, extiende nuestras capacidades, nos libera de actividades engorrosas, nos da seguridad; en resumen, nos brinda una mejor calidad de vida.

Los avances logrados hasta el momento en IA permiten que un gran número de personas, hogares y empresas cuenten con sistemas capaces de auxiliarles en la toma de decisiones en situaciones que antes hubiera sido difíciles e incluso imposible, lograr. Entre otros podemos referirnos a los sistemas consejeros, los llamados **chatbots**, etcétera. En ambos casos, se trata de un tipo particular de robot de servicio no necesariamente electromecánico que permite seleccionar, por ejemplo, el mejor lugar para comer, la película a ver la próxima semana, qué libro comprar; todo esto de acuerdo con los gustos, o simplemente para completar un trámite.

En el caso particular de robots asistenciales se trata de máquinas con capacidades especiales para interactuar con los seres humanos, para atender algunas de sus necesidades, para auxiliar, por ejemplo, a una persona con una incapacidad, en su alimentación, etcétera. En las fábricas los robots cooperantes (**cobots**) auxilian a los seres humanos a realizar las tareas difíciles con menor riesgo y con una eficiencia y eficacia que no se lograrían con únicamente grupos de personas. En materia de vigilancia y seguridad, grupos de robots aéreos, submarinos y terrestres ya monitorizan los accesos, las acciones y demás actividades en espacios cerrados y abiertos. Un uso inmediato de los robots inteligentes voladores pudiera ser el rastreo del combustible.

La IA en muchos países es un campo de investigación y aplicación con bastante avance. Esta tecnología coadyuva con el desarrollo sustentable de estos países. Constituye un espacio de grandes oportunidades para el desarrollo científico, pero sobre todo tecnológico e innovador para la apertura de nuevos mercados y de nuevos negocios. La IA ha transitado por diferentes concepciones e intereses. En sus primeros años (década de 1950), la investigación en IA pretendía reproducir procesos de inteligencia humana para dar soluciones a problemas

formulados en ambientes controlados, por ejemplo, en prueba de teoremas y la solución de juegos estratégicos con reglas previamente definidas. En los últimos años la IA se ha expandido y diversificado hasta ser considerada como una rama de las ciencias orientada a la creación de máquinas inteligentes, con habilidades para aprender, adaptarse y actuar con autonomía.

Por otro lado, el nivel de madurez en IA le confiere la calidad de una ingeniería como sucedió en su momento con la electrónica, que al principio era un campo de investigación para los físicos. Con el tiempo se transformó en una ingeniería, coadyuvando en la creación de carreras tan solicitadas como Ingeniería Electrónica e Ingeniería en Comunicaciones y Electrónica. Los problemas que aborda actualmente la IA están relacionados con el manejo de grandes volúmenes de datos para derivar información útil en la generación automática de algoritmos en la solución de problemas complejos asociados con el razonamiento, la percepción, la planificación, el aprendizaje y la habilidad de manipular objetos.

### 1.1.1 Diferencias del cómputo IA y tradicional

Mucha gente confunde el cómputo basado en IA respecto de aquel fundamentado en el cómputo tradicional. Para entender esta diferencia, consideremos la siguiente pregunta: ¿Por qué hay ciertos problemas que son muy difíciles de resolver a través del cómputo tradicional? Recuérdese que el cómputo clásico, en sus inicios, fue diseñado para hacer dos cosas muy bien:

Realizar operaciones aritméticas de manera muy rápida y seguir de forma explícita un listado de instrucciones.

**Figura 1.1:** Base de datos.

Si por ejemplo, se quiere procesar grandes cantidades de datos financieros, un programa tradicional de cómputo puede lograr el objetivo. Analicemos un problema ligeramente diferente. Se trata del reconocimiento de los números de 0 al 9 (base de datos MNIST), como se ilustra en la figura 1.1. Cada dígito en la primera fila puede ser reconocido como un "0", cada dígito en la segunda fila como un "1", y así sucesivamente. ¿Qué conjunto de reglas deberían usarse para distinguir entre caracteres? Por ejemplo, para el "0", una primera regla podría ser que este se represente por un ciclo cerrado de puntos. De la figura 1.1 se puede observar que esta regla en algunos casos funciona y en otros no.

Algo parecido puede decirse del resto de los nueve caracteres. En el proceso de deducción de reglas, es posible adicionar más y más reglas a través de un análisis profundo por días, semanas o meses. No es claro que al final se encuentre una solución. Es evidente que algo se nos escapa. La realidad es que muchos problemas caen en esta categoría:

Reconocimiento de objetos.

Comprensión de lenguaje natural.

Traducción automática, etcétera.

**Hecho**: Para este tipo de problemas no sabemos qué programas escribir, ya que como seres humanos no sabemos cómo resolver este tipo de problemas. Más aún, si lográramos escribir dicho programa, seguramente este sería terriblemente complicado.

En esta nueva revolución (en los últimos 20 años), cada vez observamos más programas que aprenden. Programas que adaptan su comportamiento de manera automática a los requerimientos de una tarea dada; programas que reconocen caras, conducir coches, recomiendan qué libros comprar o qué películas ver, etcétera. Antes, una persona (un programador) definía lo que un ordenador debería de hacer al codificar un algoritmo en un lenguaje de programación.

Hoy, en lo que se conoce como inteligencia artificial moderna o aprendizaje para máquinas, ya no se escriben programas (en el sentido clásico del término); en su lugar lo que se hace es recolectar datos. Esta información contienen instancias de lo que se debe hacer y un algoritmo de aprendizaje modifica (de forma automática) un programa capaz de aprender, de manera que compagine los requerimientos suministrados por los datos.

Para muchas aplicaciones, desde visión artificial al habla o de la traducción a la robótica no estándar, el ser humano no es capaz de diseñar buenos algoritmos a pesar de décadas de investigación, desde 1950. Sin embargo, para estas tareas es muy fácil recolectar datos. La idea en este caso es aprender en forma automática algoritmos a partir de los datos, reemplazando así al programador por programas que aprenden. Este es el nicho del Aprendizaje para Máquinas (AM) o en el caso más general, de la IA. La idea entonces, del AM es convertir estos datos en conocimiento.

Hoy por hoy, la gente que trabaja en IA busca formas de:

- Hacer uso de datos.
- Convertir estos datos en productos o servicios útiles para el ser humano.

Así por ejemplo, cuando pensamos acerca de los millones de personas que a diario compran miles de productos en línea o en las diferentes cadenas de supermercados, esto implica enormes cantidades de datos derivados de las correspondientes transacciones. Aunque difíciles de encontrar, por supuesto que hay patrones escondidos. Uno de estos permite pensar que la gente no compra al azar.

Cuando alguien compra un tipo de producto, usualmente compra también otro que se relaciona. Hay un conjunto de factores escondidos que explican el comportamiento del comprador. El objetivo de la IA es la inferencia del modelo escondido (en los datos), esto es: el conjunto de factores relacionados y su interacción a partir de dichos datos observados.

El aprendizaje es de hecho un requisito de la inteligencia. En este sentido, un sistema se dice ser inteligente si es capaz de adaptarse al medio ambiente. Debe aprender a no repetir sus errores sino sus éxitos.

En resumen, la diferencia principal entre la computación clásica y la inteligencia artificial es el tipo de problemas a resolver y, por tanto, el conjunto de técnicas a utilizar.

Las siguientes serían algunas de las diferencias entre el cómputo basado en IA y el cómputo tradicional. En el caso del cómputo basado en IA, los problemas a resolver son aquellos que:

- No se resuelven con un algoritmo existente o que se puede concebir y codificar como un programa, sino a través de un programa capaz de aprender.
- Donde la cantidad de datos no sólo es grande, sino que su análisis es también difícil de llevar a cabo.

Para los cuales es muy difícil o imposible encontrar el conjunto total de reglas para su solución. Los datos serán usados para derivar de manera automática los algoritmos que den solución a esos problemas.

Donde la incertidumbre sea alta, no es posible tener todos los datos, o bien, el ruido asociado puede ser grande.

Son difíciles de resolver, la conducción automática de coches (autos), la traducción automática entre idiomas, reconocimiento de rostros, etc.

En cuanto a la robótica clásica y la robótica basada en IA, se puede decir que la primera se encarga del diseño y puesta en operación de sistemas electromecánicos comandados por leyes de control con retroalimentación de las señales obtenidas de los sensores conectados a la máquina. No requieren de una inteligencia de alto nivel, ya que no se desempeñan en ambientes no controlados, parcialmente desconocidos o completamente desconocidos.

Los llamados robots inteligentes, por otro lado, pueden ser máquinas o sistemas de cómputo no convencional que al principio cuentan con un conocimiento mínimo del problema a resolver. Estas máquinas o sistemas inteligentes adaptan o mejoran su desempeño conforme adquieren más datos, su arquitectura *hardware* o *software* es la misma, pero son sus parámetros internos los que cambian para que el robot lleve a cabo cada vez mejor su correspondiente tarea. Una máquina de este tipo requiere algoritmos de bajo nivel, perfectamente bien probados para obtener un movimiento exacto, por ejemplo, a nivel articular, usando esquemas de control con alto desempeño.

Por otro lado, si la máquina física o programa requiere llevar a cabo tareas de razonamiento, planificación, etcétera, demanda el uso de programas capaces, por un lado, de aprender y, por otro lado, de adaptarse a las circunstancias seguramente cambiantes de su medio ambiente.

En resumen, la diferencia principal entre la robótica clásica (aquella que hace uso de algoritmos clásicos o estándar, esto es que no utilizan aprendizaje automático) y la no convencional (aquella que demanda de la operación de algoritmos que utilizan inteligencia artificial) reside de nuevo en el tipo de problemas a resolver y, por tanto, en el conjunto de técnicas a utilizar.

Las siguientes serían algunas de las diferencias entre la robótica basada en IA y la robótica clásica. La robótica basada en IA:

- No se especializa en el diseño y puesta en operación de electromecánica de las máquinas, sino más bien en su *software*.
- Se centra en el diseño de sistemas de análisis de grandes cantidades de datos para el diseño y puesta en operación de sistemas para el reconocimiento de patrones, para el entendimiento del lenguaje natural, las imágenes o el vídeo, por ejemplo.
- Se orienta al diseño y puesta en operación de sistemas para la predicción y para la recomendación.

## 1.2 Estado del arte sobre la IA

Hablando de IA, hoy por hoy esta tecnología está influyendo fuertemente en los puestos de trabajo de prácticamente todas las industrias. Durante siglos se ha visto a la automatización como algo maligno que amenaza nuestros trabajos y altera el *statu quo*. Esto ha sido notorio durante las Primera, Segunda y Tercera Revoluciones Industriales, como se ilustra en la figura 1.2.

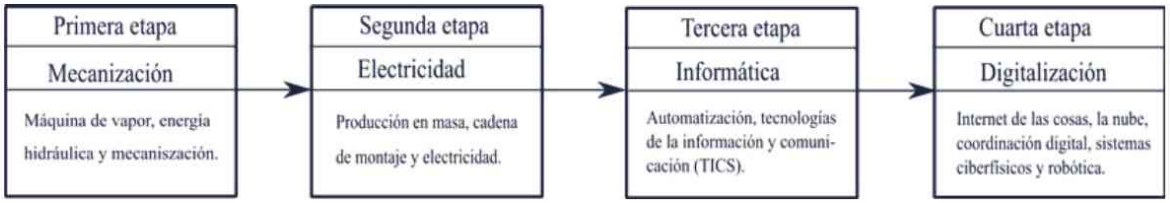



**Figura 1.2**: Etapas o revoluciones por las cuales ha avanzado el proceso de la automatización.

La introducción de la automatización ha impulsado a los trabajadores a aprender nuevas habilidades para adaptarse a las nuevas circunstancias. La automatización sigue creciendo y generando beneficios para las empresas en todo el mundo. Esto se hace patente, actualmente, en la llamada Revolución 4.0 o Industria 4.0, donde se conjuntan las siguientes tecnologías:

Internet de las cosas (IOT) y sistemas ciberfísicos.

Grandes datos, minería de datos y analítica de datos.

Simulación y fabricación aditiva (impresión 3-D).

Sistemas de integración horizontal y vertical.

Ciberseguridad.

Realidad virtual y la realidad aumentada.

Cómputo en la nube.

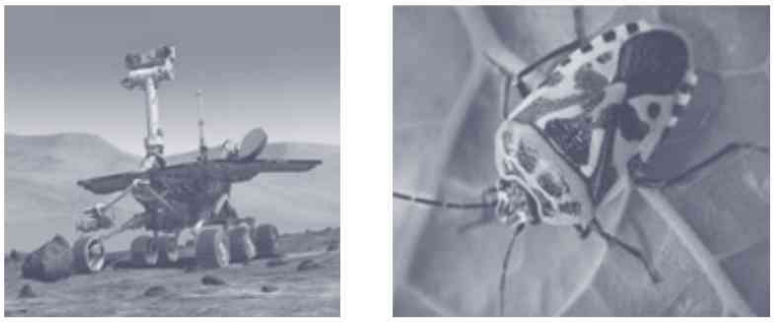
Robótica autónoma y colaborativa (cobots).

La automatización ha ido cambiando los lugares de trabajo de muchas maneras. Ha hecho obsoletos algunos tipos de trabajos, provocando la aparición de otros. Para obtener un empleo de tecnología, un trabajador debe desarrollar un nivel sofisticado de habilidades en temas difíciles.

> En esta Cuarta Revolución Industrial surge un nuevo actor: La Inteligencia Artificial. Esta puede ayudar mucho para facilitar, entre otras cosas, el proceso de aprendizaje y simplificar tareas computacionales, quitando parte de la carga a su homólogo humano.

Por otro lado, los robots han atraído la atención de mucha gente. Los vemos repetidamente en las películas de ciencia ficción o en los libros de la misma temática. Como se nos presentan estas máquinas en las películas o libros, se cree que es fácil construir artefactos inteligentes ¡La realidad es otra! En todas estas situaciones hay un pequeño truco: ¡Hay una persona detrás que habla a través de un micrófono! Los robots más avanzados de Marte (figura 1.3a) poseen, si acaso, la inteligencia de un insecto (figura 1.3b).

Una cucaracha o una rata pueden hacer mucho más. ¡Pueden maniobrar sin problema dentro de un recinto lleno de cosas, pueden encontrar escondites y reconocer el peligro! Ni la máquina más sofisticada diseñada por el ser humano es capaz todavía de entender la más simple historia para niños cuando se le lee ¡Ninguna máquina es capaz todavía de imaginar cómo lo hacemos los seres humanos! Lo mismo es aplicable todavía para cualquiera de los personajes mecánicos que se pueden encontrar en los libros o películas de ciencia ficción que se han escrito o que actualmente se escriben o se dirigen; estamos muy lejos todavía de que esto sea una realidad.



**(a)** Robot explorador tipo ROVER.

**(b)** Insecto.

**Figura 1.3**: Robot explorador tipo ROVER comparado con un insecto.

La IA es sin duda, una de las tendencias definitorias de nuestros días. En los últimos diez años, diversos tipos de máquinas (programas y robots) han sido entrenadas para resolver problemas cada vez más complejos. Actualmente las máquinas son capaces de llevar a cabo tareas solo atribuibles, en otros tiempos, al ser humano, por ejemplo:

- La identificación de personas en medio de la multitud [Web1, Web2, Web3].
- El guiado autónomo de coches [Web4, Web5].
- Los juegos complejos como ajedrez, el go, el Shogi [Web6], el póker [1], StartCraft II [Web7], entre otros.

Es indudable que algunas veces las máquinas hacen mejor las cosas que nosotros los seres humanos. Hay que tener en cuenta que las máquinas pensantes no son nuevas, por supuesto, datan de hace más de 75 años. Entre las máquinas más antiguas se pueden mencionar al

programa que juega damas inglesas diseñado por Christopher Strachey [Web8] y al algoritmo diseñado para jugar ajedrez desarrollado por Dietrich Prinz [Web9]. Ambos programas fueron desarrollados en el ordenador Ferranti Mark 1 de la Universidad de Manchester en el año de 1951.

## 1.3 ¿Qué es la Inteligencia Artificial?

Antes de definir el término Inteligencia Artificial, se debe primero reflexionar acerca de lo que es la inteligencia natural o simplemente inteligencia. Inteligencia es un término vago. Su definición ha dependido del tipo de individuo involucrado, del momento, incluso de las circunstancias.

Platón, por ejemplo, decía que la inteligencia es la actividad que permite adquirir ciencia, lo cual, por supuesto es muy restrictivo. Él propuso dos tipos de inteligencia, pero de acuerdo con Howard Gardner hay nueve [Web10], según la psicología hay doce [Web11]. En resumen, se puede decir que la inteligencia es una capacidad mental muy general que implica:

- Habilidad para razonar.
- Planificar.
- Resolver problemas.
- Pensar de forma abstracta.
- Comprender ideas complejas.
- Aprender con rapidez.
- Aprender de la experiencia.

La IA no supone el simple aprendizaje de un texto, cierta habilidad académica específica, o resolver un test o examen de forma hábil. Más bien, refleja una capacidad amplia y profunda para:

- La comprensión del entorno.
- Ser capaz de capturar el significado de las cosas y darles un sentido, o para ingeniárselas a la hora de saber qué hacer.

**Definición de Inteligencia Artificial**

Tomando esto como base, y para los propósitos de la presente obra, la Inteligencia Artificial puede definirse como la ciencia e ingeniería de las máquinas que actúan de manera inteligente.

En este sentido, una máquina es inteligente cuando es capaz de tomar decisiones apropiadas en circunstancias inciertas. Otra manera de verlo es que una máquina se dice ser inteligente cuando es capaz de aprender a mejorar su comportamiento con base en sus experiencias.

## 1.4 Tipos de Inteligencia Artificial

De acuerdo con estudios como los referenciados en [Web12-Web15], se pueden diferenciar cuatro tipos de IA: 1) Inteligencia General (IG) o inteligencia fuerte, 2) Súper Inteligencia Artificial (SIA), 3) Inteligencia Artificial Débil o aplicada (IAD) y 4) Súper Inteligencia-Súper Consciente IA (SI-SC-IA). Cada una de estas categorías de Inteligencia Artificial se explicará brevemente:

**Inteligencia General**

Inteligencia General (IG) o inteligencia fuerte. Llamada también Inteligencia Artificial General (IAG) (comparada con la inteligencia humana), [9]. Consiste en enseñarle a la máquina a aprender, a razonar y a planificar. No cubre todo el campo de la inteligencia, faltaría agregar a la máquina una mente, así como una conciencia, lo que implica estar consciente y un carácter propio, lo cual no es fácil de simular.

Una posible solución para resolver este problema sería hacer ingeniería inversa del cerebro. Muchos piensan que un 90 % de una inteligencia comparable con la inteligencia humana (Inteligencia Artificial con Nivel Humano, IANH) sería alcanzado durante este siglo, aproximadamente en el año 2075.

**Súper Inteligencia Artificial**

Súper Inteligencia Artificial (SIA). Un paso más allá es la llamada Súper Inteligencia Artificial puede alcanzarse relativamente rápido una vez que se llegue a la IAG, supuestamente porque los mayores obstáculos han sido sobrellevados.

**Inteligencia Artificial Débil**

Inteligencia Artificial Débil o aplicada (IAD). Se limita a afrontar tareas específicas, enfocadas a ayudar al ser humano. No intenta simular el rango completo de las habilidades cognitivas humanas.

Este es el tipo de IA que actualmente se está desarrollando en los ámbitos académico y empresarial. Después de la IAG y la SIA se encuentran otros dos tipos de inteligencia:

**La Súper Inteligencia-Súper Consciente IA**

La Súper Inteligencia-Súper Consciente IA (SI-SC-IA) se refiere a la capacidad que tendrían criaturas existentes en algunos de los muy probables sistemas planetarios en nuestro Universo. Si partimos del hecho de que nuestra galaxia contiene alrededor de 400,000 millones de estrellas, la probabilidad de que una de ellas haya dado origen a vida Súper-Inteligente y Súper-Consciente es muy alta.

**La inteligencia proveniente de seres de nuestra creación**

Este tipo de inteligencia se refiere a aquel nivel de razonamiento, mente y conciencia que pudieran tener personajes creados por un ser humano, por ejemplo Superman, Thor o Capitana Marvel que, como bien sabemos solo existen en la ciencia ficción.

## 1.5 Breve historia de la IA

Una muy breve historia sobre la IA se puede resumir a través de hechos relevantes, como a continuación se describe [Web16]:

384-322 AC: Las ideas más básicas se remontan a Aristóteles (384-322 a. C.). Fue el primero en postular un conjunto de reglas que describen una parte del funcionamiento de la mente para obtener conclusiones racionales.

250 AC: Ctesibio de Alejandría (250 a. C.) construye la primera máquina autocontrolada a través de un regulador del flujo de agua.

1315: Ramon Llull en su libro Ars Magna tiene la idea de que el razonamiento podía ser efectuado de manera artificial.

1936: Alan Turing diseña formalmente una máquina universal que demuestra la viabilidad de un dispositivo físico para implementar cualquier cómputo formalmente definido.

1943: Warren McCulloch y Walter Pitts presentan su modelo de neurona artificial.

1955: Herbert Simon, Allen Newell y J. C. Shaw desarrollan el primer lenguaje de programación orientado a la resolución de problemas, el IPL-11. Un año más tarde desarrollan el LogicTheorist, el cual era capaz de demostrar teoremas matemáticos.

1956: Se inventa el término Inteligencia Artificial por John McCarthy, Marvin Minsky y Claude Shannon en la Conferencia de Dartmouth.

1957: Newell y Simon continúan su trabajo con el desarrollo del General Problem Solver (GPS), el cual era un sistema orientado a la resolución de problemas.

1958: John McCarthy desarrolla en el Instituto de Tecnología de Massachusetts (MIT) LISP. Su nombre se deriva de LISt Processor. LISP fue el primer lenguaje para procesamiento simbólico.

1959: Rosenblatt introduce el Perceptrón.

1959-1961: Robert K. Lindsay desarrolla «Sad Sam», un programa para la lectura de oraciones en inglés y la inferencia de conclusiones a partir de su interpretación.

1963: Quillian desarrolla las redes semánticas como modelo de representación del conocimiento.

1964: Bertrand Raphael construye el sistema SIR (Semantic Information Retrieval) el cual era capaz de inferir conocimiento basado en información que se le suministra. Bobrow desarrolla STUDENT.

1965: Aparecen los sistemas expertos que predicen la probabilidad de una solución bajo un conjunto de condiciones.

1968: Marvin Minsky publica Semantic Information Processing.

1968: Seymour Papert, Danny Bobrow y Wally Feurzeig desarrollan el lenguaje de programación LOGO.

1969: Alan Kay desarrolla el lenguaje Smalltalk en Xerox PARC; se publica en 1980.

1968-1970: Terry Winograd desarrolla el sistema SHRDLU, que permitía interrogar y dar órdenes a un robot que se movía dentro de un mundo de bloques.

1973: Alain Colmenauer y su equipo de investigación en la Universidad de Aix-Marseille crean PROLOG (del francés PROgrammation en LOGique) un lenguaje de programación ampliamente utilizado en IA.

1973: Shank y Abelson desarrollan los guiones o *scripts*, pilares de muchas técnicas actuales en Inteligencia Artificial y la informática en general.

1974: Edward Shortliffe escribe su tesis con MYCIN, uno de los sistemas expertos más conocidos, que asistió a médicos en el diagnóstico y tratamiento de infecciones en la sangre.

1970-1980: Crece el uso de sistemas expertos.

1981: Kazuhiro Fuchi anuncia el proyecto japonés de la quinta generación de ordenadores.

1986: McClelland y Rumelhart publican Parallel Distributed Processing (redes neuronales).

1988: Se establecen los lenguajes orientados a objetos.

1997: Gari Kaspárov, campeón mundial de ajedrez, pierde ante el ordenador autónomo Deep Blue.

2006: Se celebró el aniversario con el Congreso en español 50 años de Inteligencia Artificial - Campus Multidisciplinar en Percepción e Inteligencia 2006.

2009: Se desarrollan sistemas inteligentes terapéuticos que permiten detectar emociones para poder interactuar con niños autistas.

2011: IBM desarrolla un superordenador llamado Watson, el cual gana una ronda de tres juegos seguidos de Jeopardy, venciendo a sus dos máximos campeones.

2016: Un programa informático ganó cinco a cero al triple campeón de Europa de Go.

2018: Se lanza el primer televisor con Inteligencia Artificial por parte de LG Electronics con una plataforma denominada ThinQ.

De todos estos hechos, se podría asegurar que los siguientes ocho, de alguna manera u otra, han dado forma a lo que ahora es la IA [Web17]:

1956: El proyecto de investigación en Inteligencia Artificial en el verano, en Dartmouth, acuña el nombre de un nuevo campo relacionado con el quehacer de *software* habilidoso como los humanos.

1965: Joseph Weisenbaum en MIT crea Eliza, el primer Chatbot útil como psicoterapeuta.

1975: En Stanford se desarrolla Meta-dendral, software útil para interpretar análisis químicos, aparece como publicación en una revista.

1987: Por primera vez una Van Mercedes Benz equipada con dos cámaras y varios ordenadores navega sola en terreno alemán alrededor de 20 km (55 mph). Este proyecto fue dirigido por el ingeniero Ernst Dickmanns.

1997: El ordenador Deep Blue desarrollado por IBM derrota al campeón mundial de ajedrez, Garry Kasparov.

2004: El Pentágono establece el proyecto Darpa Grand Challenge, una carrera para vehículos autónomos que da pie a esta industria.

2012: Los investigadores encuentran un nicho en lo que se conoce como aprendizaje profundo que abre un interés muy fuerte en IA; se muestra que sus ideas pueden hacer el reconocimiento del habla y las imágenes de modo más preciso.

2014: Alpha Go, creado por Google y DeepMind, derrota al campeón del mundo del entonces muy difícil juego del Go.

## 1.6 Aplicaciones comerciales de la IA

La IA ha sido usada en un amplio número de campos como la robótica, la comprensión y traducción de lenguajes, aprendizaje de palabras, etc. Los principales y más destacados campos donde se puede encontrar una notoria evolución de la IA son:

- Ciencias de la computación.
- Finanzas.
- Hospitales y medicina.
- Industria pesada.
- Servicio de atención al cliente.
- Transportación.
- Juegos.

Un estudio llevado a cabo por la empresa Gartner en 2016 muestra que para el presente año 2020, al menos 30 % de las todas las compañías usarán IA y CD en al menos un fragmento de sus procesos de venta. Algunas áreas de oportunidad de la IA y la CD son:


Asistentes personales.


Automóviles autónomos.


Banca y finanzas.


Edificios inteligentes.


Chatbots.


Ciberseguridad.


Comercio electrónico (e-commerce).


Cuidado de la salud.


Diseño de ropa, estilos de zapatos, etcétera.


Electrodomésticos.


Entretenimiento.


Logística y cadenas de suministro.

Manufactura optimizada.

Servicio al cliente en línea.

Sistemas de recomendación.

Telefonía móvil.

Turismo.

Los siguientes son ejemplos específicos de la aplicación de la IA en nuestro mundo:

**Desplazamiento**. De acuerdo con un estudio en el año 2015 por parte del Instituto de Transporte de Texas y la Universidad de Texas A&M, los tiempos de desplazamiento en EUA han aumentado drásticamente, año con año. Esto ocasionó una pérdida estimada de 160 billones de dólares en la productividad en el año de 2014. Los sistemas de predicción que utilizan GPS, los sistemas de transporte compartido y los sistemas de autopiloto en los aviones ayudan a aminorar el problema. Las soluciones planteadas a futuro son las siguientes: 1) Vehículos autónomos, 2) Transporte compartido optimizado y 3) Sistemas de semáforos inteligentes (IOT).

**Correo electrónico**. Pareciera que nuestro buzón de correo es lugar no apto para la IA, sin embargo, lo es. Los filtros de *spam*, los clasificadores inteligentes de correos son solo dos ejemplos de los muchos motores inteligentes que pueden usarse para el manejo optimizado de nuestro correo electrónico. Soluciones futuras: Los respondedores automatizados de correo. Banca y finanzas personales.

Actualmente, muchas de nuestras transacciones bancarias se llevan a cabo a través de motores basados en IA: 1) Depósito de cheques móviles, 2) Prevención de fraudes, y 3) Otorgamiento de créditos. Soluciones futuras: Sistemas consejeros inteligentes para recomendar dónde y cuándo invertir nuestro dinero.

**Redes sociales**. En este caso se obtienen ejemplos de sistemas que: 1) Reciben fotos, las analizan y si encuentran personas hacen recomendaciones de etiquetado, 2) Reciben como entrada fotografías, las analizan reconociendo los objetos ahí presentes y hacen recomendaciones de objetos similares, 3) Son capaces de identificar significado contextual de emojis, pudiendo entonces sugerir en forma automática emojis de respuesta, y 4) Son capaces de seguir movimientos faciales y adicionar efectos animados o máscaras digitales cuando dichos rostros son usados por doquier en el mundo.

Soluciones futuras: Conversación con chatbots inteligentes. Estos chatbots deberán incorporar poderosos: 1) Procesadores de lenguaje natural, 2) Reconocedores de rostros y 3) Analizadores de expresiones faciales, etcétera.

**Compras en línea**. Muchas de las compras que se realizan actualmente son a través de Internet. Incluye buscadores, recomendadores, detectores de fraudes, etcétera. Soluciones futuras: Sistemas de compra completamente personalizados.

**Telefonía móvil.** Prácticamente tres de cada cuatro personas en el mundo usan teléfono móvil. Muchos de esos teléfonos incluyen convertidores de voz a texto, asistentes personales inteligentes, capaces de sostener conversaciones sencillas. Ejemplo: Cortana de Microsoft. Soluciones futuras: Asistentes personalizados inteligentes que reduzcan la brecha entre el ser humano y las casas inteligentes. Deberán ser capaces de responder eficientemente y eficazmente a comandos verbales para tocar música, responder preguntas en lenguaje natural, enviar reportes, noticias, actualizaciones financieras, llamar al taxi, hacer citas y recordatorios, entre otros.

## 1.7 Líneas de investigación

Toda disciplina científica establecida da pie a un conjunto de líneas para la generación y aplicación de conocimiento (LGCA) en problemáticas reales con amplias aplicaciones en el entorno. Las siguientes son tan solo algunas de las líneas de generación y aplicación del conocimiento en IA [Web18-Web20]:

Aprendizaje para máquinas.

Aprendizaje profundo.

Aprendizaje supervisado.

Aprendizaje no supervisado.

Aprendizaje por refuerzo.

Inteligencia Artificial General.

Robótica inteligente.

Planificación.

Minería de datos grandes y complejos.

Servicios de atención al cliente.

Diagnósticos.

Procesamiento de lenguaje natural.

IA en medicina.

IA en educación.

IA en entretenimiento.

IA en gestión de información.

- IA en biología.
- IA en ganadería.
- IA en gestión de tráfico.
- IA en asuntos de seguridad.
- IA en el espacio.
- Diseño y prototipado de placas y dispositivos electrónicos avanzados.
- En agricultura de precisión.

## 1.8 Retos actuales para la IA

Toda técnica, conforme se establece, experimenta diversos retos. La IA no es la excepción. Aunque ya se cuenta con un mejor poder de cómputo (músculo), mejores técnicas para el manejo de grandes cantidades de datos para el aprendizaje de máquinas (aprendizaje profundo), todavía hay mucho camino que recorrer en materia de:

- Datos.
- Inteligencia Artificial multitarea.
- *Hardware*.

**Datos**. La IA necesita miles de veces más datos que los requeridos por el cerebro humano para comprender conceptos y características. La capacidad de las máquinas para ver, entender e interactuar con el mundo está creciendo a un ritmo acelerado, apoyado en el

volumen de datos que les ayuda a aprender y entender aún más rápido. Cada año la cantidad de datos que producimos se duplica. En la próxima década habrá aproximadamente 150 mil millones de sensores conectados a la red, equivalente a más de veinte veces la población de la Tierra. En este sentido, el "Big Data" ha sido y será un gran aliado de la IA para procesar esta cantidad cada vez más grande de información y volverla útil.

**Inteligencia artificial multitarea**. Una vez que una máquina de IA es entrenada, es muy efectiva para tareas como el reconocimiento facial o de voz. Solo son capaces de realizar tareas específicas. Actualmente, no hay máquinas inteligentes capaces de cambiar de una tarea a otra. Este es uno de los retos actuales de la IA. Una propuesta de solución consiste en usar las llamadas "redes neuronales progresivas", que aún se encuentran en fase de desarrollo y prueba [Web21].

**Limitaciones de *hardware***. A pesar de la gran capacidad de procesamiento que han alcanzado los ordenadores actuales, toda la información que se encuentra disponible, así como sus técnicas de procesamiento y análisis desarrolladas a lo largo de los años, la IA como tecnología aún se encuentra limitada por el *hardware*. La infraestructura necesaria para experimentar y diseñar con la IA es todavía escasa y costosa. Se necesitan equipos cada vez más potentes para tener mejores resultados.

Posibles soluciones a esta problemática involucran el desarrollo y puesta en operación de nuevos paradigmas de cómputo, como son el masivamente paralelo (basado en cooperación de múltiples procesadores muy sencillos y altamente interconectados), cuántico (el fundamentado en los llamados efectos cuánticos), DNA (que se sabe ocurre en los seres vivos), óptico (que combina la operación de múltiples lentes acoplados entre ellos), bacteriano (que supone el uso eficiente de arreglos de nanotubos), el atómico, etcétera.

## 1.9 ¿Por qué hoy es tiempo de la IA?

Sin duda a equivocarse, se puede decir que ya es el tiempo de la IA. La siguiente aseveración estaría respaldada, primeramente y como ya se dijo, por el hecho de cada día se cuenta con mejores capacidades de cómputo, se tiene acceso a más y mejores datos y se desarrollan mejores técnicas de procesamiento y análisis. En segundo lugar, porque actualmente la IA se usa en múltiples situaciones.

Se cuenta con mejores capacidades de cómputo.

Se tiene acceso a más y mejores datos.

Cada día se desarrollan mejores técnicas de procesamiento y análisis.

En segundo lugar, porque actualmente la IA se usa en múltiples situaciones. La siguiente lista es una consecuencia:

El chef que es capaz de preparar hamburguesas gourmet cada 30 segundos [Web22].

Sistemas con base en IA para predecir erupciones volcánicas [Web23].

Auto capaz de repartir mercancías [Web24].

El robot repartidor de Amazon [Web25].

Algoritmo capaz de romper texto tipo Captcha [Web26].

Toyota quiere posicionar un robot amigable en cada hogar [Web27].

Los algoritmos que son capaces de aprender y ganar a los campeones o profesionales en juegos complejos como el ajedrez, el Go, el Shogi, el póker, etcétera [Web28].

Los robots capaces de tomar decisiones por "sí mismos" en ambientes agresivos y desconocidos [Web29].

Los robots repartidores de helados en entornos semi-controlados [Web30].

Los robots que son capaces de aprender por "sí mismos" a caminar [Web31] y [Web50].

Los robots que realizan tareas antes atribuibles solo a los humanos en supermercados y restaurantes [Web32].

Las prótesis inteligentes que ayudan a las personas con amputaciones a caminar [Web33].

Los algoritmos que encuentran vulnerabilidades en programas de cómputo y las resuelven [Web34].

*Software* que son capaces de escribir noticias falsas y, a la vez, detectarlas como se indica en [Web35-Web36].

Los que son capaces de hacer funciones de un reportero [Web37].

Los que son capaces de extraer conceptos a partir de dibujos básicos [Web38].

Las máquinas que pueden diagnosticar casos de demencia [Web39].

Los robots que pueden cuidar a niños en hospitales [Web40].

Los asistentes que pueden estacionar un auto y, a la vez, optimizar el lugar de estacionamiento [Web41].

Los que inventan lenguajes no verbales para ganar a las cartas [Web42].

Los sistemas asistenciales que ayudan a reducir las fatalidades relacionadas con los peatones [Web43].

Los robots de Fedex que pronto repartirán pizzas [Web44].

Los basados en Deepmind para predecir la potencia de salida del viento [Web45].

En la conservación de especies [Web46].

Los robots que saben dónde vive una persona y qué darle de cenar [Web47].

Los usados en la India para combatir la diabetes [Web48].

Los que son capaces de escribir libros [Web49-Web50].

Los que son capaces de hacer que un auto navegue en forma autónoma en altas velocidades en carreteras [Web51].

Los que son capaces de crear deportes para que los seres humanos los jueguen [Web52].

Los que pueden ayudar a planificar la distribución de productos [Web53].

Los útiles para planificar unas vacaciones [Web54].

Los útiles para diseñar mejores sistemas de IA [Web55].

En tercer lugar, porque de acuerdo con [Web56], el año 2018 es considerado como el despegue definitivo de las soluciones basadas en la IA. Consultoras de prestigio pronosticaron un mercado de más de 31000 millones de dólares en 2019, 55 % más que en 2014. La empresa Gartner ha detectado que una de cada cuatro empresas ya usa IA o tiene planes de hacerlo a corto plazo; 73 % de los desarrolladores que no usan IA previeron aprender a usarla en 2018 [Web57]. Finalmente, porque como se sabe, en marzo de 2019 la Asociación para Cómputo en Máquinas (ACM) otorgó el prestigiado reconocimiento "Alan Turing" a los profesores Yoshua Bengio, Geoffrey Hinton y Yan LeCun [Web58]. Este hecho constituye sin duda una prueba irrefutable que la IA ocupa ya en nuestra sociedad el papel de ciencia y de tecnología.

# 1.10 Enfoques de máquinas inteligentes

De acuerdo con lo visto, se pudiera decir que un verdadero sistema de IA es uno que es capaz de: 1) Aprender por sí mismo y 2) Puede mejorar de iteraciones pasadas, siendo más inteligente y más consciente, además de mejorar sus capacidades y su conocimiento. De acuerdo con la literatura, para la construcción de máquinas inteligentes se puede utilizar uno de los dos siguientes enfoques:

El enfoque de arriba hacia abajo.

El enfoque de abajo hacia arriba.

En el marco del enfoque de arriba hacia abajo (también llamado formal), el diseño de una máquina inteligente involucra la solución de los siguientes dos problemas:

El del reconocimiento de patrones (RP).

El del sentido común (SC).

Si, por ejemplo, se quiere diseñar y poner en operación un robot que sirva como mesero en un restaurante, éste deberá ser capaz de reconocer patrones, tales como palabras, objetos, personas, etcétera.

El segundo problema es del sentido común, resulta fundamental. Cualquier persona sabe que:

El agua está mojada.

Las madres son más viejas que las hijas.

A los animales no les gusta el dolor.

Uno no regresa después de la muerte.

El tiempo no va hacia atrás.

**¡No hay todavía matemáticas que expresen estos hechos!**

Sabemos que cualquier niño aprende de su realidad. Las máquinas apenas empiezan a experimentar esto. En general hacen lo que se les ha programado de antemano. Ha habido proyectos para poner en operación un sistema para tomar en cuenta todas las leyes del sentido común. Fue el proyecto CYC, que inició en 1984 por Douglas Lenat de la empresa Cycorp.

CYC fue el equivalente al proyecto Manhattan; el primero tuvo como objetivo alcanzar el momento en el cual una máquina, por sí misma, fuera capaz de digerir más información al leer revistas y libros. Se predijo que en diez años CYC sería capaz de alcanzar 50 % de conciencia.

Hoy se sabe que CYC estuvo muy lejos de ello, a pesar del gran número de líneas de código escritas. La última versión de CYC abarca alrededor de 47,000 conceptos y 506,000 hechos. El intentar producir un programa que incluya las leyes del sentido común es aparentemente inalcanzable, ya que hay demasiadas leyes. Es bien sabido que los seres humanos lo hacemos sin esfuerzo [Web59-Web60]. Para los robots es muy difícil saber cuando una puerta está abierta y cuando una ventana es la que lo está. Esto no impide, sin embargo, que haya grandes proyectos en esta dirección, por ejemplo, el vehículo autónomo.

En resumen, el enfoque de arriba hacia abajo presenta muchas limitaciones para la construcción de robots. Tomando esto como base surge el llamado enfoque de abajo hacia arriba. Los seguidores de este último tratan de imitar la manera en que los bebés aprenden. Uno de los promotores más reconocidos del enfoque (abajo-arriba) es el Profesor Rodney Brooks del MIT [Web61].

En lugar de usar programas de cómputo elaborados para calcular las posiciones precisas, los "insectoides" del Dr. Brooks, se valen de prueba y error para coordinar los movimientos de sus patas con mucho menos poder de cómputo. Genchis, por ejemplo, es capaz de navegar sin la ayuda de un sistema de cómputo centralizado. Varios de los diseños del Dr. Brooks son usados por los robots de la NASA para recolectar y analizar materiales en el planeta rojo. Otro proyecto de Brooks (COG) intenta crear un robot con la inteligencia de un niño de seis años. A COG no se le programó ninguna inteligencia. Usa sus ojos para enfocar a un humano que le enseña por medio de ejemplos.

Un ingrediente que muchos piensan debiera de incluir una máquina son las emociones. De hecho, varios científicos piensan que para que un robot sea eficiente requiere experimentar emociones. Ejemplos de algunas de ellas que pudiera experimentar son: El desear algo, los celos, la vergüenza, el remordimiento y la soledad. Es muy probable que los robots avanzados vengan equipados con emociones. Incluso, hay quienes piensan que el sentir es un puente crucial entre la inteligencia y la conciencia. Uno se preguntaría por qué las emociones son importantes. Sabemos que en el caso de nosotros los seres humanos, las emociones nos guían a tomar decisiones. Las máquinas del futuro, muy probablemente necesitarán de emociones para fijarse metas y así tomar decisiones; de otra forma, quedarían paralizadas ante un mar de posibilidades.

## 1.11 Aspectos importantes de la IA

Al principio, cuando el ser humano habitó la tierra tuvo necesidad de hacer uso de tecnología. Para abrir una nuez y no hacerlo con los dientes (para no rompérselos) encontró que una piedra podía usarse para tal efecto al golpear la semilla. De igual forma, al consumir carne, en sus inicios lo hacía a mordidas, como lo hacía el resto de los animales. El ingenio y creatividad llevó al hombre a inventar el cuchillo para cortar la codiciada carne de los animales.

Con el devenir del tiempo se dio cuenta que la piedra y el cuchillo como tecnologías podían también ser usadas para lastimar e incluso matar a su prójimo. Con el paso de los años la convivencia entre personas en comunidades llevó al hombre a imponer reglas. A este conjunto de reglas les llamamos actualmente ética. Como toda tecnología, la Inteligencia Artificial debería ser usada con cautela y con ética.

Entre otras cosas, con esta finalidad, es importante establecer un conjunto de reglas mínimas sobre aspectos importantes de la ética en el uso de la IA.

### 1.11.1 Ética en el uso de la IA

Como toda tecnología, la Inteligencia Artificial debería de ser usada con cautela y con ética. Entre otras cosas, con esta finalidad, es importante establecer un conjunto de reglas sobre la ética en el uso de la IA.

**Reglas de ética en el uso de la IA**

- La IA debería de ser utilizada para construir máquinas para el servicio del ser humano, no para su perjuicio.
- No debería de ser usada para construir máquinas de guerra.
- Tampoco debería de ser usada para influir en las elecciones presidenciales de un país, ni para determinar si una persona debiera ser contratada para un trabajo específico.

Para esto es necesario entender los riesgos asociados a la IA; expertos explican las diferencias entre la IA débil y la fuerte, sus usos e implicaciones [Web62]. En el artículo [Web63], se describe con claridad el porqué de las tecnologías como la IA debieran de regularse.

En otros artículos, como los reportados en [Web64-Web65], se habla de los beneficios, riesgos, de lo bueno y lo malo al construir sistemas basados en IA.

### 1.11.2 Algunas estadísticas

En materia de patentes, la World Intellectual Property Organization (WIPO) estima que en los últimos cinco años se ha producido un "boom" en IA, con más de 50 % de todas las patentes desde 2013. De 2013 a 2017, el crecimiento ha pasado de 18,995 a 55,660 patentes. IBM con 8290 y Microsoft 5930, respectivamente.

Sin embargo, importantes datos estadísticos en el contexto de la Inteligencia Artificial se obtienen en [Web66], tales como:

China ostenta 22 % de patentes a nivel mundial.

A nivel mundial, la proporción de artículos/invenciones ha bajado de 8:1 en 2010 a 3:1 en 2016.

Un aspecto muy importante que hay que destacar es que el aprendizaje para máquinas se ha convertido en la técnica más usada en patentes, mientras que la visión por ordenador es la aplicación más popular.

Las aplicaciones representan un mercado que equivale hoy a 21 mil 400 millones de dólares, pero que llegará a 190 mil millones de dólares en 2025, según la firma Markets and Markets.

Los países más robotizados por cada 10,000 empleados a nivel mundial son:

Corea, Singapur, Japón, Alemania, Suecia, Taiwán, Dinamarca, EUA, Bélgica e Italia.

Entre los países latinoamericanos podemos mencionar:

México ocupa la posición 33, por cada 10,000 empleados se tienen 33 robots.

Argentina que ocupa el lugar número 36 con 16 robots por cada 10,000 empleados.

Brasil, lugar número 28 con 11 robots por cada 10,000 empleados.

En la referencia [Web67] se dan otro conjunto de estadísticas interesantes.

## 1.12 Contenido del libro

El resto de esta obra se divide en seis capítulos, con una secuencia y orden cronológico adecuado para la presentación de conceptos de la Inteligencia Artificial. A pesar de que cada capítulo puede ser estudiado de manera independiente, conviene seguir el orden secuencial preestablecido y sugerido.

En el **segundo capítulo**, en primer lugar, se estudian los fundamentos más representativos del Aprendizaje para Máquinas. Se dará una introducción a este campo de la investigación que hoy se reconoce por ser el principal motor de la Inteligencia Artificial. En segundo lugar, se describe el proceso de cómo el AM opera y se presenta cada uno de los cuatro tipos de aprendizaje que se pueden dar en una máquina, a saber: aprendizaje supervisado, aprendizaje semi-supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. En tercer lugar, se expondrá un conjunto de definiciones del álgebra lineal, se expone el tema central: regresión lineal y su solución a través de técnicas de álgebra lineal. En todos los casos se darán ejemplos para ilustrar la operación de las técnicas expuestas, en particular dos aplicaciones al campo de la robótica.

En el **tercer capítulo**, primeramente, se presenta el problema de la clasificación de patrones a través del método de discriminación lineal. En segundo lugar, se describe el problema de la regresión logística, útil también para la clasificación de patrones a través de la estimación de la probabilidad de sus correspondientes clases. Ambos temas son acompañados por ejemplos ilustrativos junto con código fuente MATLAB para una fácil comprensión al lector. Al final, se presenta un conjunto de problemas propuestos para que el lector afirme los conceptos y procedimientos expuestos.

En el **cuarto capítulo** se presentan los pormenores para el acondicionado de imágenes. Se expone el concepto de ruido en una imagen y se muestran varios ejemplos ilustrativos y bien documentados. Después se habla del proceso de filtrado como herramienta computacional para reducir el efecto del ruido causado en una imagen. Se muestra la operación de varios filtros. Más adelante, se explica cómo una imagen puede ser mejorada en su contraste a través de la modificación de su histograma. Al final del capítulo se ofrece un conjunto de ejercicios para que el lector afiance los conocimientos aprendidos y ponga en práctica sus habilidades como programador.

En el **quinto capítulo** se estudia el tema de segmentación de imágenes. Se describen los principios del umbralado automático de imágenes. Se explica un par de métodos muy populares de Otsu y de mínimo error. También se desarrollan métodos para la reducción de ruido residual después del proceso de umbralado. Enseguida, se estudian diversos métodos para el etiquetado de regiones conectadas en imágenes. Posteriormente, se analizan varias propiedades geométricas y topológicas de los objetos segmentados. Después, se discute un conjunto de técnicas para detectar la presencia de objetos y para su identificación en escenas a partir de imágenes o secuencias de imágenes. Al final, se presenta un conjunto de problemas propuestos al lector para mejorar sus habilidades y reafirmen los conceptos expuestos.

En el **sexto capítulo** se estudia la primera parte sobre las redes neuronales artificiales, que intentan simular la operación de las neuronas biológicas. Se presentan los pormenores de las arquitecturas básicas de redes neuronales más utilizadas. Se habla de la conocida unidad de umbralado lógico (UUL) y la forma que puede ser entrenada a través de la regla del perceptrón para resolver problemas de diversa índole. Enseguida, se dan los detalles sobre el perceptrón como extensión de la UUL. Se expone la neurona lineal ADALINE y cómo ésta puede ser entrenada mediante la llamada regla DELTA. Más adelante, se habla de la neurona sigmoidal. Se describen ejemplos ilustrativos, así como ejercicios para que el lector afirme los conocimientos adquiridos y ponga en práctica sus capacidades y habilidades.

Finalmente, en el **capítulo siete**, se continúa con el estudio del perceptrón como máquina de procesamiento de información, pero ahora acomodado en capas. Se explica su regla de aprendizaje, así como la función de la la capa intermedia, útil para transformar problemas no lineales en problemas lineales. Se explica cómo el término de momento permite acelerar el proceso de aprendizaje. Finalmente, se ofrecen las bases de operación de una red neuronal convolucional. Como en capítulos anteriores, se ofrece un conjunto de ejercicios para que el lector ponga en práctica sus habilidades como programador.

## 1.13 Resumen

En este capítulo, después de una introducción donde se explicaron las principales diferencias entre el cómputo basado en IA y el cómputo tradicional, se dio un estado del arte donde se explicaron brevemente las diferentes etapas por las que ha pasado la llamada automatización. Enseguida, se dio una definición del término Inteligencia Artificial, así como sus diferentes categorías.

Después, se ofreció una breve historia de la IA, así como de los principales hechos que a lo largo de la historia le han dado la forma que ahora tiene. Más adelante, se describieron algunas de sus aplicaciones en diversas áreas y el conjunto de líneas de generación del conocimiento y de su aplicación.

Enseguida, se hizo referencia acerca de los retos que todavía enfrenta la IA y de por qué a pesar de ellos se puede asegurar que ya es su momento. Más adelante, se describieron los dos grandes enfoques para la construcción de sistemas inteligentes y de la ética en su uso. Después se dieron algunas estadísticas relacionadas.

Finalmente, se enlista una serie de conclusiones iniciales: siendo dos predominantes que consisten en que todavía no se descubren los principios que gobierna el proceso de la inteligencia y que conforme la IA se va integrando en nuestras vidas, seguramente será la nueva infraestructura que impulse la siguiente revolución industrial.

De acuerdo con los conceptos de IA presentados en este capítulo, hasta el momento se puede concluir que:

Aunque todavía no se descubren las leyes que rigen el proceso de inteligencia en los seres vivos, ya es posible construir máquinas inteligentes. Esto se hace evidente a través de todos los ejemplos enlistados.

No hay leyes de la física que se opongan a esto. Al igual que en su momento la física indicaba que al hombre le era imposible volar, la imaginación y capacidad creativa de este último le permitió inventar el avión con lo que se rompieron leyes físicas; ahora el hombre se encuentra en el momento de inventar máquinas con capacidades de inteligencia antes no vistas, que le facilitarán la vida.

Se ha "rascado" la superficie a través de múltiples ejemplos de la aplicación de la IA en la vida del día a día. El cómo la IA afectará nuestras vidas a gran escala en el futuro cercano lo expresa Kevin Kelly, quien predice que conforme la IA se integre más profundamente en nuestras vidas, sin duda, será la nueva infraestructura que impulse la siguiente revolución industrial [Web68]. Muchas otras interesantes referencias del tema pueden consultarse en [Web69-Web73].

## 1.14 Problemas propuestos

En esta sección se presenta una serie de ejercicios de reflexión para el lector sobre aspectos científicos y aplicaciones comerciales de IA. Asimismo, se le proporciona una serie de situaciones de ética de la IA. En cada caso haga una búsqueda a través de Internet, conteste a las siguientes interrogantes, elabore un resumen y comente con sus compañeros de clases:

1.14.1 ¿Qué es inteligencia natural y para qué le sirve a la creatura viva?

a) ¿Cuántos tipos de inteligencia natural han sido expuestos en la literatura?

b) ¿Para qué son útiles, según la ciencia?

1.14.2 ¿Qué es la inteligencia artificial y qué se asemejaría y diferenciaría respecto a la inteligencia natural?

1.14.3 ¿En su opinión qué aspectos de la inteligencia artificial natural pueden ser modelados mediante la inteligencia artificial?

1.14.4 ¿Cuáles serían cinco aplicaciones recientes de la IA en al menos cinco áreas del saber o aplicabilidad?

1.14.5 ¿Qué es la ética y cómo puede usarse en el ámbito de la IA en al menos tres casos reales?

1.14.6 Mencione por lo menos tres aspectos éticos del por qué la IA no debe usarse en contra de las personas.

1.14.7 Desde el punto de vista ético, reflexione sobre el uso de la IA en las siguientes situaciones:

a) ¿Por qué la IA debe ser usada para mejorar la calidad de vida en las personas?

b) ¿Debería ser utilizada la IA en aplicaciones militares?

c) ¿Conviene utilizar la IA para evitar el terrorismo?

1.14.8 ¿Existe al menos un intento de ley matemática que trate de modelar el proceso de la inteligencia con posibles aplicaciones?

1.14.9 ¿Cuántas y cuáles son las llamadas leyes de la robótica según Asimov?

a) ¿Hoy en día usted cree que se pueden diseñar máquinas que sigan estas leyes?

b) ¿Cuáles serían las posibles implicaciones de que una máquina no siga las leyes?

c) En el caso hipotético de que un robot no siguiera las leyes de la robótica de Asimov ¿Qué puede proponer para corregirlo?

1.14.10 Conteste y argumente lo siguiente:

a) ¿Qué es el aprendizaje para máquinas?

b) ¿En qué tipos o categorías se divide?

c) ¿Qué aplicaciones realistas pudiera tener cada tipo de aprendizaje para máquinas?

1.14.11 ¿Qué son el tratamiento y análisis digital de imágenes y el reconocimiento de patrones?

a) ¿Cuáles serían cinco de sus aplicaciones más importantes?

1.14.12 ¿Qué es una neurona artificial y en qué se asemeja a una neurona biológica?

1.14.13 ¿Qué es una red neuronal artificial?

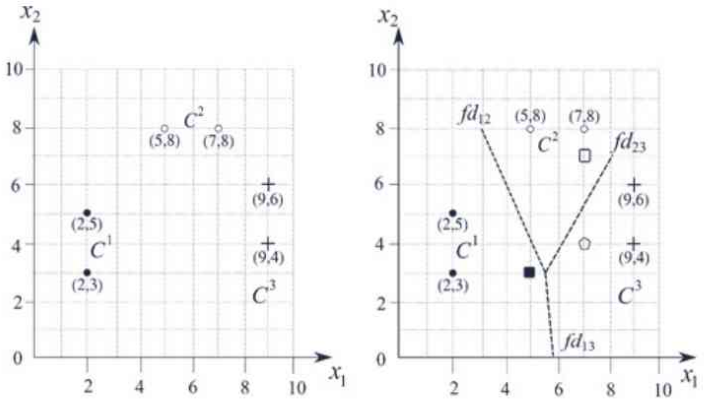
a) ¿Cuántas generaciones de este tipo de máquinas han sido obtenidas en la literatura?

b) ¿En qué se diferencia cada generación de red neuronal artificial?

1.14.14 Haciendo un recuento de los diferentes modelos de redes neuronales artificiales propuestos en la literatura, en su opinión ¿Cuáles son los más representativos?

Capítulo 2

# Aprendizaje de Máquinas: Regresión lineal



# Descripción del capítulo

En este capítulo primeramente se da la definición de Aprendizaje para Máquinas (AM), como principal motor de la Inteligencia Artificial. Enseguida, se describe el proceso de cómo el AM opera. Después, se describe cada uno de los cuatro tipos de aprendizaje que se pueden dar en una máquina, a saber: aprendizaje supervisado, aprendizaje semi-supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Más adelante, se presenta un conjunto de definiciones y conceptos provenientes del álgebra lineal, útiles para la fácil lectura y asimilación del resto del material del capítulo. Después, se expone el tema central: regresión lineal. En todos los casos se dan ejemplos ilustrativos para una fácil asimilación del material expuesto. Al final del capítulo se presenta un conjunto de problemas propuestos para que el lector ponga en acción sus habilidades y reafirme los diferentes conceptos, definiciones y métodos aprendidos.

Los siguientes temas son abordados:


Mecánica del AM.


Conceptos y definiciones básicas.


Tipos de aprendizaje.


Regresión lineal.

# 2.1 Introducción

Nuestro cerebro es un órgano increíble. Es la materia organizada más compleja del universo conocido. Dicta la manera en cómo vemos, oímos, olemos, probamos y tocamos. Es el órgano responsable de nuestro comportamiento, memorias y percepciones. Incluye el más misterioso de los fenómenos, por ejemplo, la conciencia. Es responsable de comportamientos tan simples como comer o caminar; y también de otros comportamientos tan complejos como lo son: pensar, hablar o crear.

El cerebro ocupa aproximadamente 2 % de nuestro cuerpo. Sin este maravilloso órgano seríamos organismos muy primitivos, muy semejantes a las bacterias en su forma de accionar. El cerebro es lo que nos hace inteligentes; a la edad de un año todos éramos capaces de llevar a cabo actividades extraordinarias, como identificar objetos, tomarlos y hacer tareas con ellas, por ejemplo, insertar un cubo dentro de un hueco con la forma de un cuadrado. El cerebro humano es una masa de tejido nervioso aproximadamente 1200 centímetros cúbicos y 1400 gramos. Se compone de alrededor de 86000 millones de neuronas y $10^{14}$ sinapsis (170000 kilómetros de fibra nerviosa).

Por décadas, el ser humano ha soñado con construir máquinas que posean cerebros como los nuestros. Por ejemplo, robots que realicen tareas en el hogar para limpiar o asear, cocinar, cuidar a los niños, por citar algunos. También diseñar coches que sean capaces de navegar en forma autónoma dentro de una ciudad, llevándonos desde el hogar a nuestro trabajo y viceversa. Para lograr esto, algunos de los problemas computacionales más complejos que podríamos imaginar deberían ser resueltos; problemas que nuestros cerebros resuelven de alguna forma en unos cuantos milisegundos.

Para alcanzar esta meta, se requiere:

1) Desarrollar maneras radicalmente diferentes de programar un ordenador, o

2) Usar técnicas nuevas o ya existentes, pero de forma distinta. Se sabe que el cómputo tradicional tiene límites.

Con respecto a esta afirmación, uno podría preguntarse por qué hay ciertos problemas que son difíciles de resolver mediante un ordenador. Ya se vio anteriormente que los programas de cómputo clásico son diseñados para hacer dos cosas muy bien:

1) Para realizar operaciones aritméticas de manera muy rápida, y
2) Para seguir, de manera explícita, un listado de instrucciones.

Si se quisiera, por ejemplo, procesar grandes cantidades de datos financieros, se está de suerte. Un programa de cómputo clásico puede llevar a cabo la tarea sin mucho problema. Sin embargo, hay otros problemas que no se pueden resolver bajo este esquema, como lo es, reconocer caras, conducir coches, entender lenguaje natural, etcétera.

En todos estos casos, es un hecho que no sabemos cómo escribir un programa ya que no sabemos cómo nosotros lo hacemos en nuestro cerebro. Inclusive, si supiéramos cómo hacer esto, el programa resultante podría ser horriblemente complicado.

En los últimos veinte años se ha desatado una revolución: cada vez aparecen más y más programas que aprenden. Programas que son capaces de adaptar su comportamiento de manera automática a los requerimientos de una tarea dada. Programas que reconocen caras, conducen coches, como ya se dijo, algoritmos capaces de detectar con precisión la presencia coronavirus, que recomiendan qué libros comprar o qué películas ver, etcétera.

Antes, un experto programador definía qué debería hacer un ordenador a través de la codificación de un algoritmo en un lenguaje de programación. Hoy para muchas tareas ya no se escriben programas, en lugar de esto lo que se hace es recolectar datos. Estos datos deben contener ejemplos de qué es lo que un ordenador debe hacer, y un algoritmo de aprendizaje puede modificar (en forma automática) a un programa para ser entrenado y dicho programa coteje los datos con los requerimientos específicos.

Para muchas aplicaciones, desde la visión al habla y de la traducción a la robótica, ya no somos capaces de derivar buenos algoritmos a pesar de décadas de investigación desde los años cincuenta del siglo pasado. Sin embargo, para estas tareas es muy fácil recolectar datos. La idea fundamental es aprender de manera automática algoritmos a partir de datos, reemplazando al programador por esquemas que aprenden.

En la literatura es aceptado que cuando esto sucede, se denomina Aprendizaje para M áquinas. En estos momentos es conveniente tener a la mano la siguiente definición.

**Definición 2.1: Aprendizaje para Máquinas**

El Aprendizaje para Máquinas (AM) es un campo de las ciencias de la computación relacionada con la construcción de algoritmos que para ser útiles, se basan en un conjunto de datos donde se involucra algún fenómeno. Estos datos pueden provenir de la naturaleza, confeccionados por un ser humano o generados a través de un algoritmo.

El AM puede también definirse como el proceso de resolver un problema práctico, como:

1) La recolección de datos.
2) La construcción, mediante un algoritmo de un modelo estadístico tomando como base dichos datos. Se asume que este modelo será usado de alguna manera para resolver el problema práctico.

## 2.2 Mecánica del AM

PARA poder explicar este asunto, formulemos la siguiente pregunta: ¿Cómo los seres humanos aprendemos los conceptos naturales? Es un hecho que las cosas se aprenden a través de ejemplos y no a partir de ecuaciones. Desde niños nuestros padres nos enseñaron cómo reconocer un gato, mostrándonos un gato vivo o una fotografía de ellos.

Nuestro cerebro nos proporciona un modelo que describe el mundo que percibimos de manera continua. A lo largo de nuestras vidas, este modelo se hace más preciso a través de ejemplos observados. Todo esto acontece:

1) Sin darnos cuenta, y
2) De manera inconsciente.

Si nuestro modelo viene dado por la siguiente ecuación:

$$h(\boldsymbol{x}, \boldsymbol{\theta}) \tag{2.1}$$

donde $\boldsymbol{x}$ representa cada una de las observaciones relativas al fenómeno bajo estudio representada en forma vectorial, y $\boldsymbol{\theta}$ es el vector de parámetros utilizado por $h$.

Nuestro algoritmo de entrenamiento tratará de mejorar, poco a poco, el vector $\boldsymbol{\theta}$ a través de más y más ejemplos, buscando producir un mejor modelo. En la figura 2.1 se ilustra este proceso, donde como se puede ver, a partir de un conjunto de observaciones el modelo $h(\boldsymbol{x}, \boldsymbol{\theta})$ al principio produce errores, los cuales son usados por el algoritmo de entrenamiento para ajustar (mejorar) el vector de parámetros $\boldsymbol{\theta}$; con ello se busca reducir la tasa de errores al máximo. Vale mencionar que este paradigma general puede ser utilizado para cualquier tipo de modelo, no importando el problema en cuestión.

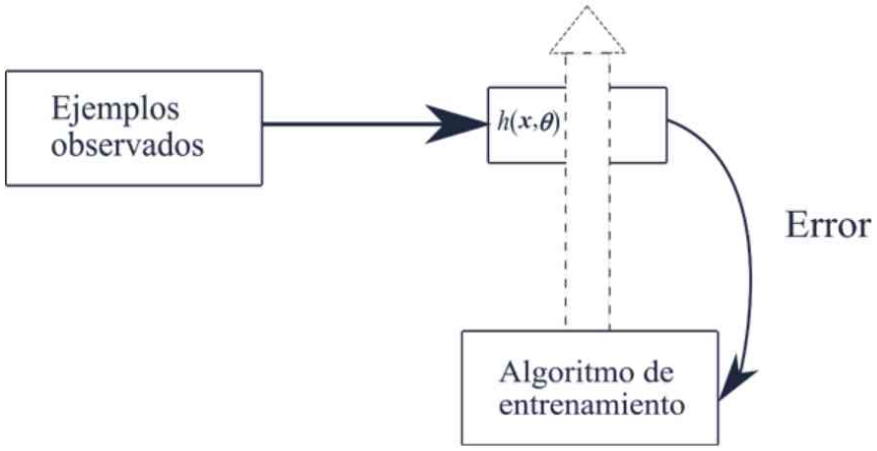



**Figura 2.1**: Ilustración del proceso de refinamiento del modelo $h(\boldsymbol{x}, \boldsymbol{\theta})$.

Para una aplicación dada, como el reconocimiento de caras, los datos pueden ser muchos. Algo interesante es que todos los datos, por ejemplo, las fotografías de las personas, pueden ser explicados en términos de un modelo relativamente simple compuesto de número oculto de factores y sus interacciones.

La meta del AM es convertir los datos en conocimiento. Hoy por hoy, la gente que trabaja en AM encuentra formas originales para usar los datos y convertirlos en productos o servicios útiles, un teléfono, un auto, etcétera. Antes la gente pensaba que para que la Inteligencia Artificial fuera una realidad, se requerían de nuevos paradigmas, maneras de pensar, modelos computacionales u otros conjuntos de algoritmos.

Tomando en cuenta los éxitos muy recientes del AM, actualmente, se puede establecer como un hecho que lo que se requiere es de:

1) Muchos datos, y

2) Suficiente poder de cómputo.

Lo anterior es para echar a andar métodos de aprendizaje con estos datos y derivar los algoritmos requeridos según sea la aplicación, no importando su complejidad. Pareciera que conforme la tecnología se desarrolla se tiene acceso a ordenadores más rápidos y se obtienen más datos; por lo que, los algoritmos de entrenamiento irán dando pie a niveles de inteligencia cada vez más sofisticados, que claramente encontrarán más usos.

Para más detalles y su correspondiente profundización sobre estos temas, se recomienda al autor dirigirse a las siguientes referencias [3] y [4].

## 2.3 Conceptos y definiciones básicas

En esta sección se recordará un conjunto de conceptos y definiciones básicas. Estos conceptos provenientes del álgebra lineal y del cálculo son necesarios para entender las siguientes secciones. En particular, se revisarán los conceptos de escalar, vector y matriz.

**Definición 2.2: Escalar**

En álgebra lineal, un escalar es un número real, constante o complejo que sirve para describir un fenómeno físico o de otro tipo con magnitud, pero sin la características vectoriales de dirección y sentido.

**Notación**

En el contexto de esta obra, el conjunto de los **números reales** se denota con el símbolo $\mathbb{R}$ y sus elementos se conocen como escalares. En esta obra, los números reales o escalares se representan con letras minúsculas en estilo *itálicas* de los alfabetos castellano o griego.

Por ejemplo:

$a, b, c, x, y, z, \alpha, \omega, \lambda \in \mathbb{R}$.

Cuando sea necesario, los elementos escalares de un vector podrán ser representados como $x_i$.

**Notación**

El conjunto de números reales positivos y negativos se representan por $\mathbb{R}_+$ y $\mathbb{R}_-$, respectivamente:

$$\begin{aligned} \mathbb{R}_+ &= \{\alpha \in \mathbb{R} : \alpha \in (0, \infty)\}, \\ \mathbb{R}_- &= \{\alpha \in \mathbb{R} : \alpha \in (-\infty, 0)\}. \end{aligned}$$

Por otro lado, el conjunto de los números enteros $Z$ son escalares de la forma: $[\cdots, -4, -3, -2, -1, 0, 1, 2, 3, 4 \cdots]$; el conjunto de los números enteros $Z$ resulta un subconjunto del conjunto de los números reales, es decir: $Z \subseteq \mathbb{R}$.

Los números naturales $N$ (enteros positivos) son $1, 2, 3, 4 \cdots$; representan un subconjunto de los números enteros: $N \subseteq Z \subseteq \mathbb{R}_+$.

Es muy importante ofrecer una correcta interpretación cuando estemos trabajando con intervalos $a \in [\alpha, \beta]$ y $a \in (\alpha, \beta)$; por ejemplo:

- El intervalo $a \in [\alpha, \beta]$ significa $\alpha \leq a \leq \beta$.
- Mientras que para $a \in (\alpha, \beta)$ significa $\alpha < a < \beta$.

**Definición 2.3: Vector**

Un vector de dimensión $n$ es una tupla o arreglo de $n$ números reales o escalares, llamados componentes del vector. En el contexto de esta obra un vector se representa como una letra minúscula de los alfabetos castellano o griego, pero en negritas (es decir, tipo **bold**).

Ejemplos de dos vectores son los siguientes: $\boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \\ \beta_{n+1} \end{bmatrix}.$$

El vector $\boldsymbol{y}$, como se puede ver tiene $n$ componentes, donde $n \in N$; mientras que el segundo $\boldsymbol{\beta}$, tiene $n + 1$ componentes.

A veces, un vector requiere ser representado en forma transpuesta $\boldsymbol{y}^T \in \mathbb{R}^{1\times n}$, se utiliza el operador transpuesta $T$, como sigue: $\boldsymbol{y} = [\, y_1 \quad y_2 \quad \cdots \quad y_n \,]^T$.

**Notación**

Los **vectores** serán representados en letras minúsculas con estilo **bold** o "**negritas**" y tipo ***itálica*** de los alfabetos castellano o griego. Usaremos tres tipos de nomenclatura para representar a un vector, dependiendo del contexto que estemos tratando.

Los siguientes vectores utilizan la notación más compacta: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{\tau}, \boldsymbol{\gamma}, \boldsymbol{\psi} \in \mathbb{R}^n$, donde $n \in N$ representa la dimensión del vector (es decir, el número de componentes escalares). El número $n$ solo puede ser natural o entero positivo.

Vectores tipo columna son arreglos verticales de $n$ renglones y una columna: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{\tau}, \boldsymbol{\gamma}, \boldsymbol{\psi} \in \mathbb{R}^{n\times 1}$.

Vectores con elementos o componentes escalares se encuentran dentro de corchetes o paréntesis,

$$\boldsymbol{x} \;=\; \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = [\, x_1, \quad x_2 \quad \cdots \quad x_n \,]^T \text{ donde } x_i \in \mathbb{R},\ i = 1, 2, \cdots, n.$$

Para los propósitos de la presente obra, los componentes o elementos $x_i \in \mathbb{R}$ de un vector $\boldsymbol{x} \in \mathbb{R}^n$ serán considerados exclusivamente números reales.

**Definición 2.4: Producto interno**

El producto interno entre dos vectores (también es conocido como producto punto o producto escalar) $\boldsymbol{a} \in \mathbb{R}^n$ y $\boldsymbol{b} \in \mathbb{R}^n$, y por tanto, con $n$ componentes como a continuación se indica: $\boldsymbol{a} = [\, a_1 \quad a_2 \quad \cdots \quad a_n \,]^T$ y $\boldsymbol{b} = [\, b_1 \quad b_2 \quad \cdots \quad b_n \,]^T$, es denotado por $\boldsymbol{a} \cdot \boldsymbol{b}$ o $\boldsymbol{a}^T\boldsymbol{b}$, cuyo resultado es un escalar: $\boldsymbol{a} \cdot \boldsymbol{b} \in \mathbb{R}$ o $\boldsymbol{a}^T\boldsymbol{b} \in \mathbb{R}$, viene dado como sigue:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^T\boldsymbol{b} \;=\; [\, a_1 \quad a_2 \quad \cdots \quad a_n \,]^T \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = a_1b_1 + a_1b_2 + \cdots + a_nb_n = \sum_{i=1}^{n} a_ib_i \in \mathbb{R}. \quad (2.2)$$

**● Ejemplo 2.1**

Obtener el producto punto de los siguientes vectores: $\boldsymbol{a} = [3 \quad 1 \quad -4 \quad 2]^T$ y $\boldsymbol{b} = [2 \quad -5 \quad 3 \quad -6]^T$.

**Solución**

El producto punto entre $\boldsymbol{a}$ y $\boldsymbol{b}$ es representado por $\boldsymbol{a} \cdot \boldsymbol{b}$ o también como $\boldsymbol{a}^T\boldsymbol{b}$; produce como resultado el siguiente escalar:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^T\boldsymbol{b} = [3 \quad 1 \quad -4 \quad 2]^T \begin{bmatrix} 2 \\ -5 \\ 3 \\ -6 \end{bmatrix} = 3(2) + 1\ (-5) + (-4)3 + 2(-6) = -23.$$

■■■

**Definición 2.5: Matriz**

Una matriz es un arreglo bidimensional de datos, conformados en $n$ renglones y $p$ columnas. Sus entradas o elementos escalares son números reales.

En el contexto de este libro, una matriz será representada por medio de letras mayúsculas de los alfabetos castellano o griego $A, B, C \cdots, Z,\ A, B, \Gamma, \cdots \Omega$ y sus elementos son escalares representados con la misma letra, pero en minúscula $a_{ij}, b_{ij}, c_{ij} \cdots, z_{ij}, \alpha_{ij}, \beta_{ij}, \gamma_{ij} \cdots, \omega_{ij}$. El doble subíndice indica la posición del elemento escalar dentro del arreglo rectangular a la que pertenecen, es decir: $i$ representa el renglón $i$-ésimo, para $i = 1, 2, \cdots, n$; mientras que $j$ indica la $j$-ésima columna, para $j = 1, 2, \cdots, p$.

Por ejemplo:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}, \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{np} \end{bmatrix}.$$

En el primer caso, note que la matriz $X$ es cuadrada de dimensión $n \times n$, mientras que $A$ es una matriz rectangular, cuya dimensión corresponde a $n \times p$.

- Dado un vector $\boldsymbol{x} \in \mathbb{R}^n$, también admite la representación de una matrix $\boldsymbol{x} \in \mathbb{R}^{n \times 1}$, es decir, tiene $n$ renglones y una columna.

- Con respecto a la notación entre vectores y matrices, debe quedar claro que $\boldsymbol{x} \in \mathbb{R}^n$ o $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ son representaciones del mismo vector por estar en minúscula y en bold, mientras que $X \in \mathbb{R}^{n\times n}$ representa una matriz (en este caso cuadrada), ya que se encuentra en mayúscula. Por lo tanto, son entidades matemáticas diferentes; es decir, no se debe confundir la notación de un vector $\boldsymbol{x}$ con una matriz $X$.

**Definición 2.6: Matriz diagonal**

Sea $X$ una matriz cuadrada, $X \in \mathbb{R}^{n\times n}$. Una matriz diagonal es aquella donde todos sus elementos sobre la diagonal principal son diferentes de cero: $x_{ij} \neq 0$ si $i = j$, y $x_{ij} = 0$ si $i \neq j$:

$$X = \begin{bmatrix} x_{11} & 0 & 0 & \cdots & 0 \\ 0 & x_{22} & 0 & \cdots & 0 \\ 0 & 0 & x_{33} & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & x_{nn} \end{bmatrix}. \tag{2.3}$$

**Definición 2.7: Matriz transpuesta**

Sea $X$ una matriz con $n$ filas o renglones y $p$ columnas. En este caso el elemento $x_{ij}$ de la matriz original $X$ se convertirá en el elemento $x_{ji}$ de la matriz transpuesta; la matriz transpuesta es denotada como $X^T$:

$$x_{ij} \in X \Rightarrow x_{ji} \in X^T,\ 1 \leq i \leq n,\ 1 \leq j \leq p. \tag{2.4}$$

Ejemplos de matrices transpuestas son los siguientes:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} \Rightarrow X^T = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{nn} \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & -3 & 5 \\ -4 & 1 & -6 \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} 2 & -4 \\ -3 & 1 \\ 5 & -6 \end{bmatrix}.$$

**Definición 2.8: Suma de matrices**

Sean $A \in \mathbb{R}^{n\times p}$ y $B \in \mathbb{R}^{n\times p}$ dos matrices con las mismas dimensiones, la suma de $A$ y $B$ denotada como $A+B$, viene dada como:

$$A+B = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{np} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1p}+b_{1p} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2p}+b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \cdots & a_{np}+b_{np} \end{bmatrix}. \quad (2.5)$$

**• Ejemplo 2.2**

Considere las siguientes 2 matrices de $2\times 2$ elementos: $A = \begin{bmatrix} 3 & 5 \\ 1 & -2 \end{bmatrix}$ y $B = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}$, realice un programa en **MATLAB** para obtener la suma $A+B$ entre las dos matrices.

**Solución**

De acuerdo con la regla para sumar matrices (2.5), $A+B$ se desarrolla de la siguiente forma:

$$A+B = \begin{bmatrix} 3 & 5 \\ 1 & -2 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 3+1 & 5+2 \\ 1+4 & -2+5 \end{bmatrix} = \begin{bmatrix} 4 & 7 \\ 5 & 3 \end{bmatrix}.$$

El código fuente en **MATLAB** para realizar la suma de dos matrices se encuentra descrito en el cuadro de código 2.1. En las líneas de programación 6 y 7 del programa cap2_SumadeMatrices.m se incializan las matrices $A$ y $B$, respectivamente.

La forma usual de realizar la suma de matrices $C1 = A + B$ es como la que presenta la línea 8. Sin embargo, una manera interesante e ilustrativa para llevar a cabo dicha suma de matrices es por medio de un algoritmo recursivo, es decir, implementado por medio de la ecuación (2.5). Con esta finalidad, primero se obtiene la dimensión de la matriz a través de la función size(·), para obtener el número de renglones y columnas, tal y como se muestra en la línea 10.

El algoritmo recursivo para sumar matrices de dimensión $n$ se encuentra comprendido entre las líneas 12 y 16 usando instrucciones for en forma anidada, el for externo se utiliza para los renglones: $i = 1, 2 \cdots$ renglones, mientras que el for interno para el procesamiento de las columnas: $j = 1, 2 \cdots$ columnas. El resultado de este procedimiento se exhibe en la pantalla de comandos de **MATLAB** por medio de la línea de programación 17, la matriz $C2$ contiene registrado el resultado de este procedimiento recursivo; el lector puede verificar que dicho resultado es idéntico al que se presenta en la línea 8.

**Código MATLAB 2.1 cap2_SumadeMatrices.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 2. Aprendizaje de Máquinas: Regresión lineal
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap2_SumadeMatrices.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
format short % formato corto con 4 dígitos después del punto decimal.
 % se declaran e inicializan las matrices A, B ∈ IR^{2×2}.
A=[3, 5; 1, -2]; % A = [3 5; 1 -2].
B=[1, 2;  4 ,5]; % B = [1 2; 4 5].
C1=A+B % exhibe resultado estándar de la suma de matrices A + B.
 % obtiene la dimensión de la matriz A.
[renglones, columnas]=size(A);
 % algoritmo recursivo para sumar matrices.
for i=1:renglones
    for j=1:columnas
        C2(i,j)=A(i,j)+B(i,j); % regla para sumar matrices: ecuación (2.5).
    end
end
  C2 % exhibe resultado de la suma de matrices del algoritmo recursivo (2.5).
```

El código fuente del programa cap2_SumadeMatrices.m se encuentra disponible en el sitio Web de la presente obra.

**Definición 2.9: Producto de matrices**

Sean $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$ dos matrices tal que el número $n$ de columnas de la matriz $A$ es igual al número $n$ de filas de la matriz $B$; el producto entre $A$ y $B$, denotado como $AB \in \mathbb{R}^{m\times p}$, viene dado como:

$$AB = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1n}b_{n1} & \cdots & a_{11}b_{1p} + a_{12}b_{2p} + \cdots + a_{1n}b_{np} \\ a_{21}b_{11} + a_{22}b_{21} + \cdots + a_{2n}b_{n1} & \cdots & a_{21}b_{1p} + a_{22}b_{2p} + \cdots + a_{2n}b_{np} \\ \vdots & \cdots & \vdots \\ a_{m1}b_{11} + a_{m2}b_{21} + \cdots + a_{mn}b_{n1} & \cdots & a_{m1}b_{1p} + a_{m2}b_{2p} + \cdots + a_{mn}b_{np} \end{bmatrix}. \quad (2.6)$$

Un caso particular de matriz diagonal (ver definición (2.6), página 47) es la matriz identidad, donde todos sus elementos sobre la diagonal principal son unitarios y se representa por $I \in \mathbb{R}^{n\times n}$:

$$I = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \quad (2.7)$$

El producto de una matriz $A \in \mathbb{R}^{n\times n}$ por la matriz identidad $I \in \mathbb{R}^{n\times n}$, satisface: $IA = AI \in \mathbb{R}^{n\times n}$.

Sean dos matrices $A \in \mathbb{R}^{n\times n}$ y $B \in \mathbb{R}^{n\times n}$, en general el producto de $A$ y $B$ no es conmutativo, es decir $AB \neq BA$. Sin embargo, solo en el caso de que $A, B \in \mathbb{R}^{n\times n}$ sean ambas matrices diagonales, entonces se satisface la propiedad conmutativa del producto de matrices; es decir: $AB = BA \in \mathbb{R}^{n\times n}$.

**• Ejemplo 2.3**

Suponga las siguientes 2 matrices de $2 \times 2$ elementos:

$$A = \begin{bmatrix} 3 & 5 \\ 1 & -2 \end{bmatrix} \text{ y } B = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}.$$

Desarrollar un programa en **MATLAB** que realice la multiplicación $AB$.

**Solución**

Utilizando la regla de multiplicación entre dos matrices (2.6), el producto $AB$ realizado a mano entre $A \in \mathbb{R}^{2\times 2}$ y $B \in \mathbb{R}^{2\times 2}$ produce lo siguiente:

$$C = AB = \begin{bmatrix} 3 & 5 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 3(1)+5(4) & 3(2)+5(5) \\ 1(1)+(-2)4 & 1(2)+(-2)5 \end{bmatrix} = \begin{bmatrix} 23 & 31 \\ -7 & -8 \end{bmatrix}.$$

El código fuente en **MATLAB** para realizar el producto de dos matrices $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$ se encuentra contenido en el programa cap2_MultiplicaAB.m (ver cuadro de código 2.2). En las líneas de programación 5 y 6 se definen e inicializan las matrices $A$ y $B$, respectivamente; en la línea 12 se presenta la multiplicación estándar $AB$ en **MATLAB**.

Una condición necesaria que se requiere para realizar la multiplicación de las matrices $AB$ es que el número de columnas $n$ de la matriz $A$ sea el mismo número $n$ de renglones en la matriz $B$, tal y como lo especifica la definición (2.9), de tal forma que la matriz resultante es $AB \in \mathbb{R}^{m\times p}$. Por este motivo en las líneas de programación 7 y 8 se obtienen las dimensiones de las matrices $A$ y $B$, respectivamente; para verificar que sus dimensiones sean compatibles por ejemplo, en la línea 9, se comprueba dicha compatibilidad para llevar a cabo la multiplicación, en otro caso se indica al usuario que hay un problema de dimensiones en las matrices $A$ y $B$. En este ejemplo estamos tratando con matrices cuadradas, donde el número de renglones es igual al número de columnas, es decir, $m = n = p$. Sin embargo, en general pueden ser matrices rectangulares.

Si las dimensiones de las matrices $A$ y $B$ son adecuadas, es decir, el número de columnas de la matriz $A$ es el mismo número de renglones que la matriz $B$, entonces en la línea 11 se realiza el algoritmo iterativo de la multiplicación $AB$, cuyo resultado está en la línea 20.

Es importante hacer notar al lector que el producto de matrices no es conmutativo, en general $AB \neq BA$. Se puede verificar que el resultado de $AB$ es diferente a $BA$ (con excepción para matrices diagonales). Si para matrices rectangulares $A$ y $B$ hay compatibilidad de dimensiones para realizar $AB$, entonces $BA$ no necesariamente se puede realizar.

**Código MATLAB 2.2 cap2_MultiplicaAB.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 2. Aprendizaje de Máquinas: Regresión lineal.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap2_MultiplicaAB.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
format short % formato corto con 4 dígitos después del punto decimal.
A=[3, 5; 1, -2]; % A = [3 5; 1 -1].
B=[1, 2;  4 ,5]; % B = [1 2; 4 5].
[m, n1]=size(A); % % obtiene la dimensión de la matriz A.
[n2, p]=size(B); % % obtiene la dimensión de la matriz B.
if  (n1~=n2) % determinan compatibilidad de dimensiones de las matrices.
    disp( 'La multiplicación de AB no puede realizarse: dimensiones inadecuadas.' )
    else
        C1=A*B % resultado de la multiplicación AB estándar en MATLAB.
        for i=1:m % número de renglones.
            for j=1:n1 % número de columnas.
                C2(i,j)=A(i,:)*B(:,j); % regla de multiplicación de matrices (2.6).
            end
        end
    end
end
 C2 % resultado de la multiplicación AB por el método iterativo (2.6).
```

**Definición 2.10: Determinante de una matriz**

Sea $A$ una matriz cuadrada de $n \times n$ elementos, es decir: $A \in \mathbb{R}^{n \times n}$. El determinante de $A$, denotado como $|A|$, viene dado como:

$$|A| = \sum_{\sigma \in p_n} \text{sgn}(\sigma) \, \Pi_{i=1}^{n} a_{i,\sigma_i} \tag{2.8}$$

donde, como se sabe, la sumatoria es calculada sobre todas las permutaciones $\sigma$ del conjunto $\{1, 2, \cdots, n\}$. La posición del elemento $i$-ésimo después de la permutación $\sigma$ se denota como $\sigma_i$. El conjunto de todas las permutaciones es $p_n$. Para cada $\sigma$, sgn($\sigma$) es el signo de $\sigma$. La función sgn($\sigma$) retorna los siguientes valores: +1 si $\sigma > 0$; 0 si $\sigma = 0$; y -1 si $\sigma < 0$. Entonces, sgn($\sigma$) es +1 si la permutación es par (resulta positivo) y si es impar (resultado negativo) adquiere el valor de sgn($\sigma$)=-1.

Para cualquiera de los $n!$ sumandos (donde $n! = (1)(2)\cdots(n-1)(n)$. El término $n!$ se lee $n$ factorial), la notación $\Pi_{i=1}^{n} a_{i,\sigma_i}$ denota el producto de las entradas en la posición $(i, \sigma_i)$, donde va desde 1 hasta $n$: $a_{1,\sigma_1} \; a_{2,\sigma_2} \cdots a_{n,\sigma_n}$. Por ejemplo, la forma práctica de calcular el determinante de matrices cuadradas de dimensión $2 \times 2$ y $3 \times 3$ es la siguiente:

En el caso de una matriz de $2 \times 2$, $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ el determinante es:

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}. \tag{2.9}$$

Para el caso de una matriz $A$ de dimensión $3 \times 3$, el determinante de $A$ puede calcularse utilizando los cofactores de cualquier fila o cualquier columna, de la siguiente manera:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{31} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{2.10a}$$

$$|A| = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \tag{2.10b}$$

$$|A| = -a_{21} \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} + a_{22} \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} - a_{23} \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \tag{2.10c}$$

$$|A| = a_{31} \begin{vmatrix} a_{22} & a_{23} \\ a_{22} & a_{23} \end{vmatrix} - a_{32} \begin{vmatrix} a_{21} & a_{23} \\ a_{21} & a_{23} \end{vmatrix} + a_{33} \begin{vmatrix} a_{11} & a_{22} \\ a_{21} & a_{22} \end{vmatrix}. \tag{2.10d}$$

**• Ejemplo 2.4**

Suponga las siguientes matrices de $A \in \mathbb{R}^{2\times 2}$ y $B \in \mathbb{R}^{3\times 3}$:

$$A = \begin{bmatrix} 3 & 5 \\ 1 & -2 \end{bmatrix}, \; B = \begin{bmatrix} -17 & -11 & 1 \\ 4 & 7 & -2 \\ 6 & 3 & -3 \end{bmatrix}$$

obtener sus respectivos determinantes.

**Solución**

El determinante de la matriz $A \in \mathbb{R}^{2\times 2}$ se obtiene utilizando la expresión (2.9):

$$|A| = \begin{vmatrix} 3 & 5 \\ 1 & -2 \end{vmatrix} = 3(-2) - 1(5) = -6 - 5 = -11.$$

Para el caso de la matriz de $B \in \mathbb{R}^{3\times 3}$ se utiliza la expresión (2.10a) que viene dada como sigue:

$$\begin{aligned} |B| &= -17\begin{vmatrix} 7 & -2 \\ 3 & -3 \end{vmatrix} + 11\begin{vmatrix} 4 & -2 \\ 6 & -3 \end{vmatrix} + 1\begin{vmatrix} 4 & 7 \\ 6 & 3 \end{vmatrix} \\ &= -17(-21+6) + 11(-12+12) + 1(12-42) = 255 + 0 - 30 = 225. \end{aligned}$$

En **MATLAB** la función que calcula el determinante de una matriz $A \in \mathbb{R}^{n\times n}$ es: det($A$). Por ejemplo, en la ventana de comandos se puede verificar el siguiente código:

```
fx >> A=[3, 5; 1. -2]; ↩
fx >> deterA=det(A) ↩
fx >> deterA=-11
```

```
fx >> B=[-17, -11, 1; 4, 7, -2; 6, 3, -3]; ↩
fx >> deterB=det(B) ↩
fx >> deterB=225
```

**Definición 2.11: Matriz adjunta**

Sea $A$ una matriz cuadrada $\mathbb{R}^{n\times n}$, la matriz adjunta de $A$ denotada como Adj($A$) es la transpuesta de la matriz de cofactores de $A$:

$$\text{Adj}(A) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}^T . \tag{2.11}$$

**• Ejemplo 2.5**

Considere la siguiente matriz de $3 \times 3$ elementos:

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 2 \\ 2 & 1 & 5 \end{bmatrix}$$

obtener la matriz adjunta $\text{Adj}(A)$.

**Solución**

La matriz adjunta $\text{Adj}(A)$ viene dada como sigue. Los cofactores de los nueve elementos de la matriz $A$ son:

$$a_{11} = \begin{vmatrix} -3 & 2 \\ 1 & 5 \end{vmatrix} = -17 \quad a_{12} = \begin{vmatrix} 0 & 2 \\ 2 & 5 \end{vmatrix} = 4 \quad a_{13} = \begin{vmatrix} 0 & -3 \\ 2 & 1 \end{vmatrix} = 6$$

$$a_{21} = \begin{vmatrix} 2 & -1 \\ 1 & 5 \end{vmatrix} = -11 \quad a_{22} = \begin{vmatrix} 1 & -1 \\ 2 & 5 \end{vmatrix} = 7 \quad a_{23} = \begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix} = 3$$

$$a_{31} = \begin{vmatrix} 2 & -1 \\ -3 & 2 \end{vmatrix} = 1 \quad a_{32} = \begin{vmatrix} 1 & -1 \\ 0 & 2 \end{vmatrix} = -2 \quad a_{33} = \begin{vmatrix} 1 & 2 \\ 0 & -3 \end{vmatrix} = -3.$$

Luego entonces, la traspuesta de la matriz de estos cofactores da como resultado la matriz adjunta de $A$:

$$\text{Adj}(A) = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^T = \begin{bmatrix} -17 & -11 & 1 \\ 4 & 7 & -2 \\ 6 & 3 & -3 \end{bmatrix}.$$

**Definición 2.12: Matriz inversa**

Sea $A$ una matriz cuadrada de $A \in \mathbb{R}^{n \times n}$, la matriz inversa de $A$ es representada por $A^{-1} \in \mathbb{R}^{n \times n}$ y viene dada como:

$$A^{-1} = \frac{\text{Adj}(A)}{|A|}. \tag{2.12}$$

Observe que la inversa de una matriz $A$ existe, si su determinante $|A|$ es diferente de cero: $|A| \neq 0$.

**• Ejemplo 2.6**

Considere la siguiente matriz $A \in \mathbb{R}^{3\times}$:

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 2 \\ 2 & 1 & 5 \end{bmatrix}$$

obtener la matriz inversa $A^{-1}$.

**Solución**

La matriz inversa de $A \in \mathbb{R}^{3\times 3}$, en términos de la ecuación (2.12) viene dada como sigue:

- La matriz adjunta $\text{Adj}(A)$ es la que se obtuvo en el ejemplo 2.5.
- El determinante de la matriz $A$ se obtiene de la siguiente forma:

$$\begin{aligned} \det(A) &= (1)\begin{vmatrix} -3 & 2 \\ 1 & 5 \end{vmatrix} - (2)\begin{vmatrix} 0 & 2 \\ 2 & 5 \end{vmatrix} + (-1)\begin{vmatrix} 0 & -3 \\ 2 & 1 \end{vmatrix} \\ &= -15 - 2 + 8 - 6 = -15. \end{aligned}$$

- La matriz inversa de $A$ está dada por:

$$\begin{aligned} A^{-1} &= \frac{\text{Adj}(A)}{|A|} = -\frac{1}{15}\begin{bmatrix} -17 & -11 & 1 \\ 4 & 7 & -2 \\ 6 & 3 & -3 \end{bmatrix} = \begin{bmatrix} \frac{17}{15} & \frac{11}{15} & -\frac{1}{15} \\ -\frac{4}{15} & -\frac{7}{15} & \frac{2}{15} \\ -\frac{6}{15} & -\frac{3}{15} & \frac{3}{15} \end{bmatrix} \\ &= \begin{bmatrix} 1.1333 & 0.7333 & -0.0667 \\ -0.2667 & -0.4667 & 0.1333 \\ -0.4000 & -0.2000 & 0.2000 \end{bmatrix}. \end{aligned}$$

- El código en **MATLAB** para verificar los cálculos numéricos es el siguiente. En la ventana de comandos:

  $fx >>$ A=[1,2,-1; 0,-3,2; 2,1,5,]; ↩

  $fx >>$ det(A) ↩

  $fx >>$ ans=
  -15

  $fx >>$ inv(A) ↩

  ans=

$$\begin{matrix} 1.1333 & 0.7333 & -0.0667 \\ -0.2667 & -0.4667 & 0.1333 \\ -0.4000 & -0.2000 & 0.2000 \end{matrix}$$

# 2.4 Tipos de aprendizaje

El AM puede ser llevado a cabo de manera supervisada, semi-supervisada, no supervisada y por refuerzo (también se conoce por reforzamiento). A continuación se describe cada una de estas clasificaciones.

## 2.4.1 Aprendizaje supervisado

Para llevar a cabo este tipo de aprendizaje se parte de una colección de datos etiquetados: $\{ (\boldsymbol{x}_1 ,t_1 ) , (\boldsymbol{x}_2, t_2 ) , \cdots , (\boldsymbol{x}_k,t_k ) , \cdots , (\boldsymbol{x}_p,t_p ) \}$. Cada elemento $\boldsymbol{x}_i = [x_1 , x_2 , \cdots , x_n]^T$ se conoce en la literatura como **vector descriptor**, en el cual, cada elemento escalar o entrada del vector $x_i$, $i = 1 , 2, \cdots , n$, representa un rasgo que describe a un objeto o concepto de alguna forma. Si, por ejemplo, $\boldsymbol{x}$ es el conjunto que representa a la familia de los coches, el primer rasgo $x_{i,1}$ puede representar la potencia del motor en caballos de potencia; el segundo rasgo $x_{i,2}$ puede representar el número de asientos y así sucesivamente.

En cualquier caso, para todos los ejemplos en la base de datos, el i-ésimo rasgo siempre contendrá el mismo tipo de información. En otras palabras, si $x_{j,1}$ representa la potencia en caballos para el auto $\boldsymbol{x}_j$,entonces $x_{k,1}$ representará también el caballaje para todo ejemplo $\boldsymbol{x}_k$, donde $k = 1, 2, \cdots , p$.

Por otro lado, la etiqueta $t_k$ puede tomar diferentes formas. Por ejemplo, una de las etiquetas del conjunto $\{1 , 2, \ldots , p\}$, podría ser un real o cualquier otro objeto útil para la tarea. El objetivo de cualquier algoritmo de aprendizaje supervisado es utilizar un conjunto de datos describiendo un problema para producir un modelo que sea capaz de recibir como entrada un vector $\boldsymbol{x}$ y que en respuesta seleccione por ejemplo, del conjunto de etiquetas $\{1, 2, \ldots , p\}$ la más apropiada a dicho vector $\boldsymbol{x}$.

## 2.4.2 Aprendizaje semi-supervisado

Para este tipo de aprendizaje, el conjunto de datos se divide en dos sub-conjuntos: uno con datos etiquetados y otro no etiquetado. En muchas ocasiones, el conjunto de datos no etiquetados es mayor que el de los datos etiquetados. El objetivo en este caso, es el mismo que

para el aprendizaje supervisado. Se espera que el uso de muchos datos no etiquetados ayude al algoritmo de aprendizaje a derivar un modelo aceptable de la información. Puede sonar contradictorio que el aprendizaje se beneficie de la adición de datos no etiquetados, pareciera que añade incertidumbre al problema. La verdad es que al agregar más datos, aunque no etiquetados, se incrementa información al problema. Un buen algoritmo de aprendizaje semi-supervisado teóricamente debería de utilizar la información extra para obtener un mejor modelo.

### 2.4.3 Aprendizaje no-supervisado

En este caso, el conjunto de información solo contiene datos no etiquetados, esto es: $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k, \cdots, \boldsymbol{x}_p\}$. De nuevo, cada $\boldsymbol{x}$ es un vector descriptor con sus entradas de la forma $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$. El objetivo del aprendizaje no-supervisado es derivar un modelo a partir del conjunto de vectores para obtener un vector nuevo que pueda ser usado para resolver un problema dado.

En acumulación o agrupamiento de datos, el modelo proporciona el identificador del cúmulo correspondiente conjunto en la base de datos. En reducción de dimensionalidad, la salida del modelo será un vector descriptor con menos rasgos que el de entrada $\boldsymbol{x}$. Finalmente, en detección de valores atípicos, la salida será un número usualmente real, indicando qué tanto $\boldsymbol{x}$ difiere del valor típico de la base de datos.

### 2.4.4 Aprendizaje por reforzamiento

Este tipo de aprendizaje es una rama del AM. En este caso, se parte del hecho de que el robot se encuentra en un medio ambiente dado y que es capaz de percibir el estado de ese medio como un estado representado por el vector descriptivo $\boldsymbol{x}$. En una secuencia de imágenes: $I_1, I_2, \cdots, I_M$ adquiridas a través del sensor del robot, cada $I_k$ puede ser descrita mediante un vector descriptor $\boldsymbol{x}_k$:

$$I_k = \begin{bmatrix} \boldsymbol{x}_{k,1} \\ \boldsymbol{x}_{k,2} \\ \vdots \\ \boldsymbol{x}_{k,n} \end{bmatrix}. \tag{2.13}$$

Sobre dichos estados el robot puede ejecutar diferentes acciones. Algunas de las cuales pueden otorgarle premios a la máquina, eventualmente provocando que el robot pase de un estado a otro.

El objetivo del aprendizaje por refuerzo es asimilar lo que se conoce como "políticas". Una política es una función similar a un modelo que recibe como entrada un vector descriptor y produce un acción optimizada dentro dicho estado. En el ejemplo utilizado, el módulo de aprendizaje por refuerzo del robot puede recibir como entrada una descripción $\boldsymbol{x}$ de una imagen dada; después procesarla para, en términos de la política previamente derivada, producir como salida una velocidad, ángulo de giro, etcétera, según las necesidades. La acción se dice ser óptima si esta maximiza la recompensa esperanza promedio.

Para más detalles sobre los temas tratados en esta sección, se recomienda al autor dirigirse a la referencia [5].

## 2.5 Regresión lineal

En esta sección se estudia el problema de la regresión lineal. Se parte de dos variables medibles $x$ e $y$ pertenecientes a una muestra de $n$ individuos: $(x_1, y_1), \cdots, (x_n, y_n)$. El objetivo consiste en analizar la relación entre ambas variables, de forma que se pueda predecir o aproximar el valor de $y$ en términos de $x$. A $y$ se llama variable respuesta o dependiente, mientras que a $x$ se le llama variable regresora o explicativa.

En el caso general, el modelo viene dado como sigue:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} \cdots + \beta_p x_{ip} = \boldsymbol{x}_i^T \boldsymbol{\beta}. \tag{2.14}$$

En este caso, de acuerdo con lo visto en la definición (2.4) de la sección 2.3, la operación matemática $\boldsymbol{x}_i^T \boldsymbol{\beta}$ corresponde al producto interno o producto escalar entre los vectores $\boldsymbol{x}_i \in \mathbb{R}^{p+1\times 1}$ y $\boldsymbol{\beta} \in \mathbb{R}^{p+1\times 1}$, siendo $\boldsymbol{\beta} = [\beta_0, \beta_1 \cdots, \beta_p]^T$ el vector de parámetros a estimar, ya que los elementos $\beta_i \in \mathbb{R}$ son parámetros desconocidos con $i = 0, 1, 2, \cdots, p$. Lo interesante es que los datos de entrenamiento pueden ser usados para aproximar los valores de los parámetros $\beta_0, \beta_1 \cdots, \beta_p$.

Como es sabido, las $n$ ecuaciones dadas por la ecuación (2.14) pueden ser acomodadas en forma matricial como sigue:

$$\boldsymbol{y} = X\boldsymbol{\beta}. \tag{2.15}$$

En este caso:

1) $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ es el vector de valores observados $y_i$, $i = 1, \cdots, n$, también conocidos como variables dependientes.

2) $X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}$ es la matriz de vectores fila de la forma $\boldsymbol{x}_i^T = [1 \quad x_{i1} \quad x_{i2} \quad \cdots \quad x_{ip}]$ de regresores o variables conocidas.

3) $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$ es el vector de parámetros de dimensión $(p+1)$ o coeficientes de regresión a estimar.

Regresemos a nuestro problema de aproximar el conjunto de parámetros $\beta_0, \beta_1, \cdots, \beta_p$, a partir del conjunto de datos de entrenamiento. Esto se puede lograr al saber por cuánto el valor predicho $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} \cdots + \beta_p x_{ip}$ difiere del valor real $y_i$ para cada uno de los pares $(x_1, y_1), \cdots, (x_n, y_n)$ del conjunto de entrenamiento.

Para esto:

1) Primeramente se toma la diferencia simple entre el valor real y el estimado, es decir, el error $e_i$: $e_i = y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} \cdots + \beta_p x_{ip})$.

1) Después se toma el cuadrado de esta diferencia o error $e_i$ y se suman todas las $n$ diferencias como sigue:

$$\sum_{i=1}^{n} e_i^2 \quad = \quad \sum_{i=1}^{n} [\, y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} \cdots + \beta_p x_{ip}) \,]^2 \,. \tag{2.16}$$

A esta suma, en la literatura, se le conoce como suma residual de cuadrados y es denotada como $\text{SRC}(\boldsymbol{\beta})$, donde como ya vimos $\boldsymbol{\beta} = [\, \beta_0 \quad \beta_1 \quad \beta_1 \quad \cdots \beta_p \,]^T \in \mathbb{R}^{p+1 \times 1}$. Se busca que esta suma residual de cuadrados sea lo más pequeña posible.

En otras palabras, se busca que el valor predicho $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} \cdots + \beta_p x_{ip}$ sea los más cercano al valor real $y_i$, y esto para cada par $(x_1, y_1), \cdots, (x_n, y_n)$. Al hacer esto, como es bien sabido, resulta en la función lineal de las variables de entrada que mejor ajusta a los datos de entrenamiento:

1) En el caso de una sola variable de entrada, una línea.

2) En el caso de dos variables de entrada, un plano.

3) En el caso general, un híper-plano.

Al minimizar la suma residual de cuadrados SRC($\boldsymbol{\beta}$), se obtienen los estimados $[\,\beta_0 \quad \beta_1 \quad \beta_2 \cdots \beta_p\,]^T$. A este método en la literatura se le conoce como el método de mínimos cuadrados [6].

En lo que sigue utilizaremos este método para obtener la solución al problema deseado. Partiendo del conjunto de datos:

$$X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}. \tag{2.17}$$

De acuerdo con lo visto, si se toma la diferencia simple o error $\boldsymbol{e} \in \mathbb{R}^{n\times 1}$, entre el valor real y el estimado:

$$\begin{aligned} \boldsymbol{e} = \boldsymbol{y} - X\boldsymbol{\beta} &= \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \beta_0 + \beta_1 x_{11} + \cdots + \beta_p x_{1p} \\ \beta_0 + \beta_1 x_{21} + \cdots + \beta_p x_{2p} \\ \vdots \quad + \quad \vdots \quad + \quad \cdots \quad + \quad \vdots \\ \beta_0 + \beta_1 x_{n1} + \cdots + \beta_p x_{np} \end{bmatrix} \\ &= \begin{bmatrix} y_1 - [\,\beta_0 + \beta_1 x_{11} + \cdots + \beta_p x_{1p}\,] \\ y_2 - [\,\beta_0 + \beta_1 x_{21} + \cdots + \beta_p x_{2p}\,] \\ \vdots \quad + \quad \vdots \quad \cdots \quad \vdots \\ y_n - [\,\beta_0 + \beta_1 x_{n1} + \cdots + \beta_p x_{np}\,] \end{bmatrix} \end{aligned} \tag{2.18}$$

donde: SRC($\boldsymbol{\beta}$) $= \boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{y} - X\boldsymbol{\beta})^T(\boldsymbol{y} - X\boldsymbol{\beta})$.

Al derivar a SRC($\boldsymbol{\beta}$) = $\boldsymbol{e}^T\boldsymbol{e}$ con respecto a cada parámetro $\beta_j$, donde $j = 0, 1, 2, \cdots, p$, se obtiene el siguiente vector:

$$\begin{bmatrix} \frac{\partial \text{SRC}(\boldsymbol{\beta})}{\partial \beta_0} \\ \frac{\partial \text{SRC}(\boldsymbol{\beta})}{\partial \beta_1} \\ \vdots \\ \frac{\partial \text{SRC}(\boldsymbol{\beta})}{\partial \beta_p} \end{bmatrix} = \begin{bmatrix} \frac{\partial [\boldsymbol{e}^T\boldsymbol{e}]}{\partial \beta_0} \\ \frac{\partial [\boldsymbol{e}^T\boldsymbol{e}]}{\partial \beta_1} \\ \vdots \\ \frac{\partial [\boldsymbol{e}^T\boldsymbol{e}]}{\partial \beta_p} \end{bmatrix} = \begin{bmatrix} -2\,(1_1 \cdots 1_n)\,(\boldsymbol{y} - X\boldsymbol{\beta}) \\ -2\,(x_{11} \cdots x_{n1})\,(\boldsymbol{y} - X\boldsymbol{\beta}) \\ \vdots \\ -2\,(x_{1p} \cdots x_{np})\,(\boldsymbol{y} - X\boldsymbol{\beta}) \end{bmatrix} = -2 \begin{bmatrix} 1_1 & \cdots & 1_n \\ x_{11} & \cdots & x_{n1} \\ \vdots & \cdots & \vdots \\ x_{1p} & \cdots & x_{np} \end{bmatrix} (\boldsymbol{y} - X\boldsymbol{\beta})$$

$$= -2X^T\,(\boldsymbol{y} - X\boldsymbol{\beta})\,. \tag{2.19}$$

De manera similar, al tomar la segunda derivada de SRC($\boldsymbol{\beta}$), esto es $\frac{\partial}{\partial \beta_j}\left[\frac{\partial \text{SRC}(\boldsymbol{\beta})}{\partial \beta_j}\right]$, se obtiene la matriz: $2X^TX$, la cual se conoce en la literatura como matriz Hesiana [Web70].

- Se deja al lector demostrar la matriz Hesiana: $\frac{\partial}{\partial \beta_j}\left[\frac{\partial \text{SRC}(\boldsymbol{\beta})}{\partial \beta_j}\right] = 2X^TX$ (ver problema propuesto 2.7.11, página 79).

Por la prueba de la segunda derivada, si la matriz Hesiana de SRC($\boldsymbol{\beta}$) es un punto crítico entonces es positiva definida y, por tanto, en esa localidad tiene un mínimo local [Web71].

Al igualar a cero el vector -2$X^T(\boldsymbol{y} - X\boldsymbol{\beta})$, esto es:

$$\begin{aligned} -2X^T(\boldsymbol{y} - X\boldsymbol{\beta}) &= \mathbf{0} \Longrightarrow -2X^T\boldsymbol{y} + 2X^TX\boldsymbol{\beta} = \mathbf{0} \Longrightarrow X^T\boldsymbol{y} = X^TX\boldsymbol{\beta} \\ \therefore & \\ \boldsymbol{\beta} &= [\,X^TX\,]^{-1}\,X^T\boldsymbol{y}. \end{aligned}$$

Es decir, el vector que minimiza la suma residual de cuadrados del error $\boldsymbol{e}$, SRC($\boldsymbol{\beta}$) (también conocido como mínimos cuadrados), se obtiene el vector de estimación de parámetros $\hat{\boldsymbol{\beta}}$ de la siguiente manera:

$$\hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{bmatrix} = [\,X^TX\,]^{-1}\,X^T\boldsymbol{y}. \tag{2.20}$$

Para reforzar el conocimiento adquirido hasta el momento, procedamos a aplicar la ecuación anterior a los siguientes dos ejemplos ilustrativos.

**•• Ejemplo 2.7**

Considere el siguiente conjunto de datos de entrenamiento mostrado en la figura 2.2a: $(x_1, y_1) = (2, 3),\ (x_2, y_2) = (5, 4),\ (x_3, y_3) = (5, 6),\ (x_4, y_4) = (8, 7)$. Estimar los valores de los parámetros asociados $\beta_0$ y $\beta_1$ mediante el método de mínimos cuadrados, así como el estimado para $x = 10$.

**Solución**

El conjunto de datos de entrenamiento se presenta en la figura 2.2a, de la teoría vista:

$$X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \boldsymbol{x}_3^T \\ \boldsymbol{x}_4^T \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 5 \\ 1 & 8 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 3 \\ 4 \\ 6 \\ 7 \end{bmatrix}.$$

El vector de parámetros $\hat{\boldsymbol{\beta}}$, de acuerdo con la ecuación (2.20), viene dado como:

$$\hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix} = [\, X^T X \,]^{-1} X^T \boldsymbol{y} = \left[ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 5 \\ 1 & 8 \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 6 \\ 7 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 20 \\ 20 & 118 \end{bmatrix}^{-1} \begin{bmatrix} 20 \\ 112 \end{bmatrix} = \begin{bmatrix} 1.6389 & -0.2778 \\ -0.2778 & 0.0556 \end{bmatrix} \begin{bmatrix} 20 \\ 112 \end{bmatrix} = \begin{bmatrix} 1.6667 \\ 0.6667 \end{bmatrix}.$$

Luego entonces, de acuerdo con el algoritmo de mínimos cuadrados establecido por la ecuación (2.20) se obtienen los parámetros identificados $\hat{\beta}_0$ y $\hat{\beta}_1$; por lo que se tiene la siguiente ecuación: $y = \hat{\beta}_0 + \hat{\beta}_1 x = 1.6667 + 0.6667x$. Finalmente, el estimado de $y$ para $x = 10$ está dado por: $y = 1.6667 + 0.6667(10) = 8.3333$.

En el cuadro de código 2.3 se describe el programa en **MATLAB** cap2_MinCuadradosA.m, los datos de entrenamiento para la matriz $X$ se encuentran en la línea 5 y para el vector $\boldsymbol{y}$ en la línea 6. Observe que en la línea 7 se encuentra implementada la ecuación (2.20), la cual permite estimar el vector de parámetros $\hat{\boldsymbol{\beta}}$ por el método de mínimos cuadrados. Mientras que en la línea 8 se obtiene el valor estimado de $y$ (como escalar) para $x = 10$; la representación gráfica de la recta $y = \hat{\beta}_0 + \hat{\beta}_1 x = 1.6667 + 0.6667x = [\, 1.6667 \quad 0.6667 \,] \begin{bmatrix} 1 \\ 10 \end{bmatrix}$ se ilustra en la figura 2.2b.

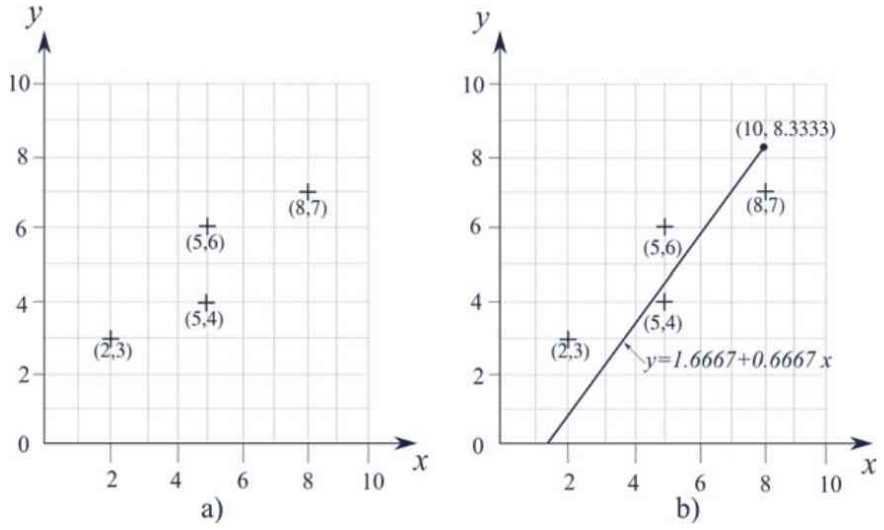



**Figura 2.2**: Conjunto de datos de entrenamiento descritos en la figura a) y en la figura b) se encuentra el valor estimado de $y$ para $x = 10$ sobrepuesto en la recta: $y = \hat{\beta}_0 + \hat{\beta}_1 x = 1.6667 + 0.6667x$.

**Código MATLAB 2.3 cap2_MinCuadradosA.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 2. Aprendizaje de Máquinas: Regresión lineal.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Programa: cap2_MinCuadradosA.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
format short % formato corto con 4 dígitos después del punto decimal.
X=[1, 2; 1, 5; 1, 5; 1, 8]; % datos de entrenamiento para la matriz X.
y=[3; 4; 6; 7]; % datos de entrenamiento del vector y.
beta_estimado=(X'*X)^(-1)*X'*y % β̂ = [β̂0; β̂1] = [X^T X]^-1 X^T y = [1.6667; 0.6667], ec. (2.20).
y_estimado=beta_estimado(1,1)+beta_estimado(2,1)*10 %valor de y para x = 10.
```

En el cuadro de código 2.4 se describe el código fuente en **MATLAB** del programa cap2_MinCuadradosB.m, el cual implementa este ejemplo. En las líneas de programación 5 y 6 se realiza la asignación de datos de entrenamiento para la matriz $X \in \mathbb{R}^{4\times 3}$ y vector $\boldsymbol{y} \in \mathbb{R}^{4\times 1}$, respectivamente. La línea 7 contiene la estimación del vector de parámetros $\hat{\boldsymbol{\beta}}$ a través de la ecuación (2.20) y en la línea 9 se obtiene la estimación de $y$ para el vector $\boldsymbol{x} = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix}$.

**Código MATLAB 2.4 cap2_MinCuadradosB.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 2. Aprendizaje de Máquinas: Regresión lineal.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap2_MinCuadradosB.m **MATLAB** versión 2020b

```
1 clc; % limpia pantalla.
2 clearvars; % remueve todas las variables del espacio actual de trabajo.
3 close all; % cierra gráficas, archivos y recursos abiertos.
4 format short % formato corto con 4 dígitos después del punto decimal.
5 X=[1, 1, 1; 1, 2, 1; 1, 1, 2; 1, 2, 2]; % datos de entrenamiento de la matriz X.
6 y=[2; 1; 1; 3]; % datos de entrenamiento del vector y.
7 beta_estimado=(X'*X)^(-1)*X'*y % β̂ = [β̂0; β̂1; β̂2] = [X^T X]^-1 X^T y = [0.25; 0.5; 0.5], ec. (2.20).
8 % Estimado de y para el vector x = [1; 3; 3]:
9 y_estimado=beta_estimado(1,1)+beta_estimado(2,1)*3+beta_estimado(3,1)*3
```

Para más detalles sobre los temas tratados en esta sección, se recomienda al autor dirigirse a la referencia [7].

**• • • Ejemplo 2.8**

Considere el siguiente conjunto de datos: $(\boldsymbol{x}_1, y_1) = \left[\begin{bmatrix}1\\1\end{bmatrix}, 2\right]$, $(\boldsymbol{x}_2, y_2) = \left[\begin{bmatrix}2\\1\end{bmatrix}, 1\right]$, $(\boldsymbol{x}_3, y_3) = \left[\begin{bmatrix}1\\2\end{bmatrix}, 1\right]$, $(\boldsymbol{x}_4, y_4) = \left[\begin{bmatrix}2\\2\end{bmatrix}, 3\right]$. Estimar los valores de los parámetros $\beta_0, \beta_1$ y $\beta_2$ mediante el método de mínimos cuadrados, así como el estimado para $\boldsymbol{x} = \begin{bmatrix}3\\3\end{bmatrix}$.

**Solución**

La presentación de los datos de entrenamiento se realiza de la siguiente manera: para la matriz $X$ se usa la ecuación (2.17):

$$X = \begin{bmatrix}\boldsymbol{x}_1^T\\ \boldsymbol{x}_2^T\\ \boldsymbol{x}_3^T\\ \boldsymbol{x}_4^T\end{bmatrix} = \begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 2\\ 1 & 2 & 2\end{bmatrix}, \text{ el vector } \boldsymbol{y} = \begin{bmatrix}2\\1\\1\\3\end{bmatrix}.$$

Entonces, usando la expresión para (2.20), el vector de parámetros $\hat{\boldsymbol{\beta}}$ viene dado:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \begin{bmatrix}\hat{\beta}_0\\ \hat{\beta}_1\\ \hat{\beta}_2\end{bmatrix} = [X^T X]^{-1} X^T \boldsymbol{y} = \left[\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 2 & 1 & 2\\ 1 & 1 & 2 & 2\end{bmatrix}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 2\\ 1 & 2 & 2\end{bmatrix}\right]^{-1}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 2 & 1 & 2\\ 1 & 1 & 2 & 2\end{bmatrix}\begin{bmatrix}2\\1\\1\\3\end{bmatrix}\\ &= \begin{bmatrix}4 & 6 & 6\\ 6 & 10 & 9\\ 6 & 910 & \end{bmatrix}^{-1}\begin{bmatrix}7\\11\\11\end{bmatrix} = \begin{bmatrix}4.75 & -1.5 & -1.5\\ -1.5 & 1 & 0\\ -1.5 & 0 & 1\end{bmatrix}\begin{bmatrix}7\\11\\11\end{bmatrix}\\ &= \begin{bmatrix}0.25\\ 0.5\\ 0.5\end{bmatrix}.\end{aligned}$$

Luego entonces, de acuerdo con la ecuación (2.20) se obtiene el siguiente regresor para: $y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = [0.25 \quad 0.5 \quad 0.5]\begin{bmatrix}1\\x_1\\x_2\end{bmatrix}$.

Finalmente, el estimado de $y$ para $x = \begin{bmatrix}3\\2\end{bmatrix}$ es: $y = [0.25 \quad 0.5 \quad 0.5]\begin{bmatrix}1\\3\\3\end{bmatrix} = 3.25.$

**• Ejemplo 2.9**

Considere un robot de un grado de libertad (gdl) tipo péndulo, el cual está formado por un servomotor y una barra de aluminio de longitud $l_p$, como el que se muestra en la figura 2.3. El péndulo se mueve en el plano vertical $x - y$, sometido a la acción de la gravedad $g$. El modelo de cinemática directa del péndulo está dado por:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_p \operatorname{sen}(q_1) \\ -l_p \cos(q_1) \end{bmatrix} \tag{2.21}$$

donde $x, y$ son las coordenadas cartesianas, $q_1$ es la coordenada articular del péndulo, cuya posición de casa ($q_1 = 0$ rad) es con respecto al eje $y_-$, es decir en la interfaz del III y IV cuadrante del plano $x - y$.

a) Obtener el modelo de regresión lineal del péndulo.

b) Implementar el algoritmo de mínimos cuadrados, ecuación (2.20) para estimar el valor de la longitud de la barra de aluminio $\hat{l}_p$. Considere un valor teórico para $l_p = 0.45$ m y un conjunto aleatorio de ángulos de rotación $q_1$.

**Solución**

El sistema de referencia del péndulo se elige de tal forma que, el eje $z$ coincida con el eje de rotación del servomotor (en este caso, perpendicular al plano $x - y$) y determinado por la regla de la mano derecha (el sentido de giro de $q_1$ es en dirección contraria a las manecillas del reloj), tal y como se ilustra en la figura 2.3. La acción de la gravedad $g$ está dirigida en dirección del eje $y$ negativo, es decir: $y_-$.

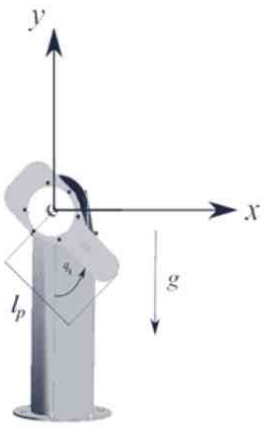



**Figura 2.3**: El robot tipo péndulo es un servomecanismo de un gdl.

El modelo de regresión lineal se establece a partir del modelo de cinemática directa (2.21) de la siguiente manera:

$$\underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\boldsymbol{y}} = \begin{bmatrix} l_p \operatorname{sen}(q_1) \\ -l_p \cos(q_1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \operatorname{sen}(q_1) \\ 1 & -\operatorname{sen}(q_1) \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}}_{\boldsymbol{\beta}}.$$

Note que en este caso $\boldsymbol{\beta} = [\,\beta_0, \beta_1\,]^T = [\,0, l_p\,]^T$; el vector $\boldsymbol{y}$ está formado por el conjunto de coordenadas cartesianas $(x, y)$ del péndulo: $\boldsymbol{y} = [\,x \quad y\,]^T$ y la matriz $X = \begin{bmatrix} 1 & \operatorname{sen}(q_1) \\ 1 & -\operatorname{sen}(q_1) \end{bmatrix}$.

Entonces, la problemática consiste en encontrar los valores estimados de $\hat{\boldsymbol{\beta}} = [\,\hat{\beta}_0, \hat{\beta}_1\,]$. Con esta finalidad, considere un conjunto de datos arbitrarios para la posición articular del péndulo $q_1$. El programa cap2_pendulo.m contiene el código fuente en **MATLAB** descrito en el cuadro de código 2.5, el cual implementa el algoritmo de estimación paramétrica por mínimos cuadrados del vector $\hat{\boldsymbol{\beta}}$. En las líneas de programación 5 y 6 se definen la dimensión del vector $\boldsymbol{\beta} \in \mathbb{R}^{p\times 1}$ ($p = 2$ , número de parámetros) y número de observaciones $n = 10$ define el número de renglones de la matriz $X \in \mathbb{R}^{2n\times p}$ y del vector $\boldsymbol{y} \in \mathbb{R}^{2n\times 1}$). En la línea 6, el lector puede probar con varios valores de observaciones, por ejemplo: $n = 20, 50, 100, 1000$.

La línea 7 genera los $n$ valores aleatorios para el ángulo $q_1$. En la línea 10 se define el valor teórico de la longitud $l_p$ de la barra del péndulo (el lector puede definir su propio valor para $l_p > 0$). Las líneas 11-12 calculan la cinemática directa del péndulo, para los $n$ datos de observaciones aleatorias de los ángulos de $q_1$. Por otro lado, la programación comprendida entre las líneas 13 y 18 se forman las componentes de las $2n$ filas o renglones y $p$ columnas de la matriz $X \in \mathbb{R}^{2n\times p}$ y $2n$ renglones del vector $\boldsymbol{y} \in \mathbb{R}^{2n\times 1}$. Con estos elementos se tiene completo el regresor del algoritmo de mínimos cuadrados (2.20). Las anteriores líneas de programación preparan los elementos requeridos y necesarios por el algoritmo de mínimos cuadrados (2.20), el cual se encuentra implementado en la línea 19.

**Importante**

La convergencia paramétrica ($\hat{\boldsymbol{\beta}} \longrightarrow \boldsymbol{\beta}$) del algoritmo de mínimos cuadrados (2.20) depende que la matriz $[\,X^T X\,]^{-1}$ sea invertible, esto se garantiza si se cumple la condición de excitación persistente (ver [7] y [43]), la cual incide directamente en los elementos de la matriz $X$. Sin embargo, dicho tópico se encuentra fuera de los alcances de esta obra.

**Código MATLAB 2.5 cap2_pendulo.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 2. Aprendizaje de Máquinas: Regresión lineal.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Programa: cap2_pendulo.m — MATLAB versión 2020b

```
1  clc; % limpia pantalla.
2  clearvars; % remueve todas las variables del espacio actual de trabajo.
3  close all; % cierra gráficas, archivos y recursos abiertos.
4  format short % formato corto con 4 dígitos después del punto decimal.
5  p=2;  % dimensión del vector de parámetros β ∈ IR^{p×1}.
6  n=10; % número de muestras (2n renglones de la matriz X ∈ IR^{2n×p}).
7  for k=1:n  % conjunto de n datos aleatorios para los ángulos q1.
8      q1(k,1)=random( 'normal' , 0, 1);
9  end
10 l_p=0.45; % valor teórico de la longitud de la barra de aluminio.
11 x_p=l_p*sin(q1); % coordenada x del péndulo, en el plano x − y.
12 y_p=-l_p*cos(q1); % coordenada y del péndulo, en el plano x − y.
13 for  k=1:n % se forma el esquema de regresión: X ∈ IR^{2n×p} y y ∈ IR^{2n×1}.
14     X(k, 1:p)=[1, sin(q1(k,1))]; % componentes x_{k,1}, x_{k,2} de X ∈ IR^{2n×p}.
15     X(k+1, 1:p)=[1, -cos(q1(k,1))]; % comp. x_{k+1,1}, x_{k+1,2} de X ∈ IR^{2n×p}.
16     y(k,1)=x_p(k,1); % componente y_{k,1} del vector y ∈ IR^{2n×1}.
17     y(k+1,1)=y_p(k,1); % componente y_{k+1,1} del vector y ∈ IR^{2n×1}.
18 end
19 beta_estimado=(X'*X)^(-1)*X'*y % β̂ = [β̂0; β̂1] = [X^T X]^{-1} X^T y ec. (2.20).
20 % despliega el resultado: β̂ = [β̂0; β̂1] = [0; 0.45].
```

**•• Ejemplo 2.10**

Considere un robot manipulador de dos gdl como el que se presenta en la figura 2.4, formado por dos servomotores y dos barras de aluminio de longitud $l_1$ y $l_2$, para las articulaciones del hombro y codo, respectivamente. El robot manipulador (también conocido como brazo robot) se mueve en el plano vertical $x-y$, sometido a la acción de la gravedad $g$. El modelo de cinemática directa del brazo robot está dado por:

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \operatorname{sen}(q_1) + l_2 \operatorname{sen}(q_1 + q_2) \end{bmatrix} \tag{2.22}$$

donde $x_r, y_r$ son las coordenadas cartesianas del extremo final del robot; $q_1, q_2$ representan las coordenadas articulares de los servomotores; la posición de casa ($q_1 = q_2 = 0$ rad) es en el primer cuadrante sobre el eje ($x_+$); el sentido de rotación es positivo cuando $q_1$ o $q_2$ giran en dirección contraria a las manecillas del reloj.

a) Obtener el modelo de regresión lineal del robot de 2 gdl.

b) Implementar el algoritmo de mínimos cuadrados, ecuación (2.20) para estimar el valor de las longitudes $\hat{l}_1$ y $\hat{l}_2$. Sean: $q_1$={0, 0.1, 0.2, 0.6, 0.8, 0.7, 0.6, 0.8, 0.78, 0.12} rad y $q_2$={0.1, 0.2, 0.3, 0.7, 0.9, 0.8, 0.7, 0.9, 0.68, 0.25} rad.

**Solución**

El modelo de regresión lineal del brazo robot de 2 gdl se establece a partir de su modelo de cinemática directa (2.22), por lo que usando el algoritmo de mínimos cuadrados (2.20) se tiene que establecer cuáles son los vectores $\boldsymbol{\beta}, \boldsymbol{y}$, así como la matriz $X$.

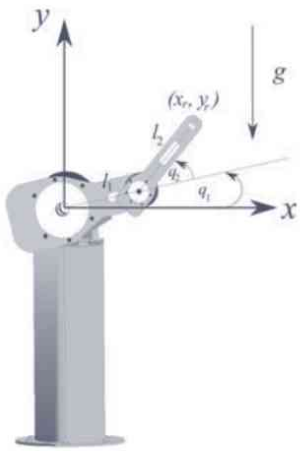



**Figura 2.4**: Brazo robot o robot manipulador de 2 gdl.

Primero, establecemos el regresor lineal de la siguiente forma:

$$\underbrace{\begin{bmatrix} x_r \\ y_r \end{bmatrix}}_{\boldsymbol{y}} = \begin{bmatrix} l_1\ \cos(q_1) + l_2\ \cos(q_1+q_2) \\ l_1\ \text{sen}(q_1) + l_2\ \text{sen}(q_1+q_2) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \cos(q_1) & \cos(q_1+q_2) \\ 1 & \text{sen}(q_1) & \text{sen}(q_1+q_2) \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}}_{\boldsymbol{\beta}}. \quad (2.23)$$

Note que $\beta_0 = 0$, $\beta_1 = l_1$ y $\beta_2 = l_2$, entonces: $\boldsymbol{\beta} = [\,\beta_0 \quad \beta_1 \quad \beta_2\,]^T = [\,0 \quad l_1 \quad l_2\,]^T \in \mathbb{R}^{3\times 1}$, por el que el número de parámetros es: $p = 3$. La matriz $X$ tiene tres columnas y dos renglones y queda determinada como: $X = \begin{bmatrix} 1 & \cos(q_1) & \cos(q_1+q_2) \\ 1 & \text{sen}(q_1) & \text{sen}(q_1+q_2) \end{bmatrix}$.

Ahora, establecer el número de mediciones que se obtendrán, lo cual define cuántos renglones tendrá el vector $\boldsymbol{y}$ y también para la matriz $X$. Es decir, por cada medición de $q_1$ y $q_2$, el vector $\boldsymbol{y}$ tendrá dos renglones (un renglón para la coordenada cartesiana $x_r$ y otro más para $y_r$). De igual forma para la matriz $X$, por cada medición se obtienen dos renglones adicionales. Por lo tanto, dado $n$ mediciones, obtenemos: $\boldsymbol{y} \in \mathbb{R}^{2n\times 1}$ y $X \in \mathbb{R}^{2n\times p}$.

En nuestro caso $n = 10$; además, se cuenta con un patrón ya definido para los ángulos de rotación $q_1$ y $q_2$, del hombro y codo, respectivamente. Sin embargo, el lector puede definir su propio patrón de rotaciones o utilizar valores aleatorios para los ángulos de las articulaciones, así como el número de mediciones $n$.

En el cuadro de código **MATLAB** 2.6 se encuentra descrito el programa cap2_robot2gdl.m, el cual implementa el algoritmo de mínimos cuadrados, ecuación (2.20), para estimar las longitudes de los eslabones del hombro y codo de un brazo robot de dos gdl. El número de parámetros a identificar se define en la línea 5, mientras que en las líneas 6 y 7 se especifica el patrón de datos para las coordenadas articulares $q_1$ y $q_2$ correspondientes a las articulaciones del hombro y codo del robot manipulador, respectivamente.

El número $n$ de observaciones del patrón de datos para las articulaciones se obtiene en la línea 8; mientras que, las líneas 10 y 11 respectivamente, se calculan con las coordenadas cartesianas $x_r$ y $y_r$ usando la cinemática directa del brazo robot, la cual es empleada en los elementos del regresor lineal a través del código comprendido entre las líneas 12 y 17. Esta información se utiliza en la línea 21 para realizar el cálculo de los valores estimados de las longitudes en los eslabones del robot (conocidos también como parámetros geométricos del robot) por medio del algoritmo de mínimos cuadrados, ecuación (2.20).

**Código MATLAB 2.6 cap2_robot2gdl.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 2. Aprendizaje de Máquinas: Regresión lineal.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Programa: cap2_robot2gdl.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
format short % formato corto con 4 dígitos después del punto decimal.
p=3; % dimensión del vector de parámetros β = [ β0, β1, β2 ]^T ∈ IR^{p×1}.
q1=[0; 0.1; 0.2; 0.6; 0.8; 0.7; 0.6; 0.8; 0.78; 0.12];
q2=[0.1; 0.2; 0.3; 0.7; 0.9; 0.8; 0.7; 0.9; 0.68; 0.25];
n=size(q1); % número de observaciones (X ∈ IR^{2n×1}, y ∈ IR^{2n×1}.
l1=0.40; l2=0.45; %longitudes de los eslabones del hombro y codo, respectivamente.
x_r=l1*cos(q1)+l2*cos(q1+q2); % cinemática directa: coord. cartesiana x.
y_r=l1*sin(q1)+l2*sin(q1+q2); % cinemática directa: coord. cartesiana y.
for k=1:n % se forman los elementos de regresión: X ∈ IR^{2n×p}, y ∈ IR^{2n×1}.
    X(k, 1:p)=[1, cos(q1(k,1)), cos(q1(k,1)+q2(k,1))]; % k-ésimo renglón de X.
    X(k+1, 1:p)=[1, sin(q1(k,1)), sin(q1(k,1)+q2(k,1))]; % k+1-ésimo renglón de X.
    y(k,1)=x_r(k,1); % k-ésimo renglón del vector y.
    y(k+1,1)=y_r(k,1); % k + 1-ésimo renglón del vector y.
end
beta_estimado=(X'*X)^(-1)*X'*y % β̂ = [β̂0; β̂1; β̂2] = [X^T X]^{-1} X^T y ec. (2.20).
% despliega el resultado: β̂ = [β̂0; β̂1; β̂2] = [0; 0.40; 0.45].
```

**• • • Ejemplo 2.11**

Sea el brazo robot de tres gdl que se ilustra en la figura 2.5, compuesto por tres servomotores y tres barras de aluminio de longitud $l_1, l_2$ y $l_3$ de las articulaciones de la base, hombro y codo, respectivamente. El robot manipulador (también conocido como robot industrial o brazo robot) se mueve en el espacio tridimensional, su modelo de cinemática directa (sin tomar en cuenta la orientación de la herramienta) es:

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} \cos(q_1)\,[\,l_2\,\cos(q_2) + l_3\,\cos(q_2+q_3)\,] \\ \operatorname{sen}(q_1)\,[\,l_2\,\operatorname{sen}(q_2) + l_3\,\operatorname{sen}(q_2+q_3)\,] \\ l_1 + l_2\,\operatorname{sen}(q_2) + l_3\,\operatorname{sen}(q_2+q_3) \end{bmatrix} \tag{2.24}$$

donde $x_r, y_r, z_r$ son las coordenadas cartesianas del extremo final del robot; $q_1, q_2, q_3$ representan las coordenadas articulares de los servomotores.

a) Obtener el modelo de regresión lineal del robot manipulador de 3 gdl.

b) Implementar el algoritmo de mínimos cuadrados, ecuación (2.20), para estimar el valor de las longitudes $\hat{l}_1, \hat{l}_2, \hat{l}_3$. Considere un patrón aleatorio formado por 1000 datos para cada articulación $q_1, q_2$ y $q_3$ del robot.

**Solución**

El modelo de regresión lineal del brazo robot de 3 gdl se establece a partir de la cinemática directa (2.24), entonces para utilizar el algoritmo de mínimos cuadrados (2.20) se requiere determinar los vectores $\boldsymbol{\beta}, \boldsymbol{y}$, así como la matriz $X$.

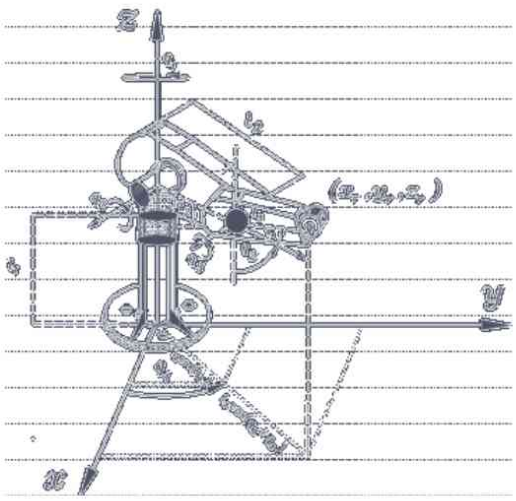


**Figura 2.5**: Robot manipulador o robot industrial de 3 gdl.

Primero, establecemos el regresor lineal del robot manipulador de 3 gdl de la siguiente forma:

$$\underbrace{\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}}_{\boldsymbol{y}} = \begin{bmatrix} \cos(q_1)\,[\,l_2\,\cos(q_2) + l_3\,\cos(q_2+q_3)\,] \\ \operatorname{sen}(q_1)\,[\,l_2\,\cos(q_2) + l_3\,\cos(q_2+q_3)\,] \\ l_1 + l_2\,\operatorname{sen}(q_2) + l_3\,\operatorname{sen}(q_2+q_3) \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & \cos(q_1)\,\cos(q_2) & \cos(q_1)\,\cos(q_2+q_3) \\ 1 & 0 & \operatorname{sen}(q_1)\,\cos(q_2) & \operatorname{sen}(q_1)\,\cos(q_2+q_3) \\ 1 & 1 & \operatorname{sen}(q_2) & \operatorname{sen}(q_2+q_3) \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}}_{\boldsymbol{\beta}}. \qquad (2.25)$$

Los componentes del vector de parámetros $\boldsymbol{\beta}$ son: $\beta_0 = 0$, $\beta_1 = l_1$, $\beta_2 = l_2$ y $\beta_3 = l_3$, entonces: $\boldsymbol{\beta} = [\,\beta_0 \quad \beta_1 \quad \beta_2 \quad \beta_3\,]^T = [\,0 \quad l_1 \quad l_2 \quad l_3\,]^T \in \mathbb{R}^{4\times 1}$, por el que el número de parámetros es: $p = 4$. El vector $\boldsymbol{y}$ tiene tres renglones por cada medición de $q_1, q_2$ y $q_3$, es decir $\boldsymbol{y} \in \mathbb{R}^{3n\times 1}$. De manera análoga, la matriz $X$ tiene cuatro columnas, sin embargo, con respecto a los renglones tiene tres filas por cada medición de $q_1, q_2$ y $q_3$. Por lo que, queda determinada como:

$$X = \begin{bmatrix} 1 & 0 & \cos(q_1)\,\cos(q_2) & \cos(q_1)\,\cos(q_2+q_3) \\ 1 & 0 & \operatorname{sen}(q_1)\,\cos(q_2) & \operatorname{sen}(q_1)\,\cos(q_2+q_3) \\ 1 & 1 & \operatorname{sen}(q_2) & \operatorname{sen}(q_2+q_3) \end{bmatrix} \in \mathbb{R}^{3n\times p}.$$

El número de mediciones $n$ define el número de renglones del vector $\boldsymbol{y} \in \mathbb{R}^{3n\times 1}$ y también la matriz $X \in \mathbb{R}^{3n\times 1}$. Se han considerado $n$=1000 datos aleatorios para $q_1, q_2$ y $q_3$. El programa cap2_robot3gdl.m contiene la implementación del algoritmo de mínimos cuadrados, ecuación (2.20), para estimar los valores geométricos del robot $\hat{\boldsymbol{\beta}} = [\,\beta_0 \quad \beta_1 \quad \beta_2 \quad \beta_3\,]^T$. La documentación técnica se encuentra en el cuadro de código **MATLAB** 2.7. El número de parámetros $p = 4$ a identificar se define en la línea 2 y el número de mediciones u observaciones ($n = 1000$) se registra en la línea 3. El código comprendido entre las líneas 4 y 8 asigna los 1000 valores aleatorios a cada una de las articulaciones del robot. Las longitudes de los eslabones de cada articulación se declaran en la línea 9.

La cinemática directa convierte las coordenadas articulares a coordenadas cartesianas, este proceso se realiza en las líneas 10-12 para obtener las coordenadas $x_r, y_r, z_r$, respectivamente. Esta información es utilizada por los elementos del regresor lineal a través del código comprendido entre las líneas 13 y 20. El algoritmo de mínimos cuadrados (2.20) que estima el valor numérico de las longitudes de cada uno de los eslabones correspondientes a las articulaciones de la base, hombro y codo del robot, se lleva a cabo en la línea de programación 21.

**Código MATLAB 2.7 cap2_robot3gdl.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 2. Aprendizaje de Máquinas: Regresión lineal.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap2_robot3gdl.m MATLAB versión 2020b

```
clc; clearvars; close all; format short
p=4; % dimensión del vector de parámetros β = [β0, β1, β2, β3]^T ∈ IR^{p×1}.
n=1000; % número de mediciones define 3n renglones del vector y y matriz X.
for k=1:n % genera n posiciones aleatorias para las articulaciones del robot.
    q1(k,1)=random( 'normal' , 0, 1); % articulación de la base q1.
    q2(k,1)=random( 'normal' , 0, 1); % articulación del hombro q2.
    q3(k,1)=random( 'normal' , 0, 1); % articulación del codo q3.
end
l1=0.85; l2=0.55; l3=0.45; % longitudes de la base, hombro y codo, respectivamente.
x_r=cos(q1).*(l2*cos(q2)+l3*cos(q2+q3)); % coordenada x_r.
y_r=sin(q1).*(l2*cos(q2)+l3*cos(q2+q3)); % coordenada y_r.
z_r=l1+l2*sin(q2)+l3*sin(q2+q3); % coordenada z_r.
for k=1:n % se calculan los elementos de los regresores para cada q1k, q2k, q3k.
    X(k, 1:p)=[1, 0, cos(q1(k,1)).*cos(q2(k,1)),cos(q1(k,1)).*cos(q2(k,1)+q3(k,1))]; % k-ésima fila.
    X(k+1, 1:p)=[1, 0, sin(q1(k,1)).*cos(q2(k,1)), sin(q1(k,1)).*cos(q2(k,1)+q3(k,1))]; % k + 1.
    X(k+2, 1:p)=[1, 1, sin(q2(k,1)), sin(q2(k,1)+q3(k,1))]; % k + 2-ésima fila de la matrix X.
    y(k,1)=x_r(k,1); % k-ésima fila del vector y correspondiente a: q1k, q2k, q3k.
    y(k+1,1)=y_r(k,1); % k + 1-ésima fila del vector y correspondiente a: q1k, q2k, q3k.
    y(k+2,1)=z_r(k,1); % k + 2-ésima fila del vector y correspondiente a: q1k, q2k, q3k.
end
beta_estimado=(X'*X)^(-1)*X'*y % β̂ = [X^T X]^{-1} X^T y ec. (2.20).
% Despliega resultado: β̂ = [β̂0  β̂1  β̂2  β̂3]^T = [0  0.85  0.55  0.45]^T, X ∈ IR^{2n×p}, y ∈ IR^{2n×1}.
```

## 2.6 Resumen

En este capítulo se estudiaron algunos de los fundamentos más representativos del AM. Primeramente, se dio una introducción a este campo de investigación que hoy por hoy se reconoce ser el principal motor de la Inteligencia Artificial. Enseguida, se describió el proceso de cómo el AM opera.

Después, se describieron de manera breve cada uno de los cuatro tipos de aprendizaje que se pueden dar en una máquina, a saber: supervisado, semi-supervisado, no supervisado y aprendizaje por refuerzo. Más adelante, se presentó un conjunto de definiciones y conceptos del álgebra lineal.

Después, se expuso el tema central de este capítulo: regresión lineal e ilustrado con ejemplos resueltos para una fácil asimilación del material expuesto.

## 2.7 Problemas propuestos

En esta sección se proporcionan un conjunto de problemas propuestos para ser resueltos por el lector, cuyo desarrollo y solución le permitirán reafirmar el conjunto de conceptos vistos en este capítulo.

2.7.1 Sean los siguientes dos vectores:

$$\boldsymbol{a} = \begin{bmatrix} 2 \\ -3 \\ 5 \\ -6 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -7 \\ -4 \\ -12 \\ 2 \end{bmatrix}.$$

a) Realizar el producto punto: $\boldsymbol{a}^T\boldsymbol{b}$.

b) Obtener $\boldsymbol{b}^T\boldsymbol{a}$.

c) ¿Son los mismos resultados obtenidos en los incisos a) y b)?

d) ¿Dependerá el resultado del orden de multiplicación de los vectores?

e) Desarrolle un algoritmo en **MATLAB**, para obtener el producto punto o producto escalar entre dos vectores de cualquier dimensión euclidiana.

Justifique adecuadamente sus respuestas.

2.7.2 Considere las siguientes matrices:

$$A = \begin{bmatrix} -2 & 6 & -1 \\ -5 & -2 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 4 & -5 \\ 4 & 7 & 1 \\ -3 & -7 & 3 \end{bmatrix}.$$

a) Obtenga las respectivas matrices transpuestas, es decir: $A^T$ y $B^T$.

b) Desarrolle un programa en **MATLAB** que obtenga la transpuesta de cualquier matriz.

2.7.3 A continuación se declaran las siguientes matrices:

$$A = \begin{bmatrix} 2 & 4 & -5 \\ 4 & 7 & 1 \\ -3 & -7 & 3 \end{bmatrix}, \; B = \begin{bmatrix} 4 & 3 & 8 \\ 8 & -3 & 9 \\ 2 & 5 & -1 \end{bmatrix}, \; C = \begin{bmatrix} 2 & -4 & 1 \\ -6 & 8 & -3 \\ 4 & -9 & 5 \end{bmatrix}.$$

a) Realice las siguientes operaciones:

$A + B$,
$A + C$,
$B + C$,
$A + B + C$,
$A + B - C$.

b) Escriba un programa en **MATLAB** que le permita verificar todos los resultados numéricos del inciso a).

2.7.4 Sean las siguientes tres matrices:

$$A = \begin{bmatrix} 1 & 7 & 2 \\ -4 & 3 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 6 & 7 \\ 3 & -5 \\ 2 & -3 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

a) ¿Qué condiciones se deben cumplir para realizar el producto entre dos matrices?

b) Obtenga $AB$.

c) ¿Es posible realizar los siguientes productos: $AC$ y $BC$?

d) Escriba un programa en **MATLAB** para realizar el producto entre dos matrices. En la elaboración del programa, tome en cuenta los requisitos solicitados en el inciso a).

2.7.5 Calcule el determinante de las siguientes matrices:

$$A = \begin{bmatrix} 3 & -6 \\ 2 & 8 \end{bmatrix}, \; B = \begin{bmatrix} -2 & 3 & 6 \\ 4 & 8 & -5 \\ 9 & -8 & 7 \end{bmatrix}$$

2.7.6 Calcule la matriz adjunta de las siguientes matrices:

$$A = \begin{bmatrix} 3 & -6 \\ 2 & 8 \end{bmatrix}, \; B = \begin{bmatrix} -2 & 3 & 6 \\ 4 & 8 & -5 \\ 9 & -8 & 7 \end{bmatrix}.$$

2.7.7 Calcule la matriz inversa de las siguientes matrices:

$$A = \begin{bmatrix} 3 & -6 \\ 2 & 8 \end{bmatrix}, \; B = \begin{bmatrix} -2 & 3 & 6 \\ 4 & 8 & -5 \\ 9 & -8 & 7 \end{bmatrix}, \; C = \begin{bmatrix} -3 & 3 & 6 & 1 \\ 4 & 2 & 8 & -5 \\ 6 & -6 & 7 & 2 \\ 7 & 2 & -2 & 9 \end{bmatrix}.$$

2.7.8 Considere el conjunto de datos de entrenamiento mostrado en la figura 2.6: $(x_1, y_1) = (1, 2)$, $(x_2, y_2) = (2, 4)$, $(x_3, y_3) = (4, 6)$ y $(x_4,\ y_4) = (6, 7)$. Estimar los valores de los parámetros asociados $\beta_0$ y $\beta_1$ mediante el método de mínimos cuadrados, así como el estimado para $x = 9$.

2.7.9 Considere el siguiente conjunto de datos:

$$(\boldsymbol{x}_1, y_1) = \left[ \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \; 4 \right], \; (\boldsymbol{x}_2, y_2) = \left[ \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \; 3 \right]$$

$$(\boldsymbol{x}_3, y_3) = \left[ \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \; 3 \right], \; (\boldsymbol{x}_4, y_4) = \left[ \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \; 5 \right].$$

Estimar los valores de los parámetros mediante el método de mínimo cuadrados, así como el estimado para $\boldsymbol{x} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$.

2.7.10 Elabore en **MATLAB** un programa que permita el cálculo, mediante el método de mínimos cuadrados, descrito en la sección 2.5, de los parámetros $\beta_0,\ \beta_1, \cdots$ para un problema dado. Verifique la correcta funcionalidad de dicho programa al aplicarlo a los problemas 2.8 y 2.9 antes descritos.

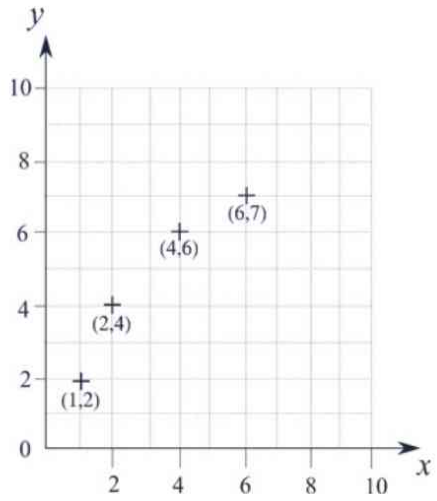



**Figura 2.6**: Conjunto de datos para el problema 2.8.

2.7.11 Demostrar que la matriz Hessiana $\frac{\partial^2 \text{SRC}(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_j} = 2X^T X$ (ver página 62).

2.7.12 Considere el siguiente sistema mecatrónico tipo centrífuga, como el que se muestra en la figura 2.7, el cual corresponde a un electrodoméstico conocido como máquina lavadora. El modelo cinemático proporciona las coordenadas cartesianas del extremo final de la barra $l_2$ y se encuentra dada por:

$$\begin{aligned} x_c &= l_2 \cos(\phi) \cos(q_1) \\ y_c &= l_2 \cos(\phi) \operatorname{sen}(q_1) \\ z_c &= l_1 + \frac{\kappa}{2} + l_2 \operatorname{sen}(\phi) \end{aligned}$$

donde:

- $x_c$, $y_c$, $z_c$ son las coordenadas cartesianas del extremo final de la barra $l_2$.
- $q_1$ es el ángulo de rotación del servomotor, el cual gira alrededor del eje $z$ en el sentido contrario a las manecillas del reloj.
- $l_1$ es la longitud de la base.
- $l_2$ es la longitud de la segunda barra, la cual mantiene un ángulo de rotación $\phi$ con respecto a la horizontal; este ángulo $\phi$ permanece constante en todo momento.
- $\kappa$ es el radio del pequeño cilindro que sirve como acoplamiento mecánico entre la barra $l_1$ y $l_2$. Por razones de simetría considere que dicho cilindro tiene la misma altura $\kappa$.

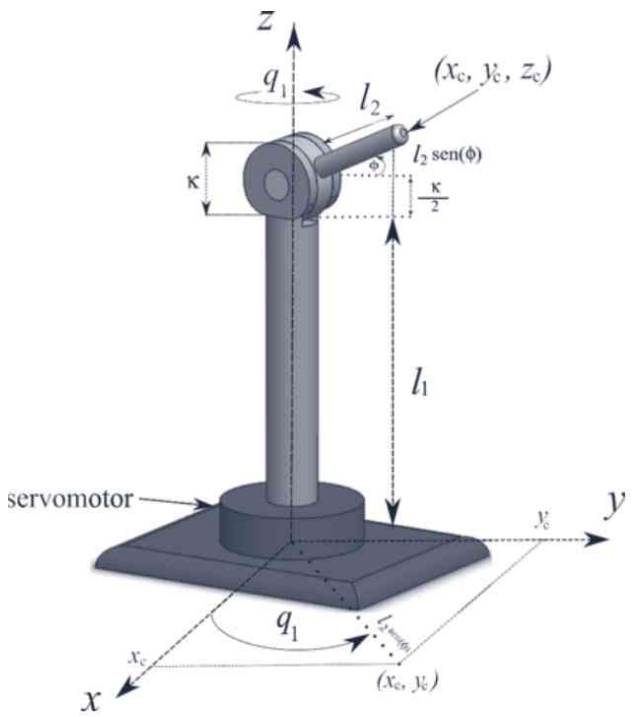



**Figura 2.7**: Sistema mecatrónico tipo centrífuga.

Sean los valores conocidos para $l_1 = 0.45$ m, el diámetro $\kappa = 0.3$ m y el siguiente patrón de datos de entrenamiento: $q_1 = \{-0.1022,$ -0.2414, 0.3192, 0.3129, -0.8649, -0.0301, -0.1649, 0.6277, 1.0933, $1.1093\}$ rad; para las coordenadas cartesianas los siguientes: $x_c = \{0.2852, 0.2784, 0.2722, 0.2728, 0.1860, 0.2866, 0.2828, 0.2321, 0.1318, 0.1277\}$ m; $y_c = \{-0.0293, -0.0686, 0.09, 0.0882, \ -0.2182, -0.0086, -0.0471, 0.1684, 0.2546, 0.2567\}$ m.

a) ¿Cómo puede establecer el modelo de regresión lineal de la centrífuga a partir de su modelo de cinemática directa?

b) Implementar en **MATLAB** el algoritmo de mínimos cuadrados, ecuación (2.20) para estimar el valor de los parámetros $\hat{\beta}_0$ y $\hat{\beta}_1 = \hat{l}_2 \text{ sen}(\hat{\phi})$.

c) ¿Qué procedimiento puede proponer para encontrar de manera individual el valor numérico de la barra $\hat{l}_2$ y el ángulo $\hat{\phi}$?

d) Note que la coordenada $z_c$ es una constante, luego entonces: ¿Cómo puede determinar el valor numérico de la coordenada $z_c$?

Capítulo 3

# Discriminación lineal y regresión logística

$$\hat{\Sigma} = \frac{1}{n_T - p} \sum_{k=1}^{p} \sum_{i=1}^{p} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_{C^k})^T (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_{C^k})$$

$$\boldsymbol{\beta}(k+1) = \boldsymbol{\beta}(k) + [X^T W X]^{-1} X^T [\boldsymbol{y} - \boldsymbol{p}]$$

# Descripción del capítulo

En este capítulo se presenta el problema de la clasificación de patrones a través del método de discriminación lineal. Por otro lado, se describe el problema de la regresión logística, útil también para la clasificación de patrones a través de la estimación de la probabilidad de sus correspondientes clases. Ambas técnicas son presentadas mediante ejemplos ilustrativos acompañados por programación en código fuente **MATLAB** para una fácil comprensión. Al final del capítulo, se muestra un conjunto de problemas propuestos para que el lector ponga en acción sus habilidades y reafirme los diferentes conceptos, definiciones y métodos aprendidos.

Los siguientes temas son abordados:

Discriminación lineal.

Región logística.

Ejemplos ilustrativos.

Programación en **MATLAB**.

# 3.1 Introducción

Este capítulo estudia el problema de la clasificación de patrones a través del método de discriminación lineal, el cual contempla la probabilidad de $p$ número de clases; para propósitos de ilustración y una fácil exposición al lector, sin pérdida de generalidad, se tratará el problema bi-clase, es decir, cuando $p = 2$. La construcción de la función de probabilidad se lleva a cabo de manera indirecta usando el conocido teorema de Bayes.

Primeramente, se construye la función de probabilidad a posteriori, como ya se dijo, a través del poderoso teorema de Bayes. Después, se modela la función de probabilidad mediante distribuciones gaussianas multivariables. Enseguida, se construyen las correspondientes funciones de discriminación lineal. Finalmente, dichas funciones de discriminación son estimadas y utilizadas.

Por otro lado, también se presenta el problema de regresión logística. Al igual que en el caso del análisis de discriminación lineal, se desea estimar la probabilidad de clases tal que, se pueda elegir la clase $C^k$ con la probabilidad más alta; pero en lugar de estimar la probabilidad de manera indirecta, como en el caso de discriminación lineal, ahora se determina de manera directa.

En primer lugar se modela la función de probabilidad a posteriori. Para esto, se utiliza el logaritmo del cociente de probabilidades de una clase y su complemento. Enseguida, se estima la función de probabilidad mediante el producto de probabilidades de los datos, lo cual permite obtener la llamada función de verosimilitud. Finalmente, esta función de verosimilitud es maximizada a través del método de Newton-Raphson multivariable.

Los ejemplos presentados para ambas técnicas, regresión lineal y discriminación logística son ilustrados mediante programación en código fuente para MATLAB, con su adecuado acompañamiento pedagógico y documentación técnica, para una fácil comprensión.

Al final del capítulo, el lector encontrará un conjunto de ejercicios que le permitirán reafirmar el conocimiento adquirido y también mejorar sus habilidades de programador a través del diseño de código fuente en MATLAB.

## 3.2 Discriminación lineal

En esta sección se estudia el problema de la clasificación de patrones mediante el método de discriminación lineal. Para esto se deberá primero calcular la probabilidad a posteriori $\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})$, esto es, la probabilidad de que $y$ sea la clase $C^k,\ k = 1, 2, \cdots, p$ (con $p$ el número de clases), dado que el vector $\boldsymbol{\chi}$ sea el vector $\boldsymbol{x}$.

Una vez calculada esta probabilidad para un vector $\boldsymbol{x}$ dado, se seleccionará la clase $C^k$ para la cual la probabilidad $\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})$ sea la más grande. Una vez esto, el patrón $\boldsymbol{x}$ es clasificado como perteneciente a la clase $C^k$. Para lograrlo, se procede como sigue:

1) **Construcción de la función de probabilidad a posteriori**. A través del teorema de Bayes, se puede demostrar que la probabilidad a posteriori $\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})$ que el vector $\boldsymbol{\chi}$ sea el vector $\boldsymbol{x}$, dado que $y$ sea la clase $C^k$ viene dada como:

$$\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x}) = \frac{\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x}|y = C^k)\mathcal{P}(y = C^k)}{\sum_{j=1}^{p} \mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x}|y = C^j)\mathcal{P}(y = C^j)}. \tag{3.1}$$

2) **Modelado de la función de probabilidad a posteriori**. Para poder estimar esta probabilidad $\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})$ se asume que la distribución condicional de $\boldsymbol{\chi}$ dado $y = C^k$ puede ser modelada a través de una función del tipo Gaussiano multivariable:

$$N(\boldsymbol{\mu}_k, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}. \tag{3.2}$$

Luego, entonces:

$$\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x}) = \frac{\mathcal{P}(y=C^k)\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}}{\sum_{j=1}^{p} \mathcal{P}(y=C^j)\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)}} \tag{3.3a}$$

$$= \frac{\mathcal{P}(y=C^k)e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}}{\sum_{j=1}^{p} \mathcal{P}(y=C^j)e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)}}. \tag{3.3b}$$

Se puede demostrar que el denominador de la ecuación anterior es igual a $(2\pi)^{\frac{n}{2}}|\sum|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x})$, en consecuencia:

$$\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x}) = \frac{\mathcal{P}(y = C^k)e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}}{(2\pi)^{\frac{n}{2}}|\sum|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x})}. \tag{3.4}$$

3) **Construcción de funciones de discriminación lineal**. Tomando en cuenta que se desea escoger la clase $C^k$ para la cual la probabilidad a posteriori dada por la ecuación (3.4) sea la más grande. Debido a que la función logaritmo preserva el orden, el maximizar la probabilidad a posteriori $\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})$ es equivalente a maximizar su logaritmo.

Al tomar el logaritmo de la ecuación (3.4), se tiene:

$$\begin{aligned}
\log\left(\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})\right) &= \log\left(\frac{\mathcal{P}(y=C^k)e^{-\frac{1}{2}(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,)^T\Sigma^{-1}(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,)}}{(2\pi)^{\frac{n}{2}}|\sum|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi}=\boldsymbol{x})}\right) \\
&= \log\left(\mathcal{P}(y = C^k)e^{-\frac{1}{2}(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,)^T\Sigma^{-1}(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,)}\right) \\
&\quad -\log\left((2\pi)^{\frac{n}{2}}\left|\sum\right|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x})\right) \\
&= \log\left(\mathcal{P}(y = C^k)\right) - \frac{1}{2}\,(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,)^T\,\Sigma^{-1}\,(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,) \\
&\quad -\log\left((2\pi)^{\frac{n}{2}}\left|\sum\right|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x})\right) \\
&= \log\left(\mathcal{P}(y = C^k)\right) - \frac{1}{2}\,[\,\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_{C^k}\,] \\
&\quad -\log\left((2\pi)^{\frac{n}{2}}\left|\sum\right|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x})\right) \\
&= \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_{C^k} - \frac{1}{2}\boldsymbol{\mu}_{C^k}^T\,\Sigma^{-1}\boldsymbol{\mu}_{C^k} + \log\left(\mathcal{P}(y = C^k)\right) \\
&\quad -\frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} - \log\left((2\pi)^{\frac{n}{2}}\left|\sum\right|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x})\right).
\end{aligned}$$

Ya que el término $-\frac{1}{2}\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} - \log\left((2\pi)^{\frac{n}{2}}\,|\sum|^{\frac{1}{2}}\,\mathcal{P}(\boldsymbol{\chi} = \boldsymbol{x})\right)$ no depende de $C^k$, entonces al maximizar: $\log\left(\frac{\mathcal{P}(y=C^k)e^{-\frac{1}{2}(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,)^T\Sigma^{-1}(\,\boldsymbol{x}-\boldsymbol{\mu}_{C^k}\,)}}{(2\pi)^{\frac{n}{2}}|\sum|^{\frac{1}{2}}\mathcal{P}(\boldsymbol{\chi}=\boldsymbol{x})}\right)$ es equivalente maximizar el término $\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_{C^k} - \frac{1}{2}\boldsymbol{\mu}_{C^k}^T\,\Sigma^{-1}\boldsymbol{\mu}_{C^k} + \log\left(\mathcal{P}(y = C^k)\right)$.

4) **Estimación de funciones de discriminación lineal**. De acuerdo con lo visto, nuestras funciones de discriminación lineal tendrán la siguiente forma:

$$fd_{C^k}(\boldsymbol{x}) \;=\; \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_{C^k} \;-\frac{1}{2}\boldsymbol{\mu}_{C^k}^T\,\Sigma^{-1}\boldsymbol{\mu}_{C^k} \;+\log\left(\mathcal{P}(y=C^k)\right) \qquad (3.5)$$

Para poder trabajar este tipo de funciones, es suficiente con estimar los valores de los parámetros $\mathcal{P}(y=C^k)$, $\boldsymbol{\mu}_k$ y $\Sigma$.

La probabilidad $\mathcal{P}(y=C^k)$ para cada clase $C^k$ se estima como el cociente entre el número de muestras o patrones de la clase $C^k(n_{C^k})$ y el número total de muestras del problema $(n_T)$, esto es: $p_{C^k} = \frac{n_{C^k}}{n_T}$.

El vector promedio para clase $C^k$, $\boldsymbol{u}_{C^k}$ puede ser estimado a través de la siguiente formulación:

$$\hat{\boldsymbol{\mu}}_{C^k} \;=\; \frac{1}{n_{C^k}}\left\{\begin{bmatrix}\boldsymbol{x}_{1,1}\\ \boldsymbol{x}_{1,2}\\ \vdots\\ \boldsymbol{x}_{1,n}\end{bmatrix}+\begin{bmatrix}\boldsymbol{x}_{2,1}\\ \boldsymbol{x}_{2,2}\\ \vdots\\ \boldsymbol{x}_{2,n}\end{bmatrix}+\cdots+\begin{bmatrix}\boldsymbol{x}_{n_{C^k},1}\\ \boldsymbol{x}_{n_{C^k},2}\\ \vdots\\ \boldsymbol{x}_{n_{C^k},n}\end{bmatrix}\right\}$$

$$= \frac{1}{n_{C^k}}\sum_{\substack{\text{vectores de}\\ \text{la clase } C^k}}\begin{bmatrix}x_{C^k,1}\\ x_{C^k,2}\\ \vdots\\ x_{C^k,n}\end{bmatrix} = \begin{bmatrix}\frac{1}{n_{C^k}}\sum_{\substack{\text{primeros}\\ \text{componentes}}} x_{C^k,1}\\ \frac{1}{n_{C^k}}\sum_{\substack{\text{segundos}\\ \text{componentes}}} x_{C^k,2}\\ \vdots\\ \frac{1}{n_{C^k}}\sum_{\substack{n\text{-ésimos}\\ \text{componentes}}} x_{C^k,n}\end{bmatrix}$$

$$= \frac{1}{n_{C^k}}\left[\,\boldsymbol{x}_1+\boldsymbol{x}_2+\cdots+\boldsymbol{x}_n\,\right] \quad \text{vectores de clase } C^k.$$

Por otro lado, la matriz de covarianza $\Sigma$ puede ser estimada a través de la siguiente expresión:

$$\hat{\Sigma} \;=\; \frac{1}{n_T-p}\sum_{k=1}^{p}\sum_{\substack{\boldsymbol{x}_i\,\in\,\{\text{vectores}\\ \text{de la clase } C^k\}}}^{p}(\,\boldsymbol{x}-\hat{\boldsymbol{\mu}}_{C^k}\,)^T\,(\,\boldsymbol{x}-\hat{\boldsymbol{\mu}}_{C^k}\,) \qquad (3.6)$$

donde $p$ es el número de clases.

**• • • Ejemplo 3.1**

Considere el siguiente conjunto de datos mostrado en la figura 3.1(a):

$$\left\{\boldsymbol{x}_1 = \begin{bmatrix}2\\3\end{bmatrix},\ \boldsymbol{x}_2 = \begin{bmatrix}3\\5\end{bmatrix},\ \boldsymbol{x}_3 = \begin{bmatrix}5\\6\end{bmatrix} \in C^1\right\} \text{ y } \left\{\boldsymbol{x}_4 = \begin{bmatrix}5\\2\end{bmatrix},\ \boldsymbol{x}_5 = \begin{bmatrix}7\\4\end{bmatrix},\ \boldsymbol{x}_6 = \begin{bmatrix}8\\6\end{bmatrix} \in C^2\right\}$$

Note que tres vectores pertenecen a la clase $C^1$, mientras que otros tres pertenecen a la clase $C^2$. Hacer lo siguiente:

a) Encontrar los estimados de los discriminantes: $fd_{C^1}(\boldsymbol{x})$ y $fd_{C^2}(\boldsymbol{x})$.

b) Encontrar la línea de separación o de decisión entre ambas clases, y

c) Determinar la clase de los patrones: $\boldsymbol{x}_1 = \begin{bmatrix}4\\4\end{bmatrix}$, $\boldsymbol{x}_2 = \begin{bmatrix}6\\4\end{bmatrix}$.

**Solución**

Para este ejemplo el número de rasgos $n = 2$ y el número de clases $p = 2$. Por otro lado, el número total de muestras $n_T = 6$, mientras que el número de muestras para la clase $C^1$ es $n_1 = 3$ y el número de muestras para la clase $C^2$ es $n_2 = 3$.

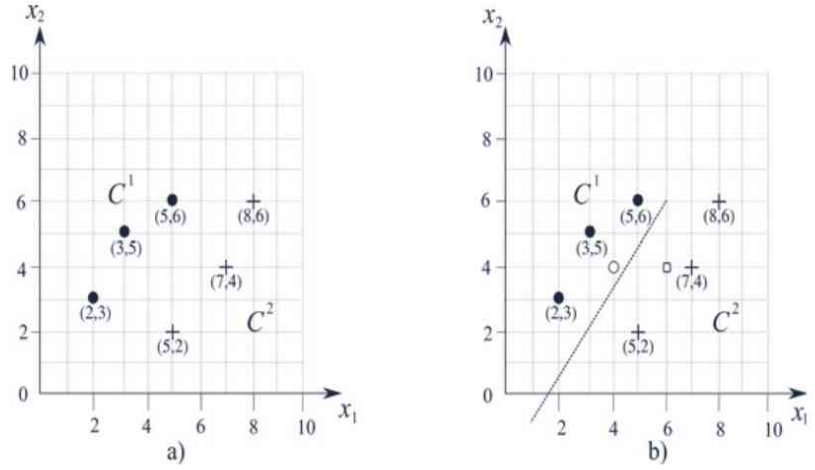



**Figura 3.1**: a) Conjunto de datos, función de separación entre las dos clases y posiciones de dos patrones a clasificar, el primero con coordenadas (4,4) representado por un círculo y el segundo con coordenadas (6,4) denotado por un cuadrado, b).

a) A continuación se ilustra el proceso numérico:

$$\begin{aligned}
\hat{p}_{C^1} &= \frac{n_{C^1}}{n_T} = \frac{3}{6} = 0.5 \\
\hat{p}_{C^2} &= \frac{n_{C^2}}{n_T} = \frac{3}{6} = 0.5
\end{aligned}$$

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_{C^1} &= \frac{1}{n_{C^1}} \sum_{\text{vectores de clase } C^1} \begin{bmatrix} x_{C^1,1} \\ x_{C^1,2} \\ \vdots \\ x_{C^1,n} \end{bmatrix} = \frac{1}{3}\left(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3\right) \\
&= \frac{1}{3}\left(\begin{bmatrix} 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 3 \\ 5 \end{bmatrix} + \begin{bmatrix} 5 \\ 6 \end{bmatrix}\right) = \begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_{C^2} &= \frac{1}{n_{C^2}} \sum_{\text{vectores de clase } C^2} \begin{bmatrix} x_{C^2,1} \\ x_{C^2,2} \\ \vdots \\ x_{C^2,n} \end{bmatrix} = \frac{1}{3}\left(\boldsymbol{x}_4 + \boldsymbol{x}_5 + \boldsymbol{x}_6\right) \\
&= \frac{1}{3}\left(\begin{bmatrix} 5 \\ 2 \end{bmatrix} + \begin{bmatrix} 7 \\ 4 \end{bmatrix} + \begin{bmatrix} 8 \\ 6 \end{bmatrix}\right) = \begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix}
\end{aligned}$$

$$\hat{\Sigma} = \frac{1}{n_T - p} \sum_{k=1}^{p} \sum_{\text{vectores } \boldsymbol{x} \text{ de clase } C^k} \left(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{C^k}\right)\left(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{C^k}\right)^T$$

$$\begin{aligned}
&= \frac{1}{6-2}\left\{\left[\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix}\right]\left[\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix}\right]^T + \left[\begin{bmatrix} 3 \\ 5 \end{bmatrix} - \begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix}\right]\left[\begin{bmatrix} 3 \\ 5 \end{bmatrix} - \begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix}\right]^T \right. \\
&\quad + \left[\begin{bmatrix} 5 \\ 6 \end{bmatrix} - \begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix}\right]\left[\begin{bmatrix} 5 \\ 6 \end{bmatrix} - \begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix}\right]^T + \left[\begin{bmatrix} 5 \\ 2 \end{bmatrix} - \begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix}\right]\left[\begin{bmatrix} 5 \\ 2 \end{bmatrix} - \begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix}\right]^T \\
&\quad \left. + \left[\begin{bmatrix} 7 \\ 4 \end{bmatrix} - \begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix}\right]\left[\begin{bmatrix} 7 \\ 4 \end{bmatrix} - \begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix}\right]^T + \left[\begin{bmatrix} 8 \\ 6 \end{bmatrix} - \begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix}\right]\left[\begin{bmatrix} 8 \\ 6 \end{bmatrix} - \begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix}\right]^T \right\}
\end{aligned}$$

$$\begin{aligned}
&= \tfrac{1}{6-2}\left\{\begin{bmatrix} -\frac{4}{3} \\ -\frac{5}{3} \end{bmatrix}\begin{bmatrix} -\frac{4}{3} & -\frac{5}{3} \end{bmatrix} + \begin{bmatrix} -\frac{1}{3} \\ \frac{1}{3} \end{bmatrix}\begin{bmatrix} -\frac{1}{3} & \frac{1}{3} \end{bmatrix} + \begin{bmatrix} \frac{5}{3} \\ \frac{4}{3} \end{bmatrix}\begin{bmatrix} \frac{5}{3} & \frac{4}{3} \end{bmatrix} + \begin{bmatrix} -\frac{5}{3} \\ -\frac{6}{3} \end{bmatrix}\begin{bmatrix} -\frac{5}{3} & -\frac{6}{3} \end{bmatrix} \right. \\
&\quad \left. + \begin{bmatrix} \frac{1}{3} \\ 0 \end{bmatrix}\begin{bmatrix} \frac{1}{3} & 0 \end{bmatrix} + \begin{bmatrix} \frac{4}{3} \\ \frac{6}{3} \end{bmatrix}\begin{bmatrix} \frac{4}{3} & \frac{6}{3} \end{bmatrix} \right\}.
\end{aligned}$$

Por lo que, la matriz de covarianza inversa $\hat{\Sigma}^{-1}$ y los estimados de las funciones discriminantes obtienen las siguientes formas:

$$\hat{\sum}^{-1} = \begin{bmatrix} 4.4272 & -3.6117 \\ -3.6117 & 3.2621 \end{bmatrix}$$

$$\begin{aligned} fd_{C^1}(\boldsymbol{x}) &= \boldsymbol{x}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^1} - \frac{1}{2}\hat{\boldsymbol{\mu}}_{C^1}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^1} + \log\mathcal{P}(y = C^1) \\ &= \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 4.4272 & -3.6117 \\ -3.6117 & 3.2621 \end{bmatrix}\begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix} - \frac{1}{2}\begin{bmatrix} \frac{10}{3} & \frac{14}{3} \end{bmatrix}\begin{bmatrix} 4.4272 & -3.6117 \\ -3.6117 & 3.2621 \end{bmatrix}\begin{bmatrix} \frac{10}{3} \\ \frac{14}{3} \end{bmatrix} + \log(0.5) \\ &= \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} -2.0971 \\ 3.1845 \end{bmatrix} - 3.9353 - 0.6931 \\ &= -2.0971x_1 + 3.1845x_2 - 4.6284. \end{aligned}$$

$$\begin{aligned} fd_{C^2}(\boldsymbol{x}) &= \boldsymbol{x}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^2} - \frac{1}{2}\hat{\boldsymbol{\mu}}_{C^2}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^2} + \log\mathcal{P}(y = C^2) \\ &= \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 4.4272 & -3.6117 \\ -3.6117 & 3.2621 \end{bmatrix}\begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix} - \frac{1}{2}\begin{bmatrix} \frac{20}{3} & \frac{12}{3} \end{bmatrix}\begin{bmatrix} 4.4272 & -3.6117 \\ -3.6117 & 3.2621 \end{bmatrix}\begin{bmatrix} \frac{20}{3} \\ \frac{12}{3} \end{bmatrix} + \log(0.5) \\ &= \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 15.068 \\ -11.0291 \end{bmatrix} - 28.167 - 0.6931 \\ &= 15.068x_1 - 11.029x_2 - 28.8601. \end{aligned}$$

b) La frontera de separación o de decisión entre ambas clases $C^1$ y $C^2$ puede ser obtenida primeramente al restar las dos funciones de discriminación antes obtenidas:

$$\begin{aligned} fd_{C^2}(\boldsymbol{x}) - fd_{C^1}(\boldsymbol{x}) &= 15.0679x_1 - 11.0296x_2 - 28.8601 - (-2.0971x_1 + 3.1845x_2 - 4.6273) \\ &= 17.165x_1 - 14.2141x_2 - 24.2328. \end{aligned}$$

Enseguida, igualamos a cero y despejamos para la variable $x_2$ como sigue:

$$x_2 = 1.2076x_1 - 1.7048.$$

La figura 3.1b muestra la frontera de separación entre las dos clases $C^1$ y $C^2$. Note cómo esta frontera pasa entre las dos clases, incluso tiende a pasar en medio de ambas clases.

c) La pertenencia de un patrón $\boldsymbol{x}$ a una de las dos clases $C^1$ y $C^2$ queda determinada por el signo del resultado al sustituir dicho patrón en la ecuación de la frontera de decisión obtenida en el paso b). Si el signo es negativo, el patrón $\boldsymbol{x}$ es clasificado en la clase $C^1$; cuando el signo obtenido resulta positivo, entonces el patrón $\boldsymbol{x}$ es clasificado en la clase $C^2$.

Para el patrón $\boldsymbol{x} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$: $17.165(4) - 14.2141(4) - 24.2328 = -12.4292$, por lo tanto, el patrón $\boldsymbol{x} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$ es puesto en la clase $C^1$. En la figura 3.1b se puede verificar que el patrón se encuentra de lado izquierdo de la frontera de decisión.

Ahora, con respecto al patrón $\boldsymbol{x} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$: $17.165(6) - 14.2141(4) - 24.2328 = 21.9008$, esto significa que el patrón $\boldsymbol{x} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$ es puesto en la clase $C^2$. Por lo que se puede verificar que el patrón se encuentra de lado derecho de la frontera de decisión, tal y como se ilustra en la figura 3.1b.

Para este ejemplo, el procedimiento anteriormente descrito ilustra los cálculos numéricos realizados a mano. Otra opción interesante para resolver el problema planteado es por medio de un algoritmo escrito en lenguaje **MATLAB**. Con esta finalidad, a continuación se detalla la programación requerida que no solo reproduce los cálculos numéricos ya mostrados, también presenta programación simbólica para obtener las funciones de discriminación.

El cuadro de código 3.1 describe el código **MATLAB** del programa cap3_DiscriminacionLineal.m, el cual reproduce el desarrollo numérico de este ejemplo. Las líneas de programación 6-9 registran el conjunto de datos para llevar a cabo las funciones de discriminación.

Los vectores $\hat{\boldsymbol{\mu}}_{C^1}$ y $\hat{\boldsymbol{\mu}}_{C^2}$ se encuentran definidos en las líneas 21 y 22, respectivamente. La línea 15 reproduce los datos mostrados en la figura 3.1a. Las líneas de programación 25 y 30 contienen el procedimiento para obtener la matriz de covarianza $\hat{\Sigma}$, la cual se realiza de acuerdo con la expresión (3.6). Sin embargo, el cálculo completo de esta matriz se obtiene en la línea 31.

En la línea 33 se utiliza programación simbólica para obtener las funciones de discriminación $fd_{C^1}(\boldsymbol{x})$ y $fd_{C^2}(\boldsymbol{x})$ en las líneas 37 y 40, respectivamente. Por otro lado, los valores numéricos de dichas funciones se encuentran en las líneas 35 y 38, respectivamente. La reproducción de la figura 3.1b se obtiene en la línea 42.

**Código MATLAB 3.1 cap3_DiscriminacionLineal.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 3. Discriminación lineal y regresión logística.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap3_DiscriminacionLineal.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
format short % formato corto con 4 dígitos después del punto decimal.
 % conjuntos de datos para las clases C^1 y C^2.
x1=[2; 3]; x2=[3; 5]; x3=[5; 6];  % vectores de la clase C^1.
C1x=[x1(1), x2(1), x3(1)]; % eje de abscisas en la figura 3.1a.
C1y=[x1(2), x2(2), x3(2)]; % eje de las ordenadas en la figura 3.1a.
x4=[5; 2]; x5=[7; 4]; x6=[8; 6];  % vectores de la clase C^2.
x=[x1, x2, x3, x4, x5, x6]; % se registran vectores de C^1 y C^2.
C2x=[x4(1), x5(1), x6(1)]; % eje de abscisas en la figura 3.1b.
C2y=[x4(2), x5(2), x6(2)]; % eje de las ordenadas en la figura 3.1b.
n=2; p=2; nT=6; % número de rasgos, clases y muestras total, respectivamente.
nC1=3; nC2=3; % número de muestras de las clases C^1 y C^2, respectivamente.
 % Figura 3.1a.
plot(C1x, C1y, 'ko' , C2x, C2y, 'k+' , 'MarkerFaceColor' ,[0,0,0])
axis([0, max([C1x C2x])+1, 0, max([C1y C2y])+1]); % límites de los ejes.
grid on % activa la retícula o malla de representación.
p_C1=nC1/nT;
p_C2=nC2/nT;
mu_C1=(1/nC1)*(x1+x2+x3); % vector μ̂_{C^1}.
mu_C2=(1/nC2)*(x4+x5+x6); % vector μ̂_{C^2}.
mu=[mu_C1, mu_C2];
Sigma_e=zeros(2,2); % inicializa la matriz de covarianza Σ̂.
```

**Continúa código 3.1: cap3_DiscriminacionLineal.m**

```
for clase=1:2 % algoritmo que obtienen la matriz de covarianza Σ̂.
    for k=1:3
        Sigma_e=Sigma_e+(x(1:2, k+3*(clase-1))-mu(1:2, clase))*...
                (x(1:2, k+3*(clase-1))-mu(1:2, clase))'; % ec. (3.6).
    end
end
Sigma_e=(1/(nT-p))*Sigma_e % cálculo de la matriz de covarianza Σ̂.
inv(Sigma_e) % matriz de covarianza inversa Σ̂^-1.
syms x1 x2 real % variables para programación simbólica.
digits(5) % formato numérico a 5 dígitos para programación simbólica.
fdC1=[x1, x2]*inv(Sigma_e)*mu_C1-(1/2)*mu_C1'*...
        inv(Sigma_e)*mu_C1+log(p_C1);
simplify(collect(vpa(fdC1))) % solución simbólica de fd_C1(x).
fdC2=[x1, x2]*inv(Sigma_e)*mu_C2-(1/2)*mu_C2'*...
        inv(Sigma_e)*mu_C2+log(p_C2);
simplify(collect(vpa(fdC2))) % solución simbólica para fd_C2(x).
figure % figura 3.1b.
plot(C1x, C1y,'ko' , C2x, C2y,'k+' , 'MarkerFaceColor' ,[0,0,0])
hold on
x1=6; x2 = 1.2076*x1 -1.7048;
x = [0 x1]; y = [-1.7048, x2]; plot(x,y,'--r' )
x=4; y=4;
plot(x,y,'ko' )
x=6; y=4;
plot(x,y,'ks' )
axis([0, max([C1x C2x])+1, -1.7048, max([C1y C2y])+1]); grid on
```

**• • • Ejemplo 3.2**

Considere el siguiente conjunto de datos mostrado en la figura 3.2a:

$$\left\{\boldsymbol{x}_1 = \begin{bmatrix}2\\3\end{bmatrix},\ \boldsymbol{x}_2 = \begin{bmatrix}2\\5\end{bmatrix}\right\} \in C^1$$
$$\left\{\boldsymbol{x}_3 = \begin{bmatrix}5\\8\end{bmatrix},\ \boldsymbol{x}_4 = \begin{bmatrix}7\\8\end{bmatrix}\right\} \in C^2$$
$$\left\{\boldsymbol{x}_5 = \begin{bmatrix}9\\4\end{bmatrix},\ \boldsymbol{x}_6 = \begin{bmatrix}9\\6\end{bmatrix}\right\} \in C^3$$

Para este ejemplo, dos vectores pertenecen a la clase $C^1$, dos vectores tienen pertenencia a la clase $C^2$ y dos se encuentran en la clase $C^3$.

Hacer lo siguiente:

a) Encontrar los estimados de las funciones discriminantes: $fd_{C^1}(\boldsymbol{x}), fd_{C^2}(\boldsymbol{x})$ y $fd_{C^3}(\boldsymbol{x})$

b) Encontrar las líneas de separación decisión entre pares de clases, y

c) Determinar la clase de los puntos: $\begin{bmatrix}5\\3\end{bmatrix}$, el cual está representado por un cuadro negro ■; $\begin{bmatrix}7\\7\end{bmatrix}$ se encuentra descrito por un cuadro blanco □; y denotado por un pentágono ⬠ a: $\begin{bmatrix}7\\4\end{bmatrix}$.

**Solución**

El número de rasgos, clases y total de muestras son $n = 2$, $p = 3$ y $n_T = 6$, respectivamente. Mientras que el número de muestras para la clase $C^1$ es $n_{C_1} = 2$, el número de muestras para la clase $C^2$ es $n_{C_2} = 2$ y el número de muestras para la clase $C^3$ es $n_{C_3} = 2$.

En la figura 3.2a se presenta el conjunto de datos de las clases $C^1, C^2, C^3$ a analizar. El desarrollo numérico realizado a mano es el siguiente:

a) Se obtiene la estimación de probabilidad de cada clase de datos:

$$\hat{p}_{C^1} = \frac{n_{C^1}}{n_T} = \frac{2}{6} = \frac{1}{3};\ \hat{p}_{C^2} = \frac{n_{C^2}}{n_T} = \frac{2}{6} = \frac{1}{3};\ \hat{p}_{C^3} = \frac{n_{C^3}}{n_T} = \frac{2}{6} = \frac{1}{3}.$$

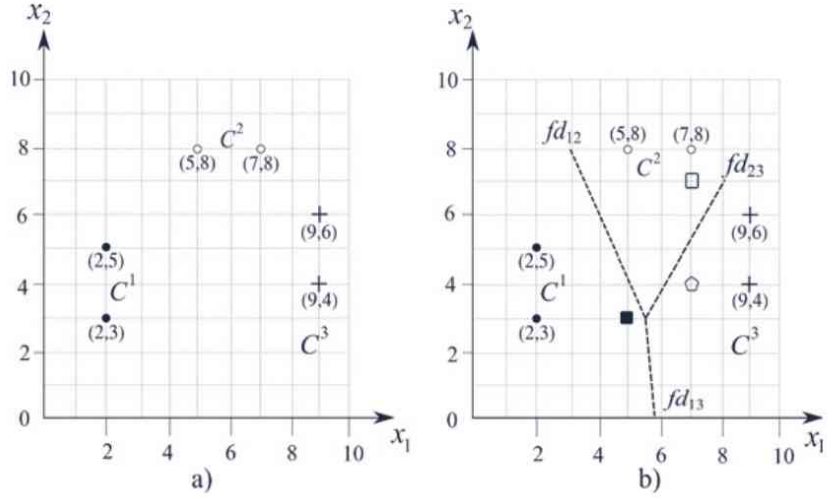



**Figura 3.2**: a) Conjunto de datos, funciones de separación entre los pares de clases y posiciones de los dos patrones a clasificar, b).

$$\hat{\boldsymbol{\mu}}_{C^1} = \frac{1}{n_{C^1}} \sum_{\text{vectores de clase } C^1} \begin{bmatrix} x_{C^1,1} \\ x_{C^1,2} \end{bmatrix} = \tfrac{1}{2}(\boldsymbol{x}_1 + \boldsymbol{x}_2) = \tfrac{1}{2}\left[\begin{bmatrix}2\\3\end{bmatrix} + \begin{bmatrix}2\\5\end{bmatrix}\right] = \begin{bmatrix}\frac{4}{2}\\\frac{8}{2}\end{bmatrix} = \begin{bmatrix}2\\4\end{bmatrix}$$

$$\hat{\boldsymbol{\mu}}_{C^2} = \frac{1}{n_{C^2}} \sum_{\text{vectores de clase } C^2} \begin{bmatrix} x_{C^1,2} \\ x_{C^1,2} \end{bmatrix} = \tfrac{1}{2}(\boldsymbol{x}_3 + \boldsymbol{x}_4) = \tfrac{1}{2}\left[\begin{bmatrix}5\\8\end{bmatrix} + \begin{bmatrix}7\\8\end{bmatrix}\right] = \begin{bmatrix}\frac{12}{2}\\\frac{16}{2}\end{bmatrix} = \begin{bmatrix}6\\8\end{bmatrix}$$

$$\hat{\boldsymbol{\mu}}_{C^3} = \frac{1}{n_{C^3}} \sum_{\text{vectores de clase } C^3} \begin{bmatrix} x_{C^3,1} \\ x_{C^3,2} \end{bmatrix} = \tfrac{1}{2}(\boldsymbol{x}_5 + \boldsymbol{x}_6) = \tfrac{1}{2}\left[\begin{bmatrix}9\\4\end{bmatrix} + \begin{bmatrix}9\\6\end{bmatrix}\right] = \begin{bmatrix}\frac{18}{2}\\\frac{10}{2}\end{bmatrix} = \begin{bmatrix}9\\5\end{bmatrix}$$

$$\hat{\sum} = \frac{1}{n_T - p} \sum_{k=1}^{p} \sum_{\text{vectores } \boldsymbol{x} \text{ de clase } C^k} (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{C^k})(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{C^k})^T$$

$$= \frac{1}{6-3}\left\{\left[\left[\begin{bmatrix}2\\3\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right]\left[\begin{bmatrix}2\\3\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right]^T + \left[\begin{bmatrix}2\\5\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right]\left[\begin{bmatrix}2\\5\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right]^T \right.\right.$$
$$+ \left[\begin{bmatrix}5\\8\end{bmatrix} - \begin{bmatrix}6\\8\end{bmatrix}\right]\left[\begin{bmatrix}5\\8\end{bmatrix} - \begin{bmatrix}6\\8\end{bmatrix}\right]^T + \left[\begin{bmatrix}7\\8\end{bmatrix} - \begin{bmatrix}6\\8\end{bmatrix}\right]\left[\begin{bmatrix}7\\8\end{bmatrix} - \begin{bmatrix}6\\8\end{bmatrix}\right]^T$$
$$\left.\left. + \left[\begin{bmatrix}9\\4\end{bmatrix} - \begin{bmatrix}9\\5\end{bmatrix}\right]\left[\begin{bmatrix}9\\4\end{bmatrix} - \begin{bmatrix}9\\5\end{bmatrix}\right]^T + \left[\begin{bmatrix}9\\6\end{bmatrix} - \begin{bmatrix}9\\5\end{bmatrix}\right]\left[\begin{bmatrix}9\\6\end{bmatrix} - \begin{bmatrix}9\\5\end{bmatrix}\right]^T \right\}$$

$$= \tfrac{1}{6-3}\left\{\begin{bmatrix}0\\-1\end{bmatrix}\begin{bmatrix}0 & -1\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix} + \begin{bmatrix}-1\\0\end{bmatrix}\begin{bmatrix}-1 & 0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}\right.$$
$$\left. + \begin{bmatrix}0\\-1\end{bmatrix}\begin{bmatrix}0 & -1\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}\right\}$$

$$= \frac{1}{6-3}\left\{\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}+\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}+\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\right\}$$
$$= \frac{1}{3}\begin{bmatrix}2 & 0\\ 0 & 4\end{bmatrix} = \begin{bmatrix}\frac{2}{3} & 0\\ 0 & \frac{4}{3}\end{bmatrix}.$$

De donde:

$$\hat{\sum}^{-1} = \begin{bmatrix}1.5 & 0\\ 0 & 0.75\end{bmatrix}$$

$$fd_{C^1}(\boldsymbol{x}) = \boldsymbol{x}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^1} - \frac{1}{2}\hat{\boldsymbol{\mu}}_{C^1}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^1} + \log\mathcal{P}(y=C^1)$$
$$= \begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}1.5 & 0\\ 0 & 0.75\end{bmatrix}\begin{bmatrix}2\\ 4\end{bmatrix} - \tfrac{1}{2}\begin{bmatrix}2 & 4\end{bmatrix}\begin{bmatrix}1.5 & 0\\ 0 & 0.75\end{bmatrix}\begin{bmatrix}2\\ 4\end{bmatrix} + \log(0.3)$$
$$= \begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}3\\ 3\end{bmatrix} - 9 - 1.204 = 3x_1 + 3x_2 - 10.0993$$

$$fd_{C^2}(\boldsymbol{x}) = \boldsymbol{x}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^2} - \frac{1}{2}\hat{\boldsymbol{\mu}}_{C^2}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^2} + \log\mathcal{P}(y=C^2)$$
$$= \begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}1.5 & 0\\ 0 & 0.75\end{bmatrix}\begin{bmatrix}6\\ 8\end{bmatrix} - \tfrac{1}{2}\begin{bmatrix}6 & 8\end{bmatrix}\begin{bmatrix}1.5 & 0\\ 0 & 0.75\end{bmatrix}\begin{bmatrix}6\\ 8\end{bmatrix} + \log(0.3)$$
$$= \begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}9\\ 6\end{bmatrix} - 51 - 1.204 = 9x_1 + 6x_2 - 52.099$$

$$fd_{C^3}(\boldsymbol{x}) = \boldsymbol{x}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^3} - \frac{1}{2}\hat{\boldsymbol{\mu}}_{C^3}^T\Sigma^{-1}\hat{\boldsymbol{\mu}}_{C^3} + \log\mathcal{P}(y=C^3)$$
$$= \begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}1.5 & 0\\ 0 & 0.75\end{bmatrix}\begin{bmatrix}9\\ 5\end{bmatrix} - \tfrac{1}{2}\begin{bmatrix}9 & 5\end{bmatrix}\begin{bmatrix}1.5 & 0\\ 0 & 0.75\end{bmatrix}\begin{bmatrix}9\\ 5\end{bmatrix} + \log(0.3)$$
$$= \begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}13.5\\ 3.75\end{bmatrix} - 70.125 - 1.204 = 13.5x_1 + 3.75x_2 - 71.224.$$

b) Las fronteras de separación o de decisión entre clases se obtienen como ya vimos, primeramente, al restar dos a dos las tres funciones previamente obtenidas:

$$fd_{C^2}(\boldsymbol{x}) - fd_{C^1}(\boldsymbol{x}) = 9x_1 + 6x_2 - 52.204 - (3x_1 + 3x_2 - 10.204)$$
$$= 6x_1 + 3x_2 - 42.$$

$$fd_{C^3}(\boldsymbol{x}) - fd_{C^1}(\boldsymbol{x}) = 13.5x_1 + 3.75x_2 - 71.329 - (3x_1 + 3x_2 - 10.204)$$
$$= 10.5x_1 + 0.75x_2 - 61.125.$$

$$fd_{C^3}(\boldsymbol{x}) - fd_{C^2}(\boldsymbol{x}) = 13.5x_1 + 3.75x_2 - 71.329 - (9x_1 + 6x_2 - 52.204).$$
$$= 4.5x_1 - 2.25x_2 - 19.125.$$

Enseguida, igualamos a cero cada una de las funciones obtenidas y en cada caso despejamos para la variable $x_2$ como sigue:

Frontera entre la clase $C^1$ y $C^2$ : $x_2 = -2x_1 + 14$.

Frontera entre la clase $C^3$ y $C^1$ : $x_2 = -14x_1 + 81.5$.

Frontera entre la clase $C^2$ y $C^3$ : $x_2 = 2x_1 - 8.5$.

En la figura 3.2b se muestran las tres fronteras de separación. Note cómo cada frontera pasa entre pares de clases.

c) La pertenencia de un patrón $\boldsymbol{x}$ a una de las tres clases $C^1$, $C^2$ y $C^3$ queda determinada por la combinación de signos del resultado al sustituir dicho patrón en las ecuaciones de las fronteras de decisión obtenidas en el paso b).

Como se puede ver de la figura 3.2b, la región de la clase $C^1$ queda delimitada por las fronteras de decisión $fd_{12}$ y $fd_{13}$, la región para la clase $C^2$ queda delimitada por las fronteras de decisión $fd_{12}$ y $fd_{23}$, mientras que la región para la clase $C^3$ queda delimitada por las fronteras de decisión $fd_{13}$ y $fd_{23}$.

Para este problema se tienen por tanto, tres casos:

1) Si al sustituir el patrón $\boldsymbol{x}$ los signos obtenidos para las fronteras de decisión $fd_{12}$ y $fd_{13}$ son negativos "-", el patrón es clasificado en la región de la clase $C^1$.

2) Si al sustituir el patrón $\boldsymbol{x}$ los signos obtenidos para las fronteras de decisión $fd_{12}$ y $fd_{23}$ son "+" y "-", respectivamente, el patrón es clasificado en la región de la clase $C^2$.

3) Si al sustituir el patrón $\boldsymbol{x}$ los signos obtenidos para las fronteras de decisión $fd_{13}$ y $fd_{23}$ son ambos positivos "+", el patrón es clasificado en la región de la clase $C^3$.

Al sustituir $\boldsymbol{x} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$ en las fronteras de decisión $fd_{12}$ y $fd_{13}$:

$$fd_{C^3}(\boldsymbol{x}) - fd_{C^1}(\boldsymbol{x}) = fd_{13} = 10.5x_1 + 0.75x_2 - 61.125 = 52.5 + 1.5 - 61.125 = -7.125.$$

$$fd_{C^2}(\boldsymbol{x}) - fd_{C^1}(\boldsymbol{x}) = fd_{12} = 6x_1 + 3x_2 - 42 = 30 + 9 - 42 = -3.$$

Funciones de separación entre los pares de clases y posiciones de los dos patrones a clasificar, el primero (5,3) representado por un cuadrado negro; el segundo con coordenadas (7,7) indicado por un cuadrado blanco y el tercero (7,4) con un pentágono ⬠. Como los signos de $fd_{12}$ y $fd_{13}$ son ambos negativos, el patrón $\boldsymbol{x} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$ es puesto en la clase $C^1$. Al observar la figura 3.2b note que efectivamente dicho punto se encuentra dentro de la clase correspondiente a la clase $C^1$.

Para el patrón $\boldsymbol{x} = \begin{bmatrix} 7 \\ 7 \end{bmatrix}$, al sustituir en las fronteras de decisión $fd_{12}$ y $fd_{23}$:

$$fd_{C^2}(\boldsymbol{x}) - fd_{C^1}(\boldsymbol{x}) = fd_{12} = 6x_1 + 3x_2 - 42 = 42 + 21 - 42 = 21.$$

$$\begin{aligned} fd_{C^3}(\boldsymbol{x}) - fd_{C^2}(\boldsymbol{x}) &= fd_{23} = 4.5x_1 - 2.25x_2 - 19.125 = 31.5 - 15.75 - 19.125 \\ &= 3.375. \end{aligned}$$

Debido a que los signos de $fd_{12}$ y $fd_{23}$ son "-" y "+", respectivamente, el patrón $\boldsymbol{x} = \begin{bmatrix} 7 \\ 7 \end{bmatrix}$ es puesto en la clase $C^2$. Al observar la figura 3.2b note que efectivamente dicho punto se encuentra dentro de la clase correspondiente a la clase $C^2$.

Finalmente, para el patrón $\boldsymbol{x} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$, al sustituir en las fronteras de decisión $fd_{12}$ y $fd_{23}$:

$$fd_{C^2}(\boldsymbol{x}) - fd_{C^1}(\boldsymbol{x}) = fd_{12} = 6x_1 + 3x_2 - 42 = 42 + 12 - 42 = 12.$$

$$\begin{aligned} fd_{C^3}(\boldsymbol{x}) - fd_{C^2}(\boldsymbol{x}) &= fd_{23} = 4.5x_1 - 2.25x_2 - 19.125 = 31.5 - 9 - 19.125 \\ &= 3.375. \end{aligned}$$

Como los signos de $fd_{12}$ y $fd_{23}$ son ambos positivos, el patrón $\boldsymbol{x} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$ es puesto en la clase $C^3$. Al observar la figura 3.2b note que efectivamente dicho punto se encuentra dentro de la clase correspondiente a la clase $C^3$.

El código fuente escrito en **MATLAB** que reproduce el desarrollo numérico de este ejemplo se encuentra en el programa cap3_DiscriminacionLineal_A.m (ver cuadro de código 3.2), cuya implementación es similar al cuadro de programación en **MATLAB** 3.1 del ejemplo 3.1.

**Código MATLAB 3.2 cap3_DiscriminacionLineal_A.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 3. Discriminación lineal y regresión logística.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Programa: cap3_DiscriminacionLineal_A.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
format short % formato corto con 4 dígitos después del punto decimal.
 % conjuntos de datos para las clases C^1, C^2 y C^3.
x1=[2; 3]; x2=[2; 5];  % vectores de la clase C^1.
x3=[5; 8]; x4=[7; 8]; % vectores de la clase C^2.
x5=[9; 4]; x6=[9; 6]; % vectores de la clase C^3.
Cx=[x1(1), x2(1), x3(1), x4(1), x5(1), x6(1)]; % eje de abscisas en la figura 3.2a.
Cy=[x1(2), x2(2), x3(2), x4(2), x5(2), x6(2)]; % eje de las ordenadas en la figura 3.2a.
x=[x1, x2, x3, x4, x5, x6];
n=2; p=3; nT=6; % número de rasgos, clases y muestras total, respectivamente.
nC1=2; nC2=2; nC3=2 % número de muestras de las clases C^1, C^2 y C^3, respectivamente.
 % Figura 3.2a.
plot(x1(1),x1(2),'ko' , x2(1), x2(2),'ko' , 'MarkerFaceColor' ,[0,0,0])
hold on
plot(x3(1), x3(2),  'ko' , x4(1), x4(2),'ko' )
plot(x5(1), x5(2),  'k+' , x6(1), x6(2),'k+' )
axis([0, max(Cx)+1, 0, max(Cy)+1]); % límites de los ejes.
grid on % activa la retícula o malla para representar.
 p_C1=nC1/nT; p_C2=nC2/nT; p_C3=nC3/nT;
 mu_C1=(1/nC1)*(x1+x2); % vector μ̂_{C^1}.
 mu_C2=(1/nC2)*(x3+x4); % vector μ̂_{C^2}.
 mu_C3=(1/nC3)*(x5+x6); % vector μ̂_{C^3}.
```

**Continúa código 3.2: cap3_DiscriminacionLineal_A.m**

```
mu=[mu_C1, mu_C2, mu_C3];
Sigma_e=zeros(2,2) % inicializa la matriz de covarianza Σ̂.
for clase=1:3
    for k=1:2
        Sigma_e=Sigma_e+(x(1:2, k+2*(clase-1))-mu(1:2, clase))*...
                (x(1:2, k+2*(clase-1))-mu(1:2, clase))';% ec. (3.6).
    end
end
Sigma_e=(1/(nT-p))*Sigma_e % cálculo de la matriz de covarianza Σ̂.
inv(Sigma_e) % matriz de covarianza inversa Σ̂^-1.
figure % figura 3.2b.
plot(x1(1),x1(2), 'ko' , x2(1), x2(2), 'ko' , 'MarkerFaceColor' ,[0,0,0])
hold on
plot(x3(1), x3(2), 'ko' , x4(1), x4(2), 'ko' )
plot(x5(1), x5(2), 'k+' , x6(1), x6(2), 'k+' )
axis([0, max(Cx)+1, 0, max(Cy)+1]);
grid on
syms x1 x2 real
digits(5)
fdC1=[x1, x2]*inv(Sigma_e)*mu_C1-(1/2)*mu_C1'*...
        inv(Sigma_e)*mu_C1+log(p_C1);
% versión simbólica de la ecuación: fd_C1(x) = 3x1 + 3x2 − 10.0993.
simplify(collect(vpa(fdC1)))
fdC2=[x1, x2]*inv(Sigma_e)*mu_C2-(1/2)*mu_C2'*...
         inv(Sigma_e)*mu_C2+log(p_C2);
% versión simbólica de la ecuación: fd_C2(x) = 9x1 + 6x2 − 52.099.
simplify(collect(vpa(fdC2)))
fdC3=[x1, x2]*inv(Sigma_e)*mu_C3-(1/2)*mu_C3'*...
        inv(Sigma_e)*mu_C3+log(p_C3);
simplify(collect(vpa(fdC3))) %versión simbólica de: fd_C3(x) = 13.5x1 + 3.75x2 − 71.224.
```

**Continúa código 3.2: cap3_DiscriminacionLineal_A.m**

```
55 x_1=3; x2 =-2*x_1+14; x = [x_1, 5.5]; y = [8, 3]; plot(x,y, '--k' )
56 x1=5.5; x2 =-14*x1+81.5; x = [5.5 5.8]; y = [3, 0]; plot(x,y, '--k' )
57 x1=2; x2 = 2*x1 -8.5; x = [5.5 8]; y = [3, 7]; plot(x,y, '--k' )
58 % tipos de clases:
59 plot(5,3, 'bs' , 'MarkerFaceColor' ,[0,0,0]) % representado por un cuadro ■.
60 plot(7,7, 'bs' ) % descrito por un cuadro blanco □.
61 plot(7,4, 'bP' ) % denotado por un pentágono ⬠.
```

## 3.3 Regresión logística

En esta sección se estudia el problema de la conocida regresión logística. Al igual que en el caso del análisis de discriminación lineal, se desea estimar la probabilidad $\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})$ con el fin de elegir la clase $C^k$ cuya probabilidad sea de nuevo la más alta. En esta sección en lugar de estimar dicha probabilidad de manera indirecta mediante el teorema de Bayes, se hará de manera directa.

Sin pérdida de generalidad, se tratará el problema bi-clase, esto es el problema para el cual $p = 2$. Dichas clases $C^1$ y $C^2$ serán denotadas por las etiquetas 0 y 1. Para estimar la probabilidad $\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})$ se aplican las siguientes tres fases:

1. **Modelado de regresión logística de la función de probabilidad a posteriori.** Se supone que el modelo de regresión logística es lineal. Para esto se considera el siguiente cociente:

$$\frac{\mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x})}{(1 - \mathcal{P}(y = C^k|\boldsymbol{\chi} = \boldsymbol{x}))}.$$

Suponga ahora el logaritmo de este cociente, esto es:

$$\ln\left(\frac{\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x})}{(1-\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x}))}\right).$$

Asuma, como es costumbre en regresión logística, que este logaritmo es una función lineal de los componentes de $\boldsymbol{x}$, eso es:

$$\ln\left(\frac{\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x})}{(1-\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x}))}\right)=\beta_0+\beta_1x_1+\cdots+\beta_nx_n$$

Resolviendo para $\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x})$:

$$\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x})=\frac{e^{\beta_0+\beta_1x_1+\cdots+\beta_nx_n}}{1+e^{\beta_0+\beta_1x_1+\cdots+\beta_nx_n}}. \tag{3.7}$$

En resumen, para estimar la probabilidad $\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x})$ es menester primero estimar el vector de parámetros $[\,\beta_0,\beta_1,\cdots,\beta_n\,]^T$.

2. **Estimado de la función de probabilidad a posteriori**. Para estimar esta función de probabilidad, como es costumbre, se parte de un conjunto de datos. Sea este conjunto de datos: $(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_p,y_p)$. En lo que sigue se supondrá que los valores de las variables $y_1,\cdots,y_p$ son 0 o 1. La probabilidad de los datos viene dada por el producto de las probabilidades de que $y=1$ para aquellos $\boldsymbol{x}_i$ sea 1 y las probabilidades de que $y=0$ para aquellos $\boldsymbol{x}_i$ sea 0, esto es:

$$\prod_{i\text{ para los cuales }y_i=1}\mathcal{P}(y=1|\boldsymbol{\chi}=\boldsymbol{x}_i)\prod_{i\text{ para los cuales }y_i=0}\mathcal{P}(y=0|\boldsymbol{\chi}=\boldsymbol{x}_i)$$

Debido a que $\mathcal{P}(y=0|\boldsymbol{\chi}=\boldsymbol{x}_i)=1-\mathcal{P}(y=1|\boldsymbol{\chi}=\boldsymbol{x}_i)$:

$$\prod_{i\text{ para los cuales }y_i=1}\mathcal{P}(y=1|\boldsymbol{\chi}=\boldsymbol{x}_i)\prod_{i\text{ para los cuales }y_i=0}\mathcal{P}(y=0|\boldsymbol{\chi}=\boldsymbol{x}_i)=\prod_{i\text{ para los cuales }y_i=1}\mathcal{P}(\boldsymbol{x}_i)\prod_{i\text{ para los cuales }y_i=0}(1-\mathcal{P}(\boldsymbol{x}_i)). \tag{3.8}$$

Para poder estimar la función de probabilidad a posteriori $\mathcal{P}(y=C^k|\boldsymbol{\chi}=\boldsymbol{x})$, ya vimos que es necesario encontrar el vector de parámetros:

$$\boldsymbol{\beta}=\begin{bmatrix}\beta_0\\ \beta_1\\ \vdots\\ \beta_n\end{bmatrix}$$

esto es, encontrar el vector $\boldsymbol{\beta}$ que maximice el producto dado por la ecuación (2.20).

A esta nueva función, que depende ahora del vector de parámetros $\boldsymbol{\beta}$, se le conoce en la literatura como función de verosimilitud [8]. Para estimar el vector $\boldsymbol{\beta}$ es menester maximizar entonces esta función de verosimilitud:

$$f\boldsymbol{\nu}(\boldsymbol{\beta}) = \prod_{i \text{ para los cuales } y_i = 1} \mathcal{P}(\boldsymbol{x}_i) \prod_{i \text{ para los cuales } y_i = 0} (1 - \mathcal{P}(\boldsymbol{x}_i)).$$

Es bien sabido que el maximizar $f\boldsymbol{\nu}(\boldsymbol{\beta})$ es equivalente a maximizar $\ln(\boldsymbol{\nu}(\boldsymbol{\beta})$. Esta nueva función es la llamada función del logaritmo de verosimilitud [8]. Note que:

$$\begin{aligned} \ln(f\boldsymbol{\nu}(\boldsymbol{\beta})) &= \sum_{i \text{ para los cuales } y_i = 1} \ln(\mathcal{P}(\boldsymbol{x}_i, \boldsymbol{\beta})) + \sum_{i \text{ para los cuales } y_i = 0} \ln(1 - \mathcal{P}(\boldsymbol{x}_i, \boldsymbol{\beta})) \\ &= \sum_{i=1}^{p} [y_i \ln(\mathcal{P}(\boldsymbol{x}_i, \boldsymbol{\beta})) + (1 - y_i) \ln(1 - \mathcal{P}(\boldsymbol{x}_i, \boldsymbol{\beta}))] \\ &= \sum_{i=1}^{p} \left[ y_i \boldsymbol{\beta}^T \boldsymbol{x}_{ia} - \ln(1 - e^{\boldsymbol{\beta}^T \boldsymbol{x}_{ia}}) \right] \end{aligned} \tag{3.9}$$

Con $\boldsymbol{x}_{ia} = \begin{bmatrix} 1 \\ x_{ia1} \\ \vdots \\ x_{ian} \end{bmatrix}$, como vector aumentado.

3. **Maximización de la función logaritmo de verosimilitud**. Para maximizar esta función usualmente se emplea el método de Newton-Raphson multivariable [9]. Solo para recordar, la aplicación de este método supone el uso de los siguientes tres pasos:

   3.1 Escoger un valor inicial para el vector de parámetros. Sea este valor $\boldsymbol{\beta}(0)$.

   3.2 Actualizar el valor del vector $\boldsymbol{\beta}(k)$ como sigue: $\boldsymbol{\beta}(k+1) = \boldsymbol{\beta}(k) - H^{-1} \bigtriangledown \ln(\boldsymbol{\beta}(k))$, donde como es bien sabido $H = \bigtriangledown^2 \ln(\boldsymbol{\beta}(k))$.

   3.3 El $\ln(\boldsymbol{\beta}(k))$ alcanza un máximo para el valor del vector $\boldsymbol{\beta}$ al cual la secuencia de valores $\boldsymbol{\beta}(0), \boldsymbol{\beta}(1), \cdots$ converge.

El lector puede demostrar que si:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$X = \begin{bmatrix} x_{10} & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p0} & x_{p1} & \cdots & x_{pn} \end{bmatrix}$$

$$\boldsymbol{p} = \begin{bmatrix} \mathcal{P}(\boldsymbol{x}_1, \boldsymbol{\beta}) \\ \vdots \\ \mathcal{P}(\boldsymbol{x}_p, \boldsymbol{\beta}) \end{bmatrix}$$

$$W = \begin{bmatrix} \mathcal{P}(\boldsymbol{x}_1, \boldsymbol{\beta})\,[\,1 - \mathcal{P}(\boldsymbol{x}_1, \boldsymbol{\beta})\,] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathcal{P}(\boldsymbol{x}_n, \boldsymbol{\beta})\,[\,1 - \mathcal{P}(\boldsymbol{x}_n, \boldsymbol{\beta})\,] \end{bmatrix}$$

entonces, el gradiente y el hessiano de la función $\ln(\boldsymbol{\beta})$ vienen dados, respectivamente como:

$$\begin{aligned} \bigtriangledown \ln(\boldsymbol{\beta}) &= \sum_{i=1}^{p} \boldsymbol{x}_{ia}\,[\,y_i - \mathcal{P}(\boldsymbol{x}_i, \boldsymbol{\beta})\,] = X^T(\boldsymbol{y} - \boldsymbol{p}) \\ H &= \bigtriangledown^2 \ln(\boldsymbol{\beta}) = -\sum_{i=1}^{p} \boldsymbol{x}_{ia}\ \boldsymbol{x}_{ia}^T \mathcal{P}(\boldsymbol{x}_i, \boldsymbol{\beta})\,[\,1 - \mathcal{P}(\boldsymbol{x}_i, \boldsymbol{\beta})\,] \\ &= -X^T W X. \end{aligned}$$

Al sustituir estas dos expresiones en el segundo paso del método de Newton-Raphson:

$$\begin{aligned} \boldsymbol{\beta}(k+1) &= \boldsymbol{\beta}(k) - H^{-1} \bigtriangledown \ln(\boldsymbol{\beta}(k)) \\ &= \boldsymbol{\beta}(k) - [\,-X^T W X\,]^{-1} X^T\,[\,\boldsymbol{y} - \boldsymbol{p}\,] \\ &= \boldsymbol{\beta}(k) + [\,X^T W X\,]^{-1} X^T\,[\,\boldsymbol{y} - \boldsymbol{p}\,] \end{aligned} \tag{3.10}$$

### • • • Ejemplo 3.3

Considere el siguiente conjunto de datos mostrado en la figura 3.3a:

$$\left\{\left(\boldsymbol{x}_1=\begin{bmatrix}2\\3\end{bmatrix},\ y_1=0\right),\ \left(\boldsymbol{x}_2=\begin{bmatrix}5\\2\end{bmatrix},\ y_2=0\right),\ \left(\boldsymbol{x}_3=\begin{bmatrix}2\\5\end{bmatrix},\ y_3=0\right)\right\}\in C^1$$

$$\left\{\left(\boldsymbol{x}_4=\begin{bmatrix}5\\8\end{bmatrix},\ y_4=1\right),\ \left(\boldsymbol{x}_5=\begin{bmatrix}7\\8\end{bmatrix},\ y_5=1\right),\ \left(\boldsymbol{x}_6=\begin{bmatrix}7\\6\end{bmatrix},\ y_6=1\right)\right\}\in C^2.$$

Aplicar el método de regresión logística para estimar la función de probabilidad y clasificar el punto con coordenadas (2,9), representado como un círculo vacío "○", mostrado en la figura 3.3a, con base en esta probabilidad estimada.

### Solución

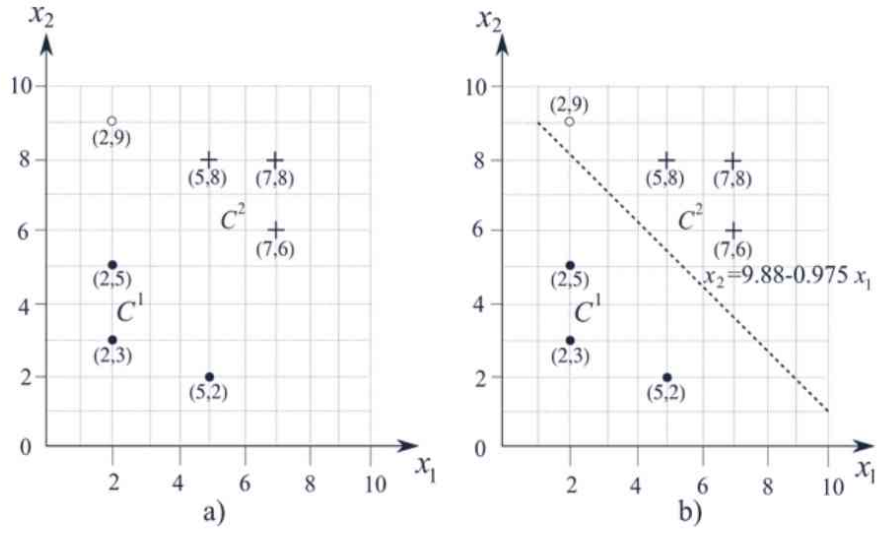



**Figura 3.3**: Conjunto de datos, figura a) y función de separación $fd(\boldsymbol{x})$ entre las dos clases $C^1=0$ y $C^2=1$, figura b).

De acuerdo con la ecuación (3.10), primero que todo encontramos los valores de la matrices $X$, $W$ y vectores: $\boldsymbol{y},\boldsymbol{p}$, como sigue:

$$X = \begin{bmatrix}1&2&3\\1&5&2\\1&2&5\\1&5&8\\1&7&8\\1&7&6\end{bmatrix},\quad \boldsymbol{y}=\begin{bmatrix}0\\0\\0\\1\\1\\1\end{bmatrix},\quad \boldsymbol{p}=\begin{bmatrix}\mathcal{P}(\boldsymbol{x}_1,\boldsymbol{\beta})\\\mathcal{P}(\boldsymbol{x}_2,\boldsymbol{\beta})\\\vdots\\\mathcal{P}(\boldsymbol{x}_6,\boldsymbol{\beta})\end{bmatrix}$$

$$W = \begin{bmatrix} \mathcal{P}(\boldsymbol{x}_1,\boldsymbol{\beta})\,[1-\mathcal{P}(\boldsymbol{x}_1,\boldsymbol{\beta})] & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathcal{P}(\boldsymbol{x}_2,\boldsymbol{\beta})\,[1-\mathcal{P}(\boldsymbol{x}_2,\boldsymbol{\beta})] & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathcal{P}(\boldsymbol{x}_3,\boldsymbol{\beta})\,[1-\mathcal{P}(\boldsymbol{x}_3,\boldsymbol{\beta})] & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathcal{P}(\boldsymbol{x}_4,\boldsymbol{\beta})\,[1-\mathcal{P}(\boldsymbol{x}_4,\boldsymbol{\beta})] & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathcal{P}(\boldsymbol{x}_5,\boldsymbol{\beta})\,[1-\mathcal{P}(\boldsymbol{x}_5,\boldsymbol{\beta})] & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathcal{P}(\boldsymbol{x}_6,\boldsymbol{\beta})\,[1-\mathcal{P}(\boldsymbol{x}_6,\boldsymbol{\beta})] \end{bmatrix}.$$

Enseguida, sabiendo que en paso de iteración $k$-ésimo $\mathcal{P}(\boldsymbol{x}_{i_a},\boldsymbol{\beta}(k)) = \dfrac{e^{(\,\boldsymbol{\beta}^T(k)\boldsymbol{x}_{i_a}\,)}}{1+e^{(\,\boldsymbol{\beta}^T(k)\boldsymbol{x}_{i_a}\,)}}$ se escoge un valor inicial $\boldsymbol{\beta}(0)$ para el vector $\boldsymbol{\beta}(k)$. Supongamos que este valor es $\boldsymbol{\beta}(0)=\mathbf{0}$. En consecuencia:

$$\boldsymbol{p} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \in \mathbb{R}^{6\times 1},\ W = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} \end{bmatrix} \in \mathbb{R}^{6\times 6}.$$

Al aplicar el primer paso de iteración ($k=1$) a través de la ecuación (3.10 ):

$$\begin{aligned} \boldsymbol{\beta}(1) &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \left[ \begin{pmatrix} 3 & 2 & 5 & 8 & 8 & 6 \\ 2 & 5 & 2 & 5 & 7 & 7 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 1 & 5 & 2 \\ 1 & 2 & 5 \\ 1 & 5 & 8 \\ 1 & 7 & 8 \\ 1 & 7 & 6 \end{pmatrix} \right]^{-1} \begin{pmatrix} 3 & 2 & 5 & 8 & 8 & 6 \\ 2 & 5 & 2 & 5 & 7 & 7 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \left[ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \right] \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1.5 & 7 & 8 \\ 7 & 39 & 41 \\ 8 & 41 & 50.5 \end{bmatrix}^{-1} \begin{pmatrix} 3 & 2 & 5 & 8 & 8 & 6 \\ 2 & 5 & 2 & 5 & 7 & 7 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \left[ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \right] \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{pmatrix} 5.31797235023041 & -0.470046082949308 & -0.460829493087557 \\ -0.470046082949308 & 0.216589861751152 & -0.101382488479263 \\ -0.460829493087557 & -0.101382488479263 & 0.175115207373272 \end{pmatrix} \begin{bmatrix} 0 \\ 5 \\ 6 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{pmatrix} -5.11520737 \\ 0.47465438 \\ 0.5437788) \end{pmatrix} = \begin{pmatrix} -5.11520737 \\ 0.47465438 \\ 0.5437788) \end{pmatrix}. \end{aligned}$$

Al aplicar el segundo paso de iteración ($k = 2$):

$$\boldsymbol{p} = \begin{bmatrix} 0.0734704801492283 \\ 0.160518144043816 \\ 0.190462987822969 \\ 0.833174525407095 \\ 0.928082679254701 \\ 0.813064390312151 \end{bmatrix}, W = \begin{bmatrix} 0.068 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.1347 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.1541 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.1389 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.0667 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.1519 \end{bmatrix}, \boldsymbol{\beta}(2) = \begin{bmatrix} -8.4205248 \\ 0.80023996 \\ 0.87902665 \end{bmatrix}.$$

Al aplicar el tercer paso de iteración ($k = 3$):

$$\boldsymbol{p} = \begin{bmatrix} 0.01502405 \\ 0.06529767 \\ 0.08129268 \\ 0.93168645 \\ 0.98541929 \\ 0.92094925) \end{bmatrix}, W = \begin{bmatrix} 0.014 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.061 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.075 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.064 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.014 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.073 \end{bmatrix}, \boldsymbol{\beta}(3) = \begin{bmatrix} -11.65931515 \\ 1.12245563 \\ 1.20457181) \end{bmatrix}.$$

Si continuamos con este proceso iterativo hasta la décima iteración, por ejemplo, se puede demostrar que el vector de probabilidades estará tomando valores muy cercarnos a cero en sus primeros tres componentes y valores muy cercanos a uno en sus últimos tres componentes.

El vector de parámetros $\boldsymbol{\beta}$, por otro lado, habrá tomado los siguientes valores en la iteración 10: $\boldsymbol{\beta}(10) = [\, -34.78053313, 3.43230622, 3.51891718 \,]^T$.

Si detenemos el proceso en esta iteración, podremos representar la línea de separación correspondiente entre las dos clases.

$$fd(\boldsymbol{x}) = 3.43230622x_1 + 3.51891718x_2 - 34.78053313. \tag{3.11}$$

Al igualar a 0 y despejar para $x_2$, se tiene que:

$$x_2 = 9.8839 - 0.9754x_1.$$

En la figura 3.3b aparece representada esta línea de separación. Finalmente, procedamos a representar el punto con coordenadas (2,9), representado como un círculo vacío "○" y mostrado en las figuras 3.3a y 3.3b. Como ya vimos, para encontrar la clase de pertenencia de este punto, sustituyamos sus coordenadas en la ecuación (2.21):

$$fd(\boldsymbol{x}) = 3.43230622 \times 2 + 3.51891718 \times 9 - 34.78053313 = 3.7543.$$

Como el signo es positivo, de acuerdo con la teoría, el punto con coordenadas (2,9) se considera como perteneciente a la clase $C^2 = 1$, lo cual es coincidente con lo observado en la figura 3.3b, donde se ve que dicho punto se encuentra del lado de la clase $C^2$.

El procedimiento numérico realizado a mano para este ejemplo puede ser verificado a través del programa cap3 RegresionLogistica.m, cuya programación fuente en **MATLAB** y se encuentra documentado en el cuadro de código 3.3. El conjunto de datos para las clases $C^1$ y $C^2$ está declarado en las líneas de programación 6 y 7, respectivamente. Mientras que la formación de elementos del vector $\boldsymbol{y} \in \mathbb{R}^{6\times 1}$ se realiza en la línea 10, así como las entradas de la matrix $X \in \mathbb{R}^{6\times 6}$ en la línea 12.

Generalmente es común tomar como condiciones iniciales $\boldsymbol{\beta}(0) \in \mathbb{R}^{3\times 1}$ del vector $\boldsymbol{\beta}(k) \in \mathbb{R}^{3\times 1}$ al vector $\mathbf{0} \in \mathbb{R}^{3\times 1}$, es decir, todas sus componentes son cero (como se ilustra en la línea 13). Por otro lado, la figura 3.3a que indica los datos de las clases $C^1$ y $C^2$ se obtiene usando el código comprendido entre las líneas 14 a la 23.

El algoritmo para implementar la ecuación recursiva (3.10) se realiza entre las líneas 24 y 41. El lector puede ejecutar el programa cap3_RegresionLogistica.m y verificar el comportamiento numérico en cada $k$-ésima iteración de los vectores $\boldsymbol{p}$ y $\boldsymbol{\beta}(k)$ y de la matriz $W$. Con esta finalidad en la línea 40 se inserta la función pause, la cual detiene la ejecución del programa, manteniendo la exhibición de valores numéricos en la ventana de comandos de **MATLAB**; entonces el usuario puede presionar cualquier tecla para continuar con la siguiente iteración.

La figura 3.3b se obtiene a través del código contenido entres las líneas de programación 42 y 49. Observe que la figura 3.3b muestra la línea de separación entre las dos clases de datos $C^1$ y $C^2$.

También se puede ver que la función de probabilidad $\mathcal{P}(\boldsymbol{x}_{i_a}, \boldsymbol{\beta}(k)) = \frac{e^{(\boldsymbol{\beta}^T(k)\boldsymbol{x}_{i_a})}}{1 + e^{(\boldsymbol{\beta}^T(k)\boldsymbol{x}_{i_a})}}$ se encuentra implementada como parte del código del programa principal (ver línea 51). La función de probabilidad permite la actualización numérica en cada iteración del vector $\boldsymbol{\beta}(k)$ a través del vector $\boldsymbol{p}$ y de la matriz diagonal $W$.

A continuación se presenta en el cuadro de código 3.3 la descripción y documentación del programa cap3-RegresionLogistica.m con el código en **MATLAB** para resolver el planteamiento de este ejemplo.

**Código MATLAB 3.3 cap3_RegresionLogistica.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 3. Discriminación lineal y regresión logística.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Programa: cap3_RegresionLogistica.m MATLAB versión 2020b

```
clc; % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
format longG % formato largo.
 % vectores de las clases C^1 y C^2.
x1_a=[1;  2; 3]; x2_a=[1; 5; 2]; x3_a=[1; 2; 5]; % vectores de la clase C^1.
x4_a=[1; 5; 8]; x5_a=[1; 7; 8]; x6_a=[1; 7; 6];  % vectores de la clase C^2.
y1=0; y2=0; y3=0; y4=1; y5=1; y6=1; % componentes del vector y ∈ IR^{6×1}.
 % vector y ∈ IR^{6×1}.
y=[y1; y2; y3; y4; y5; y6];
 % matriz X ∈ IR^{6×6}.
X=[x1_a' ; x2_a' ; x3_a' ; x4_a' ; x5_a' ; x6_a' ];
beta_k=[0; 0; 0]; % condiciones iniciales del vector β(0) ∈ IR^{3×1}.
figure % figura 3.3a.
 % vectores de la clase C^1.
plot(x1_a(2),x1_a(3),'ko' , x2_a(2), x2_a(3), 'ko' , x3_a(2), x3_a(3),'ko' ,'MarkerFaceColor' ,[0,0,0])
hold on
 % vectores de la clase C^2.
plot(x4_a(2), x4_a(3),'k+' , x5_a(2), x5_a(3), 'k+' , x6_a(2), x6_a(3), 'k+' )
 % punto con coordenadas (2,9), representado como un círculo vacío ○
plot(2,9 ,'ko' )
axis([0, 10, 0, 10]);
grid on
```

**Continúa código 3.3: cap3_RegresionLogistica.m**

```
for k=1:10 % proceso recursivo del vector β(k+1) = β(k) + [X^T W X]^-1 X^T [y − p].
    p=[probab(x1_a, beta_k);
       probab(x2_a, beta_k);
       probab(x3_a, beta_k);
       probab(x4_a, beta_k);
       probab(x5_a, beta_k);
       probab(x6_a, beta_k)]; %  vector p ∈ IR^{6×1}.
    W=[probab(x1_a, beta_k)*(1-probab(x1_a, beta_k)), 0, 0,0,0,0;
        0, probab(x2_a, beta_k)*(1-probab(x2_a, beta_k)), 0, 0, 0, 0;
        0, 0, probab(x3_a, beta_k)*(1-probab(x3_a, beta_k)), 0,0,0;
        0,0,0, probab(x4_a, beta_k)*(1-probab(x4_a, beta_k)),0,0;
        0,0,0,0, probab(x5_a, beta_k)*(1-probab(x5_a, beta_k)),0;
        0,0,0,0,0,probab(x6_a, beta_k)*(1-probab(x6_a, beta_k))]; % W ∈ IR^{6×6}.
    k % despliega la k-esima iteración.
    beta_k=beta_k+ (X'*W*X)^(-1)*X '*(y-p) % ec. (3.10).
    disp( 'Para continuar, oprima cualquier tecla.' )
    pause
end
figure % figura 3.3b.
plot(x1_a(2),x1_a(3), 'ko' , x2_a(2), x2_a(3), 'ko' , x3_a(2), x3_a(3), 'ko' , 'MarkerFaceColor' [0,0,0])
hold on
plot(x4_a(2), x4_a(3), 'k+' , x5_a(2), x5_a(3), 'k+' , x6_a(2), x6_a(3), 'k+' )
plot(2,9 , 'ko' )
axis([0, 10, 0, 10]);
grid on
plot([1, 10], [9.88-0.975*(1), 9.88-0.975*(10)], '--k' )
% función probabilidad P(x_ia, β(k))
function probalidad=probab(xia, beta_k)
    probalidad=exp(beta_k'*xia)/(1+exp(beta_k'*xia)); %P(x_ia, β(k)) = e^(β^T(k) x_ia) / (1 + e^(β^T(k) x_ia)).
end
```

Se recuerda al lector que el código fuente en **MATLAB** de todos los programas que refuerzan a los ejemplos con solución se encuentran disponibles en el sitio Web de esta obra (previo registro).

---

A continuación se presenta otro ejemplo minuciosamente desarrollado para ilustrar la metodología de regresión logística.

### • • • Ejemplo 3.4

Considere el siguiente conjunto de datos mostrado en la figura 3.4a:

$$\left\{\left(\boldsymbol{x}_1=\begin{bmatrix}2\\5\end{bmatrix},\ y_1=0\right),\ \left(\boldsymbol{x}_2=\begin{bmatrix}4\\3\end{bmatrix},\ y_2=0\right),\ \left(\boldsymbol{x}_3=\begin{bmatrix}5\\6\end{bmatrix},\ y_3=0\right)\right\}\in C^1$$

$$\left\{\left(\boldsymbol{x}_4=\begin{bmatrix}5\\8\end{bmatrix},\ y_4=1\right),\ \left(\boldsymbol{x}_5=\begin{bmatrix}7\\6\end{bmatrix},\ y_5=1\right),\ \left(\boldsymbol{x}_6=\begin{bmatrix}6\\4\end{bmatrix},\ y_3=1\right)\right\}\in C^2.$$

Aplicar el método de regresión logística para estimar la función de probabilidad y clasificar el punto con coordenadas (4,8), representado como un círculo vacío "○", mostrado en la figura 3.4a, con base en esta probabilidad estimada.

### Solución

De manera similar al ejemplo 3.3, usando la ecuación (3.10 ), primero que todo encontramos los valores de la matrices $X,\ W$ y los vectores $\boldsymbol{y},\boldsymbol{p}$, como a continuación se indica:

$$X = \begin{bmatrix}1&2&5\\1&4&3\\1&5&6\\1&5&8\\1&7&6\\1&6&4\end{bmatrix},$$

$$\boldsymbol{y} = \begin{bmatrix}0\\0\\0\\1\\1\\1\end{bmatrix},$$

$$\boldsymbol{p} = \begin{bmatrix} \mathcal{P}(\boldsymbol{x}_1, \boldsymbol{\beta}(k)) \\ \mathcal{P}(\boldsymbol{x}_2, \boldsymbol{\beta}(k)) \\ \vdots \\ \mathcal{P}(\boldsymbol{x}_6, \boldsymbol{\beta}(k)) \end{bmatrix},$$

$$W = \begin{bmatrix} \mathcal{P}(\boldsymbol{x}_1,\boldsymbol{\beta}(k))[1-\mathcal{P}(\boldsymbol{x}_1,\boldsymbol{\beta}(k))] & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathcal{P}(\boldsymbol{x}_2,\boldsymbol{\beta}(k))[1-\mathcal{P}(\boldsymbol{x}_2,\boldsymbol{\beta}(k))] & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathcal{P}(\boldsymbol{x}_3,\boldsymbol{\beta}(k))[1-\mathcal{P}(\boldsymbol{x}_3,\boldsymbol{\beta}(k))] & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathcal{P}(\boldsymbol{x}_4,\boldsymbol{\beta}(k))[1-\mathcal{P}(\boldsymbol{x}_4,\boldsymbol{\beta}(k))] & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathcal{P}(\boldsymbol{x}_5,\boldsymbol{\beta}(k))[1-\mathcal{P}(\boldsymbol{x}_5,\boldsymbol{\beta}(k))] & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathcal{P}(\boldsymbol{x}_6,\boldsymbol{\beta}(k))[1-\mathcal{P}(\boldsymbol{x}_6,\boldsymbol{\beta}(k))] \end{bmatrix}.$$

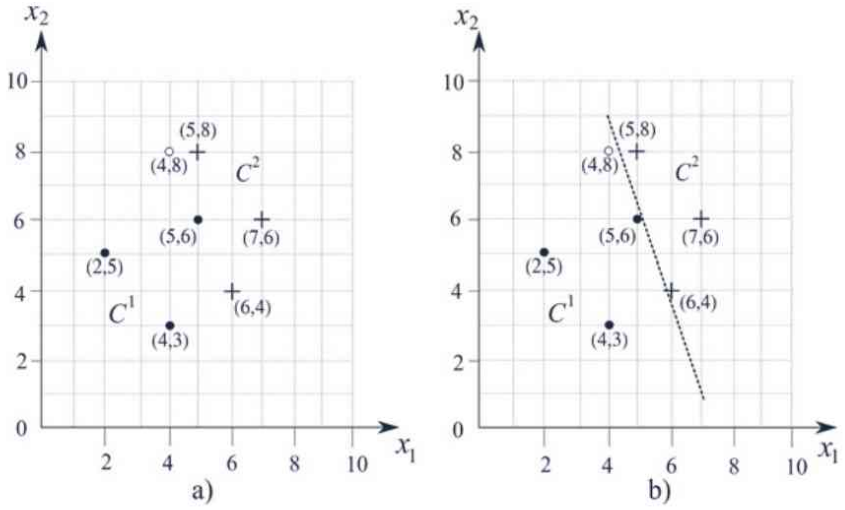



**Figura 3.4**: Conjunto de datos y punto a clasificar.

Enseguida, como parte del procedimiento de iteración de la función de probabilidad $\mathcal{P}(\boldsymbol{x}_{i_n}, \boldsymbol{\beta}(k)) = \frac{e^{(\boldsymbol{\beta}^T(k)\boldsymbol{x}_{i_n})}}{1+e^{[\boldsymbol{\beta}^T(k)\boldsymbol{x}_{i_n})}}$, se requiere un valor inicial (también conocido como condiciones iniciales) $\boldsymbol{\beta}(0) \in \mathbb{R}^{3\times 1}$ para el vector $\boldsymbol{\beta}(k) \in \mathbb{R}^{3\times 1}$. Supongamos que este valor es el vector $\mathbf{0} \in \mathbb{R}^{3\times 1}$.

En consecuencia:

$$\boldsymbol{p} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \in \mathbb{R}^{6\times 1},\ W = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} \end{bmatrix} \in \mathbb{R}^{6\times 6}.$$

Al aplicar la primera iteración ($k = 1$) a través de la ecuación (3.10), un cálculo a mano conduce al siguiente resultado:

$$\boldsymbol{\beta}(1) = \begin{bmatrix} -5.1744186 \\ 0.6627907 \\ 0.34883721 \end{bmatrix}.$$

Al aplicar la segunda iteración, es decir cuando $k = 2$, se obtiene lo siguiente:

$$\boldsymbol{\beta}(2) = \begin{bmatrix} -7.15885021 \\ 0.97173851 \\ 0.39690093 \end{bmatrix}.$$

Si se procede así hasta la octava iteración ($k = 8$), entonces el siguiente resultado numérico se obtiene:

$$\boldsymbol{\beta}(8) = \begin{bmatrix} -8.47069131 \\ 1.17223909 \\ 0.4235152 \end{bmatrix}.$$

De nuevo, si detenemos el proceso en esta iteración, podremos representar la línea de separación correspondiente entre las dos clases.

$$fd(\boldsymbol{x}) = 1.17223909x_1 + 0.4235152x_2 - 8.47069131. \tag{3.12}$$

Al igualar a 0 y despejar para $x_2$, se tiene que:

$$x_2 = 20.0 - 2.7679x_1.$$

En la figura 3.4b aparece representada esta línea de separación.

Finalmente, procedamos a representar el punto con coordenadas (4,8), representado como un círculo vacío "○" y mostrado en las figuras 3.4a y 3.4b. Como ya vimos, para encontrar la clase de pertenencia de este punto, sustituyamos sus coordenadas en la ecuación (3.12):

$$fd(\boldsymbol{x}) = 1.17223909 \cdot 4 + 0.4235152 \cdot 8 - 8.47069131 = -0.3936.$$

Como el signo es negativo, de acuerdo con la teoría establecida, el punto con coordenadas (4,8) se considera como perteneciente a la clase $C^1$, lo cual es coincidente con lo observado en la figura 3.4b, donde se ve que dicho punto se encuentra del lado de la clase $C^1$.

A continuación se describe en el cuadro de código **MATLAB** 3.4 la documentación y explicación técnica del programa fuente cap3_RegresionLogisticaA.m, con el cual se da la solución al ejemplo 3.4.

**Código MATLAB 3.4 cap3_RegresionLogisticaA.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 3. Discriminación lineal y regresión logística.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap3_RegresionLogisticaA.m **MATLAB** versión 2020b

```
1  clc; % limpia pantalla.
2  clearvars; % remueve todas las variables del espacio actual de trabajo.
3  close all; % cierra gráficas, archivos y recursos abiertos.
4  format longG % formato largo.
5  x1_a=[1; 2; 5]; x2_a=[1; 4; 3]; x3_a=[1; 6; 6]; % vectores de la clase C^1.
6  x4_a=[1; 5; 8]; x5_a=[1; 7; 6]; x6_a=[1; 6; 4]; % vectores de la clase C^2.
7  y1=0; y2=0; y3=0; y4=1; y5=1; y6=1;
8  % vector y ∈ IR^{6×1}.
9  y=[y1; y2; y3; y4; y5; y6];
10 % Matriz X ∈ IR^{6×6}.
11 X=[x1_a'; x2_a'; x3_a'; x4_a'; x5_a'; x6_a'];
12 % condiciones iniciales del vector β.
13 beta_k=[0; 0; 0]; % valor inicial β(0) = [0,0,0]^T.
14 figure
15 plot(x1_a(2),x1_a(3),'ko' ,x2_a(2),x2_a(3), 'ko' ,x3_a(2), x3_a(3), 'ko' , 'MarkerFaceColor' ,[0,0,0])
16 hold on
17 plot(x4_a(2), x4_a(3),'k+' , x5_a(2), x5_a(3), 'k+' , x6_a(2), x6_a(3), 'k+' )
18 plot(4,8 ,'ko' ) % punto con coordenadas (4,8).
19 axis([0, 10, 0, 10]);
20 grid on
```

**Continúa código 3.4: cap3_RegresionLogisticaA.m**

```
for k=1:8 % proceso recursivo del vector β(k+1) = β(k) + [X^T W X]^-1 X^T [y - p].
    p=[probab(x1_a, beta_k);
        probab(x2_a, beta_k);
        probab(x3_a, beta_k);
        probab(x4_a, beta_k);
        probab(x5_a, beta_k);
        probab(x6_a, beta_k)]; % vector p ∈ IR^{6×1}.
    W=[probab(x1_a, beta_k)*(1-probab(x1_a, beta_k)), 0, 0,0,0,0;
        0, probab(x2_a, beta_k)*(1-probab(x2_a, beta_k)), 0, 0, 0, 0;
        0, 0, probab(x3_a, beta_k)*(1-probab(x3_a, beta_k)), 0,0,0;
        0,0,0, probab(x4_a, beta_k)*(1-probab(x4_a, beta_k)),0,0;
        0,0,0,0, probab(x5_a, beta_k)*(1-probab(x5_a, beta_k)),0;
        0,0,0,0,0,probab(x6_a, beta_k)*(1-probab(x6_a, beta_k))]; %W ∈ IR^{6×6}.
    k % despliega el número de iteraciones.
    beta_k=beta_k+ (X'*W*X)^(-1)*X'*(y-p) % β_{k+1} = β_k + [X^T W X]^-1 X^T [y - p]
    disp( 'Para continuar, oprima cualquier tecla' )
    pause
end
figure
plot(x1_a(2),x1_a(3),'ko' ,x2_a(2),x2_a(3),'ko' ,x3_a(2),x3_a(3),'ko' , 'MarkerFaceColor' ,[0,0,0])
hold on
plot(x4_a(2), x4_a(3),'k+' , x5_a(2), x5_a(3), 'k+' , x6_a(2), x6_a(3), 'k+' )
plot(4,8 ,'ko' ) % punto con coordenadas (4,8) a clasificar.
axis([0, 10, 0, 10]);
grid on
plot([4, 7], [20-2.7679*(4), 20-2.7679*(7)],'--k' )
% Función probabilidad P(x_ia, β(k))
function probalidad=probab(xia, beta_k)
    probalidad=exp(beta_k'*xia)/(1+exp(beta_k'*xia)); % P(x_{i_a}, β(k)) = e^{(β^T(k) x_{i_a})} / (1 + e^{(β^T(k) x_{i_a})}).
end
```

Con la finalidad de fortalecer los conocimientos del lector en la metodología de regresión logística, en el sitio Web de la presente obra, se encuentran ejemplos adicionales desarrollados con programación en MATLAB.

## 3.4 Resumen

En este capítulo se estudiaron dos técnicas muy útiles para el aprendizaje de máquinas. La primera técnica útil para la discriminación entre clases de patrones en forma lineal se fundamenta en el cálculo de las probabilidades a posteriori de las diferentes clases de patrones.

Primeramente, la probabilidad a posteriori de cada una de las clases involucradas en el problema es calculada. Con base en estas probabilidades, enseguida la función de probabilidad a posteriori es calculada a través del conocido teorema de Bayes.

Estas funciones de distribución a posteriori son después modeladas mediante funciones multivariables de tipo gaussiano, las cuales son usadas más adelante en la construcción de las correspondientes de discriminación por clase. Finalmente son utilizadas para estimar las funciones de discriminación entre clases, para así emplearlas en la determinación de la clase de pertenencia de un patrón de entrada dado.

La segunda técnica estudiada en este capítulo, conocida como regresión logística, también estima la probabilidad de la clase de pertenencia de un patrón de entrada. Dicha estimación se hace de manera directa, en lugar de manera indirecta a través del bien conocido teorema de Bayes, como se hace en el caso de la discriminación lineal.

Para esto, la función de probabilidad a posteriori es modelada. Esto se hace a través del logaritmo del cociente entre la probabilidad y su complemento de cada clase, lo cual permite representar la probabilidad de cada clase como un cociente entre dos exponenciales. Enseguida, se estima la función de probabilidad a posteriori. Esto se hace mediante el producto de probabilidades de los datos, lo cual permite obtener la conocida función de verosimilitud. Finalmente, esta función de verosimilitud es maximizada a través del método de Newton-Raphson multivariable.

En ambos casos, en discriminación lineal y regresión logística se presentan ejemplos ilustrativos para que el lector entienda y asimile de manera fácil el material expuesto con su correspondiente programación en código fuente MATLAB, acompañado de su documentación técnica.

Enseguida, se presenta un conjunto de ejercicios que permitirán reafirmar el conocimiento adquirido y ejercitar las habilidades como programador.

## 3.5 Problemas propuestos

En esta sección el lector encontrará un conjunto de problemas propuestos, cuyo desarrollo y solución le permitirán reafirmar el conjunto de conceptos vistos en este capítulo.

3.5.1 Considere el siguiente conjunto de datos mostrado en la figura 3.5:

$$\left\{ \boldsymbol{x}_1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix},\ \boldsymbol{x}_2 = \begin{bmatrix} 2 \\ 5 \end{bmatrix},\ \boldsymbol{x}_3 = \begin{bmatrix} 4 \\ 7 \end{bmatrix} \right\} \in C^1$$

$$\left\{ \boldsymbol{x}_4 = \begin{bmatrix} 6 \\ 2 \end{bmatrix},\ \boldsymbol{x}_5 = \begin{bmatrix} 8 \\ 3 \end{bmatrix},\ \boldsymbol{x}_6 = \begin{bmatrix} 8 \\ 7 \end{bmatrix} \right\} \in C^2$$

Note que tres vectores pertenecen a la clase $C^1$, mientras que otros tres pertenecen a la clase $C^2$. Realizar lo siguiente:

a) Encontrar los estimados de las funciones discriminantes: $fd_{C^1}(\boldsymbol{x})$ y $fd_{C^2}(\boldsymbol{x})$.

b) Encontrar la línea de separación o de decisión entre ambas clases.

c) Determinar la clase de los patrones: $\boldsymbol{x}_a = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$ y $\boldsymbol{x}_b = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$.

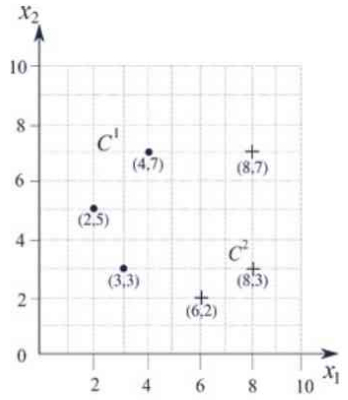



**Figura 3.5**: Conjunto de datos para el problema 3.5.1.

3.5.2 Considere el siguiente conjunto de datos mostrado en la figura 3.6:

$$\left\{\boldsymbol{x}_1 = \begin{bmatrix}3\\5\end{bmatrix},\ \boldsymbol{x}_2 = \begin{bmatrix}2\\4\end{bmatrix},\ \boldsymbol{x}_3 = \begin{bmatrix}3\\4\end{bmatrix},\ \boldsymbol{x}_4 = \begin{bmatrix}4\\4\end{bmatrix},\ \boldsymbol{x}_5 = \begin{bmatrix}5\\5\end{bmatrix}\right\} \in C^1$$

$$\left\{\boldsymbol{x}_6 = \begin{bmatrix}10\\8\end{bmatrix},\ \boldsymbol{x}_7 = \begin{bmatrix}6\\8\end{bmatrix},\ \boldsymbol{x}_8 = \begin{bmatrix}8\\8\end{bmatrix},\ \boldsymbol{x}_9 = \begin{bmatrix}8\\3\end{bmatrix},\ \boldsymbol{x}_{10} = \begin{bmatrix}8\\6\end{bmatrix}\right\} \in C^2.$$

Para este ejemplo, cinco vectores pertenecen a la clase $C^1$ y cinco vectores pertenecen a la clase $C^2$. Hacer lo siguiente:

a) Encontrar los estimados de las funciones discriminantes: $fd_{C^1}(\boldsymbol{x})$ y $fd_{C^2}(\boldsymbol{x})$.

b) Encontrar las líneas de separación de decisión entre pares de clases, y

c) Determinar la clase de los vectores: $\boldsymbol{x}_a = \begin{bmatrix}3\\6\end{bmatrix}$ descrito por un círculo vacío ○, y $\boldsymbol{x}_b = \begin{bmatrix}4\\8\end{bmatrix}$ representado por un cuadro en blanco □.

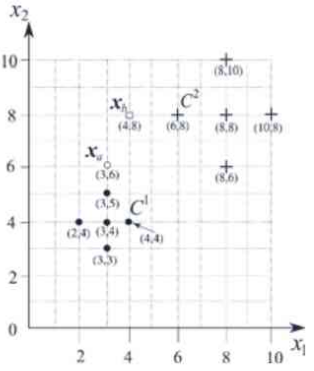



**Figura 3.6**: Conjunto de datos para el problema 3.5.2.

3.5.3 Considere el siguiente conjunto de datos mostrado en la figura 3.7:

$$\left\{\boldsymbol{x}_1 = \begin{bmatrix}2\\3\end{bmatrix},\ \boldsymbol{x}_2 = \begin{bmatrix}2\\5\end{bmatrix},\ \boldsymbol{x}_3 = \begin{bmatrix}4\\5\end{bmatrix}\right\} \in C^1$$
$$\left\{\boldsymbol{x}_4 = \begin{bmatrix}5\\8\end{bmatrix},\ \boldsymbol{x}_5 = \begin{bmatrix}7\\8\end{bmatrix},\ \boldsymbol{x}_6 = \begin{bmatrix}8\\10\end{bmatrix}\right\} \in C^2$$
$$\left\{\boldsymbol{x}_7 = \begin{bmatrix}8\\2\end{bmatrix},\ \boldsymbol{x}_8 = \begin{bmatrix}8\\4\end{bmatrix},\ \boldsymbol{x}_9 = \begin{bmatrix}10\\6\end{bmatrix}\right\} \in C^3.$$

Para este ejemplo, tres vectores pertenecen a la clase $C^1$, tres vectores pertenecen a la clase $C^2$ y tres pertenecen a la clase $C^3$. Llevar a cabo lo siguiente:

a) Encontrar los estimados de las funciones discriminantes: $fd_{C^1}(\boldsymbol{x})$, $fd_{C^2}(\boldsymbol{x})$ y $fd_{C^3}(\boldsymbol{x})$.

b) Encontrar las líneas de separación decisión entre pares de clases.

c) Determinar la clase de los puntos: $\boldsymbol{x} = \begin{bmatrix}5\\2\end{bmatrix}$, $\boldsymbol{x} = \begin{bmatrix}7\\7\end{bmatrix}$ y $\boldsymbol{x} = \begin{bmatrix}8\\5\end{bmatrix}$.

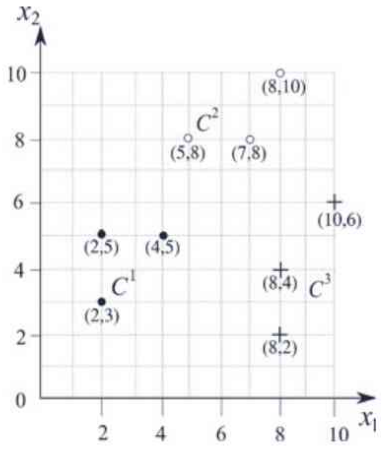



**Figura 3.7**: Conjunto de datos para el problema 3.5.3.

3.5.4 Elabore en **MATLAB** un programa que permita encontrar mediante el método de discriminación lineal descrito en la sección 3.2, lo siguiente:

a) Los estimados de las funciones discriminantes: $fd_{C^1}$, $fd_{C^2}, \cdots fd_{C^p}$.

b) Las líneas de separación entre pares de clases.

c) Determinar la clase de un punto $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$.

Verifique la correcta funcionalidad de dicho programa al aplicarlo a los problemas 3.5.1 y 3.5.3 antes descritos.

3.5.5 Considere el siguiente conjunto de datos mostrado en la figura 3.8:

$$\left\{\left(\boldsymbol{x}_1=\begin{bmatrix}2\\2\end{bmatrix},\ y_1=0\right),\ \left(\boldsymbol{x}_2=\begin{bmatrix}3\\4\end{bmatrix},\ y_2=0\right),\ \left(\boldsymbol{x}_3=\begin{bmatrix}6\\5\end{bmatrix},\ y_3=0\right)\right\}$$

$$\left\{\left(\boldsymbol{x}_4=\begin{bmatrix}5\\3\end{bmatrix},\ y_4=1\right),\ \left(\boldsymbol{x}_5=\begin{bmatrix}5\\8\end{bmatrix},\ y_5=1\right),\ \left(\boldsymbol{x}_6=\begin{bmatrix}9\\6\end{bmatrix},\ y_6=1\right)\right\}.$$

Aplicar el método de regresión logística para estimar la función de probabilidad y clasificar el punto con coordenadas (2,7), representado como un círculo vacío "○" y mostrado en la figura 3.8, con base en esta probabilidad estimada.

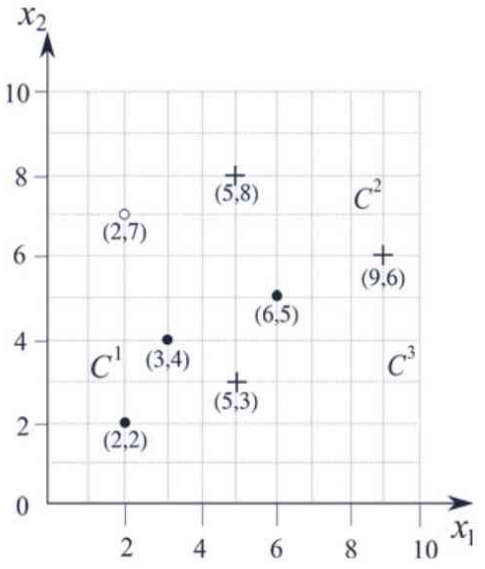



**Figura 3.8**: Conjunto de datos para el problema 3.5.5.

3.5.6 Considere el siguiente conjunto de datos mostrado en la figura 3.9:

$$\left\{\left(\boldsymbol{x}_1=\begin{bmatrix}3\\7\end{bmatrix},\ y_1=0\right),\ \left(\boldsymbol{x}_2=\begin{bmatrix}6\\6\end{bmatrix},\ y_2=0\right),\ \left(\boldsymbol{x}_3=\begin{bmatrix}8\\4\end{bmatrix},\ y_3=0\right),\ \left(\boldsymbol{x}_4=\begin{bmatrix}8\\2\end{bmatrix},\ y_4=0\right)\right\}$$

$$\left\{\left(\boldsymbol{x}_5=\begin{bmatrix}3\\1\end{bmatrix},\ y_5=1\right),\ \left(\boldsymbol{x}_6=\begin{bmatrix}2\\3\end{bmatrix},\ y_6=1\right),\ \left(\boldsymbol{x}_7=\begin{bmatrix}3\\5\end{bmatrix},\ y_7=1\right),\ \left(\boldsymbol{x}_8=\begin{bmatrix}5\\7\end{bmatrix},\ y_8=1\right)\right\}.$$

a) Aplicar el método de regresión logística para estimar la función de probabilidad y clasificar el punto con coordenadas (5,4), el cual se encuentra representado como un círculo vacío "○" y mostrado en la figura 3.9, con base en esta probabilidad estimada.

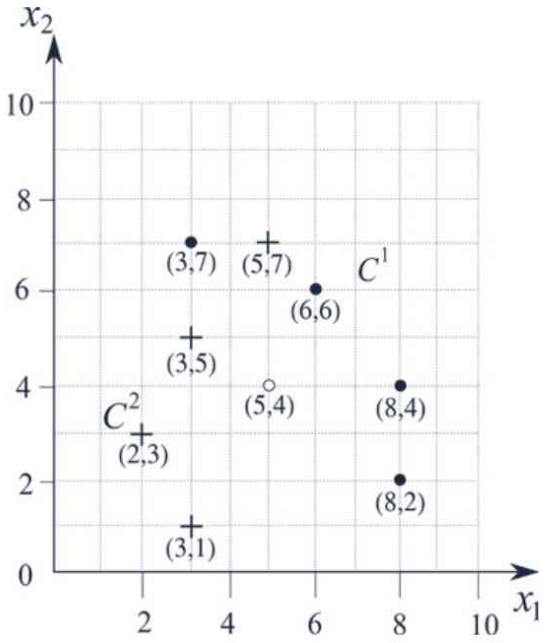



**Figura 3.9**: Conjunto de datos para el problema 3.5.6.

3.5.7 Elabore en **MATLAB** un programa que permita encontrar, mediante el método de regresión logística descrito en la sección 3.3, la función de probabilidad requerida para poder clasificar un punto con coordenadas

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Verifique la correcta funcionalidad de dicho programa, si se aplica a los siguientes:

a) Problema 3.5.5, antes descrito.

b) Problema 3.5.6, anteriormente descrito.

# Capítulo 4

# Tratamiento de imágenes

Filtrado de imágenes

```
for y=h hasta y=(m-1)
    for  x=h hasta x=(n-1)
        v=0;
        for i=1 hasta i=h
            for  j=1 hasta j=h
                v=v+(f(x+i+1-h, y+j+1-h)*h_pg(i,j));
            end
        end
        g(x,y)=trunc(v);
    end
end
```

# Descripción del capítulo

En este capítulo se presentan los conceptos fundamentales, así como las técnicas básicas de filtrado relacionadas con el tratamiento o procesamiento digital de imágenes. Primeramente, se ofrecen los conceptos de imagen, imagen digital, imagen binaria e histograma de una imagen. En todos los casos se muestran ejemplos alusivos para que el lector asimile rápidamente y de manera sencilla dichos conceptos. Enseguida se dan los detalles sobre el acondicionado de una imagen. Se expone el concepto de ruido en una imagen y se muestran ejemplos. Después se habla del proceso de filtrado como herramienta computacional para reducir el efecto del ruido causado en una imagen. Se expone la operación de tres filtros muy usados por la comunidad en el acondicionado de una imagen: filtrado promedio aritmético, filtrado gaussiano y filtrado mediano. En los tres casos se presentan ejemplos. Más adelante, se explica cómo una imagen puede mejorarse en su contraste a través de la modificación de su histograma. Al final se ofrece un conjunto de ejercicios para que el lector afiance los conocimientos aprendidos y ponga en práctica sus habilidades como programador.

Los siguientes temas son abordados:

- Procesamiento básico de imágenes.
- Ruido en imágenes.
- Acondicionamiento y filtrado de imágenes.
- Filtrado por promedio aritmético, gaussiano y mediano.
- Mejoramiento del contraste de una imagen.

# 4.1 Introducción

Hoy en día, los robots se utilizan en múltiples situaciones: en labores de rescate, de manipulación de residuos sólidos, en la enseñanza, etcétera. Para que estos robots sean capaces de desplazarse y llevar a cabo las tareas dentro de su espacio de trabajo, dichas máquinas se valen de sensores, por ejemplo, la cámara digital; a través de este dispositivo, el robot adquiere imágenes o vídeo del entorno circundante. Estas imágenes deben ser tratadas y analizadas para que el robot sea capaz de ubicar e identificar los objetos a su alrededor y así desplazarse entre ellos y eventualmente para realizar tareas específicas con los mismos, como manipuladores. Para esto el sistema de cómputo del robot debe realizar sobre las imágenes adquiridas un conjunto de pasos estratégicamente escogidos, con el objeto de obtener el mejor desempeño posible.

En la figura 4.1 se muestra una de tantas estrategias. Como se puede ver después de que una imagen ha sido adquirida a través de la cámara o una de las cámaras del robot, es tratada para reducir de ella el ruido causado inherente al proceso de adquisición, las condiciones atmosféricas, diferentes fuentes de iluminación, brillos en los objetos, sombras entre objetos, etcétera. Otra operación de tratamiento que se puede hacer sobre la imagen de entrada es mejorar su contraste con el objetivo de realzar su apariencia visual.

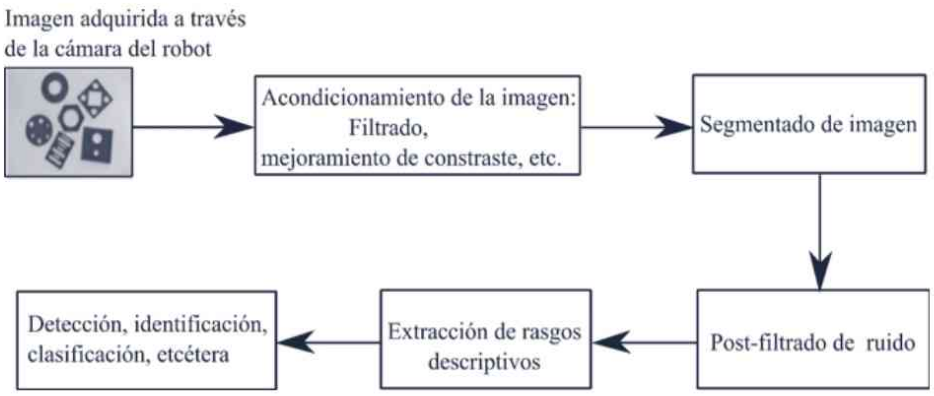



**Figura 4.1**: Secuencia de pasos realizados sobre una imagen a la entrada del sensor del robot para que este último sea capaz de ubicarse, desplazarse y realizar tareas con los objetos a su alrededor dentro de su espacio de trabajo.

Una vez que la imagen de entrada ha sido tratada en forma digital, esta debe ser segmentada, es decir, dividida en regiones con un significado lo más cercano al de los diferentes objetos en la escena observada por la cámara. Las regiones segmentadas todavía pueden venir acompañadas de ruido parásito. Este ruido debería ser eliminado o reducido para un mejor éxito en etapas posteriores. Esto se puede lograr mediante técnicas de post-filtrado, como se muestra en la figura 4.1. Enseguida, para cada uno de los objetos segmentados de la imagen se calcula un conjunto de rasgos de tipo geométrico o topológico que permiten describirlos en forma global o local. Estas descripciones pueden ser usadas en una etapa final para detectar, identificar y ubicar dentro de la imagen y, en consecuencia, dentro del espacio de trabajo del robot a los diferentes objetos.

En este capítulo describe en primer lugar, un conjunto de técnicas comúnmente utilizadas para reducir el ruido en imágenes. Para esto se presenta un conjunto de conceptos y definiciones útiles para que el lector sea capaz de entender la operación de los métodos y técnicas expuestas. Los conceptos que se presentan son el de imagen, el de imagen digital, el de imagen binaria y el de histograma. Los cuatro conceptos son ilustrados con ejemplos para que lector entienda y asimile de manera sencilla los conceptos.

En segundo lugar se presentan los conceptos fundamentales relacionados con el tratamiento o acondicionado de una imagen. Se muestra la operación de dos conjuntos de técnicas, por un lado, las relacionadas con el filtrado de una imagen para reducir el efecto del ruido que puede venir durante el proceso de adquisición. Por otro lado, se ilustra cómo el histograma de una imagen puede ser usado para mejorar el contraste al ensanchar o ecualizar los niveles de gris de la imagen. Todos los casos se ilustran con ejemplos.

## 4.2 Conceptos básicos sobre imágenes

En esta sección se presenta un conjunto de definiciones básicas sobre imágenes para seguir fácilmente la lectura del material correspondiente de este capítulo y también de los subsecuentes. Se proporcionan las definiciones de imagen, imagen digital y de imagen binaria. Además, se presenta e ilustra el concepto de histograma el cual, como veremos más adelante, resulta muy útil para la segmentación de imágenes así como para el mejoramiento de contraste.

**Definición 4.1: Función imagen**

Una función de imagen o imagen es cualquier función real $g(x, y)$ con integral finita y soporte compacto $SC$ tal que, para todo punto $p \in SC$, $x > 0$, $y > 0$.

Para ser útil, una imagen de acuerdo con la definición anterior requiere ser muestreada y cuantizada [16] y [17]. Estas operaciones se realizan normalmente a través de lo que se conoce como digitalizador o numerizador. Para más detalles, referirse por ejemplo a [16]. El resultado de digitalizar una imagen, según la definición (4.1), da como resultado una imagen digital, la cual puede ser definida como sigue:

**Definición 4.2: Imagen digital**

Una imagen digital denotada como $f(x, y)$ es un arreglo bidimensional $\in Z \times Z$, si $(x, y)$ son números enteros de $Z \times Z$, y $f$ una función que asigna a cada par $(x, y)$ un número de $Z$; donde $Z$ es el conjunto de los enteros.

**Ejemplo 4.1**

En la figura 4.2 se muestran tres imágenes digitales discreteadas a una resolución de 416×312 píxeles, la primera con tres objetos, la segunda con cuatro y la tercera con cinco. Note la complejidad variable de los objetos en las tres imágenes.

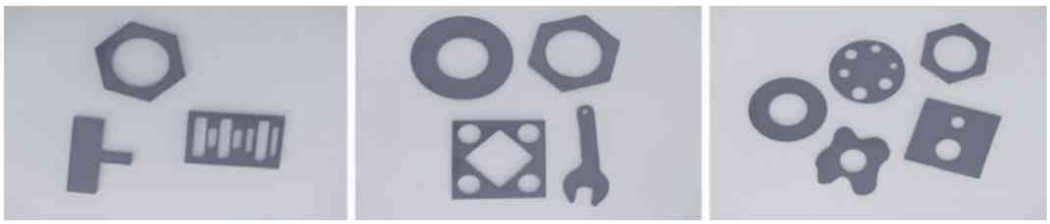


**Figura 4.2**: Tres imágenes digitales muestreadas a una resolución de 416×312 píxeles y cuantizadas a 256 niveles de gris.

**Definición 4.3: Imagen binaria**

Una imagen binaria denotada como $b(x, y)$ es una imagen digital $f(x, y)$ que ha sido cuantizada a dos niveles de intensidad, 0 y 1.

**Ejemplo 4.2**

En la figura 4.3 se muestran tres versiones binarias de las imágenes digitales de la figura 4.2. Note que a pesar de que la información de las imágenes originales fue reducida de 256 a dos niveles de gris, la información esencial de los objetos contenidos se sigue manteniendo. Esto, como veremos más adelante, es esencial en tareas como la detección y la identificación de objetos.

**Figura 4.3**: Versiones binarias de las tres imágenes digitales de la figura 4.2.

**Definición 4.4: Histograma de una imagen**

El histograma de una imagen $f(x, y)$ con $l$ niveles de intensidad o de gris en el rango $[0, l-1]$, denotado como $h(r_i)$, es una función discreta, dada por:

$$h(r_i) = \frac{n_i}{n_T} \tag{4.1}$$

donde $r_i$ es el $i$-ésimo nivel de gris, $n_i$ es el número de píxeles en la imagen con el nivel de intensidad $r_i$ y, $n_T$ el número total de píxeles en la imagen $f(x, y)$.

**• Ejemplo 4.3**

Desarrollar un algoritmo en **MATLAB** que permita obtener el histograma de una imagen en tonos de gris.

**Solución**

En la figura 4.4 se muestran dos imágenes de 416×312 píxeles en 256 niveles de gris, una imagen con tres objetos y la otra con seis y sus correspondientes histogramas. Un procedimiento sencillo para obtener el histograma de una imagen es el que se muestra con pseudocódigo en el cuadro 4.1.

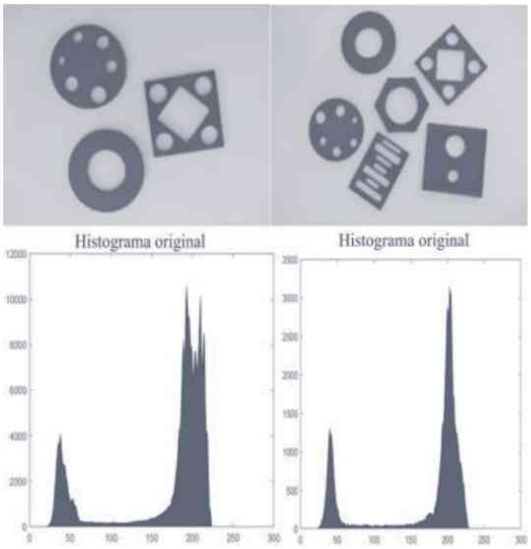



**Figura 4.4**: Dos imágenes de 416×312 píxeles y 256 niveles de gris, una con tres objetos y la otra con seis objetos y sus correspondientes histogramas. Note cómo en ambos casos los histogramas son bi-modales.

**Pseudocódigo 4.1**

Algoritmo para obtener el histograma de una imagen $f(x, y)$

Definir un contador $c$ con tantas casillas como niveles de gris se puedan encontrar en la imagen. Si se sabe por ejemplo, que la imagen fue cuantizada a 256 niveles de intensidad, entonces el contador $c$ contendrá 256 casillas.

Inicializar en cero todas las casillas del contador $c(i) = 0,\ i = 1, \cdots, l.$
**for** y=1 hasta y=m
  **for** x=1 hasta x=n
    i=f(x,y); % $f(x, y)$ es el nivel de gris del píxel con coordenadas $(x, y)$.
    c(i)=c(i)+1; % la casilla $i$ del contador $c(i)$ es incrementada en 1.
  **end**
**end**
Dibujar histograma;

Al final de la ejecución de este procedimiento, el contador $c(i)$ se encontrará lleno de números. Cada uno de estos números representará, como ya se dijo anteriormente, las veces que un nivel de gris dado se encuentra en la imagen.

La idea presentada en el pseudocódigo 4.1 se encuentra implementada en lenguaje MATLAB con el programa cap4_Histograma.m, cuya documentación y descripción están contenidas en el cuadro de código 4.1. En la línea de programación 4 se descarga a memoria la imagen de prueba cap4_imaprueba1.png, y su correspondiente visualización se realiza en la línea 6. Las dimensiones $n \times m$ de la imagen $f(x, y)$ se obtienen en la línea de programación 8; mientras que el contador $c(i)$ se inicializa en ceros (ver línea 9). Las dos imágenes de prueba fueron cuantizadas en 256 niveles de gris, estos se asignan a la variable: niveles, tal y como se indica en la línea 10.

**Procedimiento para ejecutar el programa: cap4_Histograma.m**

- Entrada: imagen en niveles de gris $f(x, y)$.
- Salida: histograma $h(r_i)$.
- Los archivos de las imágenes de prueba: cap4_imaprueba1.png y cap4_imaprueba2.png, así como el código fuente en MATLAB del programa cap4_Histograma.m (ver cuadro de código 4.1) se encuentran disponibles en el sitio Web de esta obra.
- Descargar todos los archivos en la misma carpeta, cuya trayectoria en disco duro es definida y seleccionada por el usuario.

El código principal del histograma se encuentra entre las líneas 11-16; observe que las dimensiones obtenidas en la línea 8 se utilizan para hacer el barrido adecuado de los $n$ renglones y $m$ columnas de la imagen $f(x, y)$. De hecho, en la línea 13 se obtiene el nivel de gris de la imagen $f$ correspondiente a sus respectivas coordenadas $(x, y)$. Este nivel de gris, se contabiliza en el registro $i$-ésimo de la variable contador $c(i)$.

En la línea 14 se dibuja el histograma de la imagen $f(x, y)$; y, para propósitos de análisis, este resultado se compara con el histograma de la misma imagen $f(x, y)$ usando la función de MATLAB imhist(f) en la línea 20 (su resultado gráfico se lleva a cabo en la línea 21).

El lector puede también probar el algoritmo con la imagen cap4_imaprueba2.png. Como se esperaba, en ambos casos, los dos histogramas son bi-modales.

**Código MATLAB 4.1 cap4_Histograma.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 4. Tratamiento de imágenes.
Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**"  .

Programa: cap4_Histograma.m MATLAB versión 2020b

```
1  clc; % limpia pantalla.
2  clearvars; % remueve todas las variables del espacio actual de trabajo.
3  close all; % cierra gráficas, archivos y recursos abiertos.
4  f=imread( 'cap4_imaprueba1.png' ); % lee imagen de prueba f(x,y).
5  figure(1)
6  imshow(f) % muestra imagen de prueba (ver figura 4.4).
7  title( 'Imagen original' )
8  [n, m]=size(f); % dimensiones de la imagen f(x,y).
9  c=zeros(256,1); % inicializa en ceros el registro del contador c(i).
10 niveles=1:256; % niveles de gris.
11 for x=1:n  % barrido en renglones de la imagen f(x,y).
12     for y=1:m % barrido en columnas de la imagen f(x,y).
13         i=f(x,y); % obtiene el nivel de gris de f(x,y).
14         c(i)=c(i)+1;  % contador de píxeles a niveles de gris.
15     end
16 end
17 figure(2) % (ver figura 4.4).
18 fill(niveles,c, 'b' ); % dibuja histograma h(r_i) de la imagen f(x,y).
19 title( 'Histograma original' ); figure(3)
20 [Contador_matlab,nivel_gris] = imhist(f);  % hist., con función MATLAB.
21 fill(nivel_gris,Contador_matlab, 'x' , niveles, Contador, 'O' );
22 title( 'Histograma con función de Matlab' );
```

## 4.3 Acondicionamiento de imágenes

En esta se sección se describe, por un lado, un conjunto de técnicas para reducir la influencia de ruido en imágenes. En particular, se expone la operación de tres filtros en el dominio de la imagen: 1) filtro promedio aritmético, 2) el filtro gaussiano y 3) el filtro mediano. Los dos primeros son de tipo lineal, mientras que el tercero es de tipo no lineal. Por otro lado, se estudia un conjunto reducido de técnicas para modificar el contraste de la imagen de entrada o pre-procesada a través de un filtro de ruido.

El objetivo de la aplicación de un modificador de contraste de imagen es mejorar su apariencia visual. En ambos casos se busca que el desempeño del sistema completo de detección y reconocimiento de objetos ofrezca el mejor desempeño posible para que en consecuencia el robot sea capaz de realizar correctamente la tarea encomendada dentro de su espacio de trabajo.

### 4.3.1 Ruido en imágenes

Las imágenes al ser adquiridas usualmente vienen contaminadas con algún tipo de ruido. El ruido puede deberse a varias causas, por ejemplo, durante el proceso de adquisición. Al tomar varias imágenes de una misma escena a través del mismo captor, aun en situaciones controladas, uno puede darse cuenta de que el valor de un píxel a lo largo de la secuencia de imágenes no es el mismo.

Estas fluctuaciones pueden introducir errores en procesamientos posteriores. El ruido puede deberse también a las siguientes causas:

1) Fluctuaciones espurias en los niveles de gris de los píxeles, introducidas por el sistema de adquisición.

2) Fluctuaciones aleatorias o inexactitudes en los datos, precisión limitada del equipo de cómputo usado o redondeo.

3) Agrupamientos erróneos de partes de los objetos en las imágenes.

### 4.3.2 Algunos tipos de ruido

En la sección anterior se mencionó que las imágenes al ser adquiridas pueden venir contaminadas con ruido. Esto puede afectar la operación del sistema de detección e identificación de objetos. Algunos tipos de ruido que suelen usarse para probar el desempeño de filtros son los siguientes. Sea $f(x, y)$ una imagen adquirida mediante una de las cámaras del robot. En lo que sigue, diremos que esta imagen puede venir alterada o contaminada con ruido mezclado como se muestra. Una versión ruidosa $\tilde{f}(x, y)$ de la imagen de entrada $f(x, y)$ se obtiene al sumar o restar al azar a cada píxel de $f(x, y)$ un entero $c$, tal que $\tilde{f}(x, y) = f(x, y) \pm c$.

Una manera de generar una versión ruidosa $\tilde{f}(x, y)$ de una imagen $f(x, y)$, es la siguiente:

$$\tilde{f}(x,y) \quad = \quad \begin{cases} f(x,y) & a < l^* \\ f(x,y) + bc & a \geq l^* \end{cases} \tag{4.2}$$

En este caso, $a \in [0, 1]$ es una variable aleatoria uniformemente distribuida, el parámetro $l^*$ controla cuanto de la señal $f(x, y)$ es contaminada, la variable $c$ determina qué tan severo es el ruido. Cuando $b = 1$ el ruido adicionado a $f(x, y)$ es de tipo aditivo. De igual forma, cuando $b = -1$ el ruido adherido al patrón es de tipo substractivo.

En el caso de imágenes con valores en el rango $[0,\ l-1]$, el valor de $c$, sustraído o adicionado a cada píxel será tal que:

$$0 \quad \leq \quad f(x,y) \pm c \leq l - 1 \tag{4.3}$$

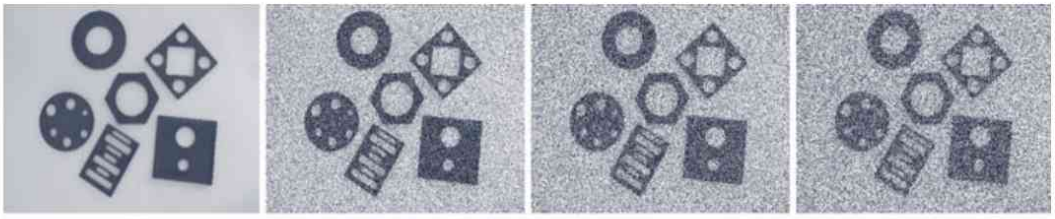
Cuando no se conoce la naturaleza del ruido presente en una imagen, se asume que este sigue un comportamiento de tipo gaussiano. En este caso la variable $c$ toma valores de acuerdo con la siguiente ecuación:

$$c \quad = \quad e^{-\frac{z^2}{2\sigma^2}} \tag{4.4}$$

donde $z \in \mathbb{R}_+ \cup 0$, y $\sigma \in \mathbb{R}_+$ es un parámetro de amplitud gaussiana.

**Ejemplo 4.4**

En la figura 4.5a se muestra una imagen y tres versiones ruidosas con ruido mezclado gaussiano para tres valores de $\sigma$. Note que conforme el valor de $\sigma$ aumenta, el efecto del ruido sobre la imagen incrementa tendiendo a una pérdida cada vez más notoria de la información, tal y como se ilustra en las figuras 4.5b-4.5d, respectivamente.



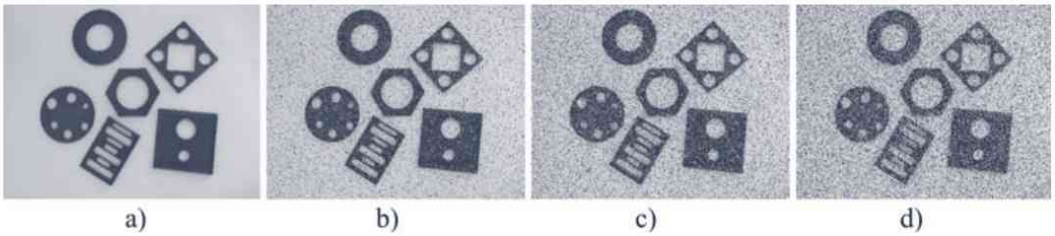
a) b) c) d)

**Figura 4.5**: Imagen con seis objetos a), b)-d) son tres versiones de la misma imagen pero contaminadas con ruido gaussiano, $\sigma = 0.1$, $\sigma = 0.2$ y $\sigma = 0.3$, respectivamente.

Otro tipo de ruido que suele acompañar a las imágenes es el tipo sal y pimienta. Este ruido se caracteriza por saturar el valor de uno o más píxeles de $f(x, y)$ hacia el extremo positivo (ruido tipo sal, $l - 1$) o hacia el extremo inferior (ruido pimienta, 0). Para generar ruido saturado tipo sal y pimienta usando la ecuación (4.2), es suficiente que para cada píxel con coordenadas $(x, y)$, $b = 1$ y $c = 0$ (pimienta) o $c = l - 1$ (sal), con $l$ el valor máximo de saturación que cada píxel puede tomar.

**Ejemplo 4.5**

En las figuras 4.6(b-d) se muestran tres versiones ruidosas con ruido mezclado sal y pimienta de la imagen mostrada en la figura 4.6a, para tres valores del parámetro $l^*$. Al igual que en el caso del ruido gaussiano, note que conforme el valor del parámetro $l^*$ aumenta, más notorio es el efecto del ruido sobre la imagen.



a) b) c) d)

**Figura 4.6**: Imagen con seis objetos (a), y (b-d) tres versiones contaminadas de dicha imagen con ruido sal y pimienta, las cuales corresponden a los casos $l^* = 0.1$, $l^* = 0.2$ y $l^* = 0.3$, respectivamente.

# 4.4 Filtrado de imágenes

El objetivo de las técnicas de filtrado consiste en reducir o eliminar, en el mejor de los casos, el ruido presente en una imagen. Debido a que el proceso de modelado del ruido es algo muy complicado, en un escenario ideal se obtiene una reducción del mismo presente en la señal. En esta sección se estudia un conjunto de técnicas muy importantes para reducir la influencia del ruido en varias ocasiones presente en las imágenes adquiridas a través de la cámara del robot.

Se estudia la operación de filtros de tipo lineal y no lineal en el dominio de los elementos de la imagen de entrada. En particular se analiza la operación del filtro promedio aritmético, promedio gaussiano y mediano.

## 4.4.1 Filtro promedio aritmético

A continuación se define el filtrado promedio aritmético.

**Definición 4.5: Filtro promedio aritmético**

En el dominio de los píxeles, un filtro promedio aritmético $h_{pa}$ es un arreglo de $h \times h$ elementos o pesos $w_1, w_2, \cdots, w_{(h \times h)}$, tales que $w_i > 0$, usualmente es entero.

**• Ejemplo 4.6**

Aplicar el filtro aritmético $h_{pa}$ a una imagen de entrada $f(x, y)$ para obtener una versión modificada $g(x, y)$.

**Solución**

En la figura 4.7 se muestran respectivamente, cuatro filtros de promedio aritmético, dos de tamaño $3 \times 3$ y dos de tamaño $5 \times 5$. Note cómo, en todos casos, los pesos son enteros positivos.

En su aplicación, el filtro promedio aritmético $h_{pa}$ es desplazado a lo largo de la imagen de entrada $f(x, y)$, usualmente, de izquierda a derecha y de arriba abajo, para obtener una versión modificada $g(x, y)$.

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \quad \frac{1}{10}\begin{bmatrix}1&2&1\\2&5&2\\1&2&1\end{bmatrix} \quad \frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix} \quad \frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&2&2&2&1\\1&2&5&2&1\\1&2&2&2&1\\1&1&1&1&1\end{bmatrix}$$

**Figura 4.7**: Cuatro filtros de promedio aritmético, dos de tamaño $3 \times 3$ y dos de tamaño $5 \times 5$.

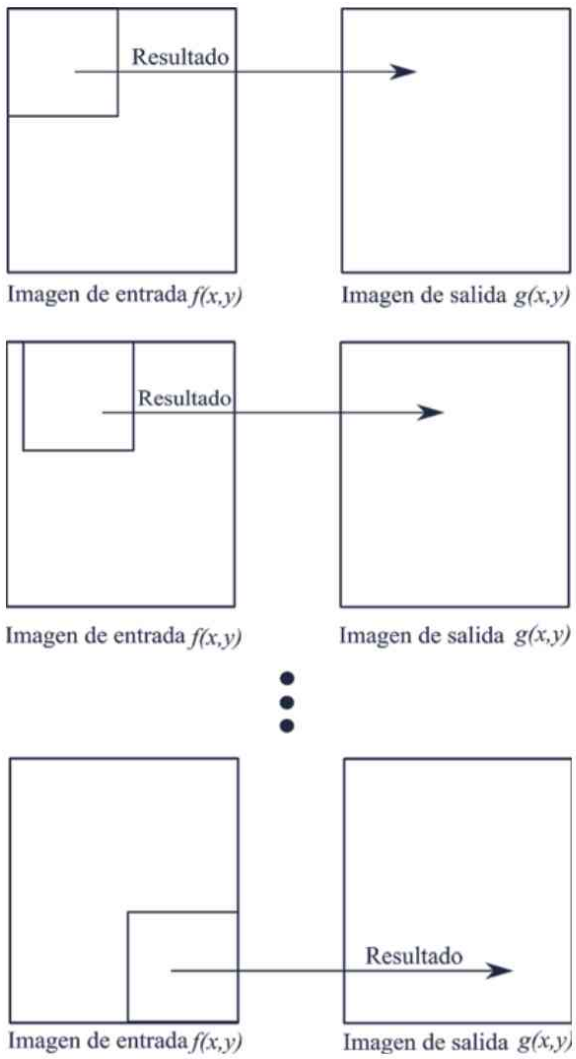



**Figura 4.8:** Filtro promedio aritmético.

La figura 4.8 muestra cómo el filtro se desplaza sobre los diferentes elementos en la imagen de entrada para ir obteniendo sus correspondientes valores de la imagen procesada $g(x, y)$. El resultado de aplicar el filtro aritmético $h_{pa}$ sobre la posición $(x, y)$ de la imagen de entrada $f(x, y)$ es puesto en la correspondiente coordenada cartesiana $(x, y)$ de la imagen de salida $g(x, y)$.

Otra forma de interpretar el resultado de la aplicación del filtro promedio aritmético $h_{pa}$ sobre un píxel con coordenadas $(x, y)$ en la imagen de entrada $f(x, y)$, es que puede ser visto como una suma ponderada entre el vector de pesos $w_1, w_2, \cdots, w_{(h\times h)}$ del filtro $h_{pa}$ y los valores de gris de la imagen de entrada $f_1, f_2, \cdots, f_{(h\times h)}$, dentro de la vecindad de tamaño $h \times h$ del filtro $h_{pa}$, alrededor del píxel $p$ con coordenadas $(x, y)$ en la imagen de entrada $f(x, y)$. Es decir, puede ser expresado como el producto punto o producto escalar.

Entonces, el resultado puede ser descrito de la siguiente manera:

$$r = \Sigma_{i=1}^{h\times h} w_i \cdot f_i. \tag{4.5}$$

Matemáticamente, esta operación puede ser vista como el producto punto escalar o producto interno entre dos vectores de dimensión $h \times h$, el de pesos en la máscara o ventana de filtrado y el de valores de nivel de gris de la imagen de entrada $f(x, y)$ en la posición $(x, y)$, según lo mostrado en la figura 4.8. El resultado será puesto finalmente en la posición $(x, y)$ en la imagen de salida $g(x, y)$ viene dado usualmente, por el valor truncado del promedio de la suma de los $h \times h$ productos $w_i \cdot f_i$, como sigue:

$$g(x,y) = \text{trunc}\left(\frac{1}{h\times h} r\right) = \text{trunc}\left(\frac{1}{h\times h} w_i \cdot f_i\right). \tag{4.6}$$

Como es bien sabido, un filtro de tipo promedio aritmético tiende a suavizar variaciones locales en la señal actuando por tanto, como un filtro pasa-bajas. Como tal, este filtro tiende a desenfocar la imagen, mitigando a la vez los efectos del ruido asociados por lo general, con altas frecuencias en la señal.

**•• Ejemplo 4.7**

Aplicar el filtro aritmético $h_{pa}$ a una imagen de entrada $f(x, y)$ usando las ecuaciones (4.5) y (4.6) sobre la posición marcada en gris en la imagen de tamaño $5 \times 5$ de la figura 4.9a.

**Solución**

En la figura 4.9b se muestra el resultado de la aplicación del primer filtro de la figura 4.7 y las ecuaciones (4.5) y (4.6) sobre la posición marcada en gris en la imagen de tamaño $5 \times 5$ de la figura 4.9a.

El valor 3 que se muestra en la figura 4.9b fue obtenido al aplicar las ecuaciones (4.5) y (4.6) como sigue:

$$\begin{aligned} g(x,y) &= \text{trunc}\left(\tfrac{1}{h\times h} w_i \cdot f_i\right) \\ &= \text{trunc}\left(\frac{1}{9}\left(1\cdot 4+1\cdot 3+1\cdot 3+1\cdot 3+1\cdot 7+1\cdot 2+1\cdot 3+1\cdot 3+1\cdot 3\right)\right) \\ &= \text{trunc}\left(\frac{1}{9}\left(31\right)\right) \\ &= \text{trunc}(3.4444) = 3. \end{aligned}$$

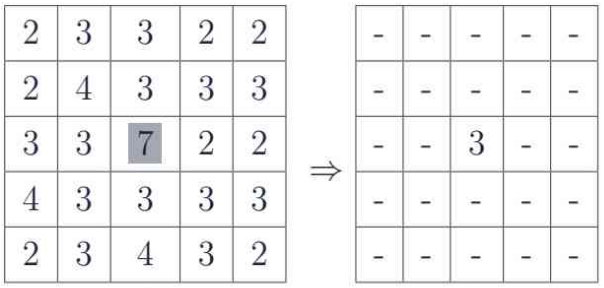



**Figura 4.9**: Aplicación del primer filtro de tamaño $3 \times 3$ de la figura 4.7 sobre el píxel en la posición central la imagen de la izquierda $f(x, y)$. El resultado es puesto en la misma posición, pero en la imagen de salida $g(x, y)$.

Un procedimiento sencillo por filtrado promedio aritmético de una imagen $f(x, y)$ usando una ventana de tamaño $h \times h$ para obtener la imagen filtrada de salida $g(x, y)$, es el que a continuación se presenta:

**Pseudocódigo 4.3**

Algoritmo de filtrado promedio aritmético de imágenes

Procedimiento para filtrar una imagen $f(x, y) \in \mathbb{R}^{n \times m}$ con máscara $h_{pa}(i, j)$ de $h \times h$ píxeles:

Entrada: imagen en niveles de gris $f(x, y) \in \mathbb{R}^{n \times m}$.
Salida: imagen filtrada g(x,y). % $g(x, y) \in \mathbb{R}^{n \times m}$.

g(x,y)=zeros(n, m); % la imagen $g(x, y) \in \mathbb{R}^{n \times m}$ se llena de ceros.
Diseñar máscara $h_{pa}(i, j)$ de tamaño $h \times h$, con valores positivos.

```
for y=h hasta y=(m-1)
    for  x=h hasta x=(n-1)
        v=0;
        for i=1 hasta i=h
            for  j=1 hasta j=h
                v=v+(f(x+i+1-h, y+j+1-h)*h_pa(i,j))/(h*h);
            end
        end
        g(x,y)=trunc(v);
    end
end
```

% fin del procedimiento.

**•• Ejemplo 4.8**

Desarrollar un programa en **MATLAB** que permita filtrar una imagen $f(x, y)$ a través del filtrado promedio aritmético.

**Solución**

En las figuras 4.10b-4.10d se muestra el efecto de aplicar un filtro promedio aritmético con diferentes tamaños de máscara $3 \times 3$, $5 \times 5$, $7 \times 7$, $9 \times 9$ y $11 \times 11$ a la imagen de la figura 4.10a. Nótese cómo entre más grande es la ventana de promedio, más severo es el efecto de atenuación o desenfocado, con la pérdida de detalle que esto conlleva.

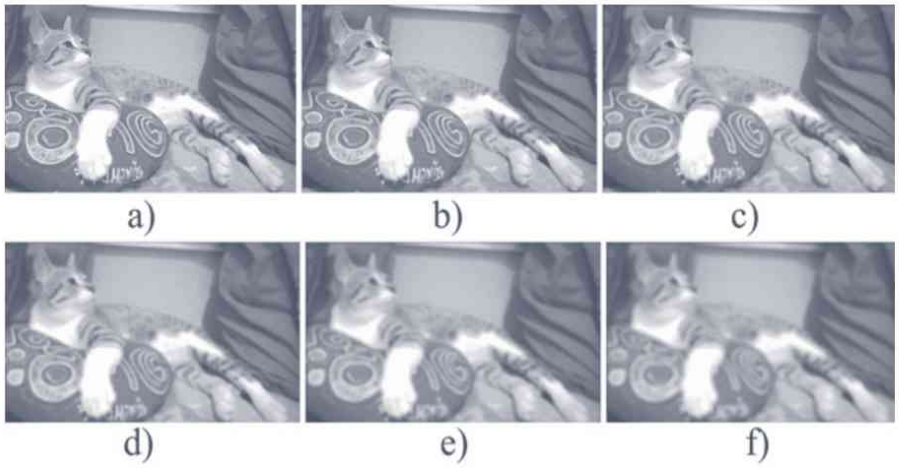



**Figura 4.10**: Efecto de aplicar el filtro de promediado aritmético de varios tamaños sobre una imagen con niveles de gris. a) Imagen sin filtro, b) Imagen filtrada con máscara de dimensión 3×3, c) Imagen filtrada con máscara de 5×5, d) Resultado con máscara de 7×7, e) Resultado con máscara de 9×9 y f) Resultado con ventana de 11×11.

El cuadro de código **MATLAB** 4.2 describe el programa cap4_filtropa.m para implementar el filtro promedio aritmético $h_{pa}$ de acuerdo con el pseudocódigo 4.3. El filtro $h_{pa}$ es aplicado a la imagen $f(x, y)$ sin filtro mostrada en la figura 4.10a. El primer filtro $h_{pa}$ que se utiliza, corresponde al filtro $3 \times 3$ lleno de unos descrito en la figura 4.7, el cual reproduce a la figura 4.10b.

**Código MATLAB 4.2 cap4_filtropa.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 4. Tratamiento de imágenes.
Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**"  .

Programa: cap4_filtropa.m MATLAB versión 2020b

```
clc % limpia pantalla.
clearvars; % remueve todas las variables del espacio actual de trabajo.
close all; % cierra gráficas, archivos y recursos abiertos.
f=imread( 'cap4_GatoManchas.png' ); % lee imagen de prueba f(x,y).
figure(1)
imshow(f) % muestra imagen de prueba (ver figura 4.10a).
title( 'Imagen original' )
[n, m]=size(f); % dimensiones de la imagen f(x,y).
g=zeros(n,m);
hpa=[1, 1, 1;
     1, 1, 1;
     1, 1, 1];
[ h, h]=size(hpa);
for y=h:(m-1)
    for  x=h:(n-1)
        v=0; % se limpia la variable v de cómputo.
        for  i=1:h
            for  j=1:h
                v=v+(f(x+i+1-h, y+j+1-h)*hpa(i,j))/(h*h);
            end
        end
    end
    g(x,y)= uint8(round(v)); % registro del píxel filtrado por promedio aritmético.
end
```

**Continúa código 4.2: cap4_filtropa.m**

```
25 figure(2)
26 imshow(g)
27 title( 'Imagen filtrada' )
28 imagesc(g,[0 255]); colormap(gray);
```

La línea de programación 4 en el cuadro de código **MATLAB** 4.2 del programa cap4_filtropa.m lee la imagen de prueba y la descarga a memoria directamente en la variable $f(x, y)$. Esta imagen se despliega en la línea 6 para obtener la figura 4.10a. En la línea 8 se obtiene la dimensión de dicha imagen y con este resultado se declara la imagen de salida, es decir, la variable $g(x, y)$, la cual se llena de ceros; de esta forma, $g(x, y)$ representa la imagen de salida con la dimensión adecuada, tal y como se ilustra en la línea de programación 9. El filtro promedio aritmético $h_{pa}$ con dimensión $3 \times 3$ lleno de unos, se define en la línea 10. Si bien la dimensión $h \times h$ del filtro $h_{pa}$ es conocida, puesto que el usuario la propone, en la línea 13 se obtiene verificando en este caso que $h = 3$.

El código principal del filtro promedio aritmético $h_{pa}$ se encuentra comprendido entre las líneas 14 y 24. Su aplicación se realiza con desplazamientos de izquierda a derecha y de arriba hacia abajo a lo largo de toda la imagen $f(x, y)$, obteniendo la imagen de salida $g(x, y)$; es decir, la imagen filtrada.

Es muy importante indicar al lector que en el proceso de filtrado, la variable de cómputo $v$ definida en la línea 16 se debe de limpiar o inicializar a cero, cada vez que se ha terminado un barrido completo alrededor de una cierta región del píxel $(x, y)$, es decir, aplicando el filtro $h_{pa}$ en la región $(x + i + 1 - h, y + j + 1 - h)$. Antes de ser limpiada esta la variable $v$, su valor ponderado debe ser registrado en la imagen de salida $g(x, y)$ en las coordenadas $(x, y)$, entonces en el píxel filtrado por promedio aritmético se toma en cuenta únicamente la parte entera del valor computado en $v$; para realizar esto, se utiliza la función round, tal y como se indica en la línea 23.

El conjunto de píxeles filtrados $v$ se van registrando en forma recursiva en la imagen de salida filtrada $g(x, y)$ como enteros positivos de 8 bits, la imagen filtrada es desplegada en la línea 28, cuyo resultado es mostrado directamente en tonos de gris, como se ilustra en la imagen de la figura 4.10b.

El lector puede reproducir los resultados mostrados de las imágenes contenidas en las figuras 4.10c a la 4.10f, tomando ventanas de filtros $h \times h$ cada vez más grandes, como las que se describen en la figura 4.8. Es importante resaltar que entre más grande sea la ventana del filtro de promedio aritmético $h_{pa}$, entonces será más severo el efecto de atenuación o desenfocado; esto significa que se irán perdiendo los detalles de la imagen a medida que la dimensión de la ventana aumenta. Se invita al lector a modificar el código fuente con ventanas más grandes de filtros $h_{pa}$ en el programa cap4_filtropa.m del cuadro de código MATLAB 4.2.

### 4.4.2 Filtro promedio gaussiano

Un tipo de filtrado diferente al promedio aritmético $h_{pa}$, es el denominado filtro promedio gaussiano representado por $h_{pg}$, el cual incluye una forma estructural exponencial de las coordenadas cuadráticas $(x, y)$ de la imagen. El filtro promedio gaussiano se define como:

**Definición 4.6: Filtro promedio gaussiano**

En el dominio de los píxeles, un filtro promedio gaussiano $h_{pg}$ es un arreglo de $h \times h$ elementos o pesos $w_1, w_2, \cdots, w_{(h \times h)}$, tales que $w_i > 0$, son ajustados en términos de:

$$h_{pg} \quad = \quad H(x, y) = c\, e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.7}$$

donde $c$ es una constante de normalización.

Por lo que, de la ecuación (4.7) se puede obtener lo siguiente:

$$\frac{H(x, y)}{c} = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.8}$$

una vez seleccionado el valor de $\sigma^2$ se puede evaluar el núcleo para la ventana $h$.

Dada una imagen $f(x, y)$ de dimensiones $n \times m$ píxeles, un procedimiento sencillo para filtrar a la imagen $f(x, y)$ usando un filtro promedio gaussiano establecido por la ecuación (4.8) y utilizando una máscara de dimensión $h \times h$ píxeles para obtener la correspondiente imagen filtrada $g(x, y)$, es el que se encuentra descrito en el pseudocódigo 4.3 y que a continuación se describe:

**Pseudocódigo 4.3**

Algoritmo de filtrado promedio gaussiano de imágenes

Procedimiento para filtrar una imagen $f(x,y) \in \mathbb{R}^{n\times m}$ con máscara $h_{pg}(i,j) \in \mathbb{R}^{h\times h}$.

Entrada: imagen en niveles de gris $f(x,y) \in \mathbb{R}^{n\times m}$.
Salida: imagen filtrada $g(x,y) \in \mathbb{R}^{n\times m}$.

g(x,y)=zeros(n, m); % la imagen $g(x,y)$ se llena de ceros.
Diseñar máscara $h_{pg}(i,j)$ de tamaño $h \times h$, con valores positivos.

```
for y=h hasta y=(m-1)
    for  x=h hasta x=(n-1)
        v=0;
        for i=1 hasta i=h
            for  j=1 hasta j=h
                v=v+(f(x+i+1-h, y+j+1-h)*h_pg(i,j));
            end
        end
        g(x,y)=trunc(v);
    end
end
% fin del procedimiento.
```

**• • • Ejemplo 4.9**

Considere la imagen $f(x,y)$ mostrada en la figura 4.11a, desarrolle un programa en MATLAB para implementar el filtro promedio gaussiano con $\sigma^2 = 2$ y $h = 7$ y obtenga la correspondiente imagen de salida filtrada $g(x,y)$.

**Solución**

Para obtener el filtro gaussiano con el parámetro $\sigma^2 = 2$ y dimensión de la ventana $7 \times 7$ se procede de la siguiente manera. Considere una ventana de dimensión $7 \times 7$ con los siguientes datos: $x = \{-3,-2,-1,0,1,2,3\}$, así como $y = \{-3,-2,-1,0,1,2,3\}$, entonces los valores de la ventana para el filtro establecido por la ecuación (4.8), es decir: $\frac{H(x,y)}{c} = e^{-\frac{x^2+y^2}{2\sigma^2}}$ son los que se muestran en la tabla 4.1.

La máscara de la tabla 4.1 utilizada de forma directa sobre la imagen $f(x,y)$ de la figura 4.11a reduce la influencia del ruido sobre la imagen, obteniendo la imagen filtrada de salida $g(x,y)$ tal y como se muestra en la figura 4.11b.


**Figura 4.11**: Imagen original a). En la figura b) se muestra la imagen filtrada de salida $g(x, y)$ procesada a través de un filtro promedio gaussiano, usando $\sigma^2 = 2$ y una ventana $7 \times 7$.

**Tabla 4.1**: Filtro gaussiano (4.8), con $\sigma^2 = 2$ y $h = 7$.

| $(x, y)$ | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| -3 | 0.0111 | 0.0388 | 0.0821 | 0.1054 | 0.0821 | 0.0388 | 0.0111 |
| -2 | 0.0388 | 0.1353 | 0.2865 | 0.3679 | 0.2865 | 0.1353 | 0.0388 |
| -1 | 0.0821 | 0.2865 | 0.606 | 0.7788 | 0.606 | 0.2865 | 0.0821 |
| 0 | 0.1054 | 0.3679 | 0.7788 | 1.0 | 0.7788 | 0.3679 | 0.1054 |
| 1 | 0.0821 | 0.2865 | 0.6065 | 0.7788 | 0.6065 | 0.2865 | 0.0821 |
| 2 | 0.3688 | 0.1353 | 0.2865 | 0.3679 | 0.2865 | 0.1353 | 0.0388 |
| 3 | 0.0111 | 0.0388 | 0.0821 | 0.1054 | 0.0821 | 0.0388 | 0.0111 |

El cuadro de código 4.3 contiene la implementación en lenguaje MATLAB del filtro promedio gaussiano, cuya descripción se encuentra en el programa cap4_filtropg.m; en la línea de programación 4 se descarga la imagen de entrada a memoria, almacenándola en la variable $f(x, y)$ y en la línea de programación 6 se exhibe en pantalla, correspondiendo a la imagen que muestra en la figura 4.11a.

Posteriormente en la línea 11 se inicializa la imagen de salida $g(x, y)$ con ceros y también se definen el parámetro $\sigma$ y la dimensión de la ventana.

El código que implementa al filtro promedio gaussiano $e^{-\frac{x^2+y^2}{2\sigma^2}}$, ecuación (4.8) se encuentra desarrollado entre las líneas 10 y 20. Observe que en la línea 15 se invoca a la función $h_{pg}$, la cual se ubica en la línea 27; esta función es la que genera la máscara de la tabla 4.1.

**Código MATLAB 4.3 cap4_filtropg.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 4. Tratamiento de imágenes.
Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap4_filtropg.m MATLAB versión 2020b

```
1  clc % limpia pantalla.
2  clearvars; % remueve todas las variables del espacio actual de trabajo.
3  close all; % cierra gráficas, archivos y recursos abiertos.
4  f=imread( 'cap4_GatoManchas.png' ); % lee imagen de prueba f(x,y).
5  figure(1)
6  imshow(f) % muestra imagen de prueba (ver figura 4.11a).
7  title( 'Imagen original' )
8  [n, m]=size(f); % dimensiones de la imagen f(x,y).
9  g=zeros(n,m); h=7; sigma=sqrt(2); %
10 for y=h:(m-1) % código MATLAB del filtro promedio gaussiano.
11     for x=h:(n-1)
12         v=0;
13         for i=1:h
14             for j=1:h
15                 v=v+(f(x+i+1-h, y+j+1-h)*hpg(i,j, sigma));
16             end
17         end
18         g(x,y)= uint8(round(v));
19     end
20 end
```

**Continúa código 4.3: cap4_filtropg.m**

```
figure(2)
imshow(g) % imagen de salida figura 4.11b.
title( 'Imagen filtrada' )
 % representación de la imagen en tonos de gris.
imagesc(g,[0 255]); colormap(gray);
 % Función filtro gaussiano
 function filtro=hpg(x,y, sigma) % e^-((x^2+y^2)/(2σ^2))
    a=(x^2+y^2)/(2*sigma^2);
    filtro=exp(-a);
end
```

Sin embargo, si se quiere trabajar directamente con valores enteros es necesario primero un rescalamiento de los pesos de la máscara, de forma que el valor más pequeño quede en 1.0. El valor de rescalamiento $k$ se calcula como se indica:

$$k = \text{trunc}\left(\frac{1}{0.0111}\right) = 90$$

la nueva máscara de filtrado gaussiano se presenta en la tabla 4.2, donde $x$ representa los renglones o filas, $y$ las columnas.

Como en el caso del filtro de promedio aritmético descrito en la sección 4.4.1, para que no se presente una saturación en los valores de la imagen resultante al aplicar la máscara obtenida se agrega un factor de normalización obtenido por la suma de todos los pesos del núcleo:

$$c = \sum_{i=-\frac{n}{2}}^{h=\frac{n}{2}} \sum_{k=-\frac{n}{2}}^{k=\frac{n}{2}} H(x_i, y_k) \tag{4.9}$$

para este ejemplo, $c = 1 + 3 + 7 \cdots + 7 + 3 + 1 = 1128$.

En la tabla 4.3 se presenta la normalización de los pesos del núcleo establecidos por la ecuación (4.9).

**Tabla 4.2**: Nueva máscara de filtrado gaussiano.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 3 | 7 | 9 | 7 | 3 | 1 |
| 3 | 12 | 26 | 33 | 26 | 12 | 3 |
| 7 | 26 | 55 | 70 | 55 | 26 | 7 |
| 9 | 33 | 70 | 90 | 70 | 33 | 9 |
| 7 | 26 | 55 | 70 | 55 | 26 | 7 |
| 33 | 12 | 26 | 33 | 26 | 12 | 3 |
| 1 | 3 | 7 | 9 | 7 | 3 | 1 |

**Tabla 4.3**: Normalización de los pesos del núcleo.

$\frac{1}{1128}$

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 3 | 7 | 9 | 7 | 3 | 1 |
| 3 | 12 | 26 | 33 | 26 | 12 | 3 |
| 7 | 26 | 55 | 70 | 55 | 26 | 7 |
| 9 | 33 | 70 | 90 | 70 | 33 | 9 |
| 7 | 26 | 55 | 70 | 55 | 26 | 7 |
| 33 | 12 | 26 | 33 | 26 | 12 | 3 |
| 1 | 3 | 7 | 9 | 7 | 3 | 1 |

Para demostrar en la práctica que la nueva máscara gaussiana establecida en la tabla 4.3, con normalización de pesos es correcta, se puede verificar el mismo resultado de filtrado sobre la imagen de la figura 4.11a (obtenido anteriormente usando el cuadro de código **MATLAB** 4.3 del programa cap4_filtropg.m); con esta finalidad, considere el programa cap4_filtropgA.m descrito en el cuadro 4.4, el cual es una versión modificada del código para el filtro promedio gaussiano.

Observe que en la línea de programación 14 del cuadro 4.4 se encuentra implementada la máscara gaussiana descrita por la tabla 4.3, y por lo tanto, en esta nueva versión del código en **MATLAB** ya no se requiere a la función del filtro promedio gaussiano que tiene el programa cap4_filtropg.m (por ejemplo, ver la línea de programación 27 del cuadro de código 4.3). La imagen de salida $g(x, y)$ (imagen de la figura 4.11b) resulta de aplicar el filtro promedio gaussiano a la imagen de entrada $f(x, y)$ de la figura 4.11a.

En el código comprendido entre las líneas de programación 21 a la 31 se encuentra la implementación del filtro promedio gaussiano, el cual utiliza la máscara de la tabla 4.3 en la línea de programación 26. El resultado corresponde a la imagen de salida $g(x, y)$ de la figura 4.11b.

**Código MATLAB 4.4 cap4_filtropgA.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 4. Tratamiento de imágenes.
Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap4_filtropgA.m MATLAB versión 2020b

```
1  clc % limpia pantalla.
2  clearvars; % remueve todas las variables del espacio actual de trabajo.
3  close all; % cierra gráficas, archivos y recursos abiertos.
4  f=imread( 'cap4_GatoManchas.png' ); % lee imagen de prueba f(x,y).
5  figure(1)
6  imshow(f) % muestra imagen de prueba (ver figura 4.11a).
7  title( 'Imagen original' )
8  [n, m]=size(f); % dimensiones de la imagen f(x,y) ∈ IR^{n×m}.
9  g=zeros(n,m); % se declara la imagen de salida g(x,y) ∈ IR^{n×m}.
10 h=7; % dimensión para la ventana de la máscara del filtro.
11 sigma=sqrt(2); %  % parámetro σ del filtro promedio gaussiano.
12 c=1128; % constante de normalización para la tabla 4.3.
13  % máscara gaussiana establecida en la tabla 4.3.
14 hpg=(1/c)*[1, 3, 7, 9, 7, 3, 1; % máscara de filtrado de la tabla 4.3.
15              3,12, 26, 33, 26, 12, 3;
16              7,26, 55, 70, 55, 26, 7;
17              9, 33, 70, 90, 70, 33, 9;
18              7, 26, 55, 70, 55, 26, 7;
19              33, 12, 26, 33, 26, 12, 3;
20              1, 3, 7, 9, 7, 3, 1];
```

**Continúa código 4.4: cap4_filtropgA.m**

```
for y=h:(m-1) % código MATLAB del filtro promedio gaussiano.
    for x=h:(n-1)
        v=0;
        for i=1:h
            for j=1:h
                v=v+(f(x+i+1-h, y+j+1-h)*hpg(i,j, sigma));
            end
        end
        g(x,y)= uint8(round(v));
    end
end
figure(2)
imshow(g) % imagen de salida figura 4.11b.
title( 'Imagen filtrada' )
 % representación de la imagen en tonos de gris.
imagesc(g,[0 255]); colormap(gray);
```

■■■

**•• Ejemplo 4.10**

Considere la imagen de entrada $f(x, y)$ que se presenta en la figura 4.12a, la cual se encuentra contaminada con ruido gaussiano correspondiente a $\sigma^2$=0.05. Desarrollar un programa en MATLAB que permita filtrar la imagen $f(x, y)$ usando los filtros promedio aritmético $h_{pa}$ y el filtro promedio gaussiano $h_{pg}$ con $\sigma^2 = 1$ y una ventana $h = 5$, para ambos filtros.

**Solución**

Como ejemplo ilustrado del filtrado promedio gaussiano, en la figura 4.12a se muestra una imagen contaminada con ruido gaussiano correspondiente a $\sigma^2$=0.05. En las figuras 4.12b y 4.12c se muestran las imágenes resultantes por utilizar un filtro promedio aritmético de

$5 \times 5$ y una máscara gaussiana también de $5 \times 5$ con $\sigma^2 = 1.0$. Note cómo el filtro gaussiano tiende a reducir la influencia del ruido y, a la vez, a preservar mejor el detalle en la imagen que el filtro promedio aritmético.


**Figura 4.12**: a) Imagen contaminada con ruido gaussiano con $\sigma^2 = 0.05$. (b) Imagen filtrada usando filtro promedio aritmético con ventana $5 \times 5$. (c) Imagen filtrada con un filtro gaussiano de dimensión $5 \times 5$ y $\sigma^2 = 1$.

El programa en MATLAB cap4_programaejemplo10.m contiene el código fuente de este ejemplo y se encuentra disponible en el sitio Web de esta obra.

**Procedimiento para ejecutar el programa del ejemplo 4.10**

- Entrada: imagen en niveles de gris $f(x, y)$ (cap4_GatoManchas.png ).
- Salida: imagen filtrada $g(x, y)$.
- El archivo de la imagen de prueba: cap4_GatoManchas.png, así como el código fuente en MATLAB del programa cap4_programaejemplo10.m se encuentran disponibles en el sitio Web de esta obra.
- Descargar todos los archivos en la misma carpeta, cuya trayectoria en disco duro es definida y seleccionada por el usuario.

### 4.4.3 Filtro mediano

El filtro mediano es uno de los más usados para filtrar imágenes. Siendo del tipo no lineal, básicamente lo que el filtro mediano hace es reemplazar el nivel de gris de la imagen de salida $g(x, y)$ por la mediana de los niveles de gris en una vecindad del píxel bajo consideración en la imagen de entrada $f(x, y)$.

Este método es particularmente útil cuando el ruido contiene valores pico que se salen del promedio, los cuales se quiere eliminar. Solo para recordar, la mediana o valor mediano $m$ de un conjunto de valores es tal que la mitad de los valores en el conjunto es menor que $m$, y la otra mitad es mayor que $m$.

Para aplicar este filtro a un píxel con coordenadas $(x, y)$ hay que: ordenar los valores del píxel y sus vecinos en forma ascendente. Determinar la mediana de este conjunto de valores, y asignar este valor al píxel bajo consideración en la imagen de salida $g(x, y)$. En el caso dos-dimensional, para una vecindad de $3 \times 3$, la mediana es el quinto valor; para una vecindad de $5 \times 5$, la mediana es el treceavo valor, y así sucesivamente. En la tabla 4.4 se ilustra el procedimiento para facilidad de comprensión.

**Tabla 4.4**: Vecindades de $3 \times 3$ y $5 \times 5$, respectivamente. En el primer caso el valor mediano a considerar se encuentra en la quinta posición; en el segundo caso, el valor mediano a considerar es el de la treceava posición.

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |

**• Ejemplo 4.11**

Obtener el valor mediano en la vecindad de $3 \times 3$ de la tabla 4.5.

**Solución**

El valor mediano de los nueve valores en la vecindad de $3 \times 3$, después del ordenamiento de dichos valores es: 3.

**Tabla 4.5**: (a)Vecindad de $3 \times 3$ con valores de niveles de gris desordenados. (b) Secuencia de valores ordenados.

| | | |
|---|---|---|
| 2 | 1 | 2 |
| 6 | 1 | 4 |
| 3 | 5 | 4 |

$\Longrightarrow$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 4 | 4 | 5 | 6 |

De lo anterior, tenemos la siguiente definición.

**Definición 4.7: Filtro mediano**

Un filtro mediano $h_m$ en el espacio de los píxeles, es una función que toma una secuencia desordenada de $h \times h$ valores de intensidad luminosas alrededor de $h$ píxeles coordenadas $(x, y)$ en la imagen de entrada $f(x, y)$:

1) Los ordena.

2) Selecciona de entre ellos el valor mediano.

3) Asigna dicho valor a la misma posición $(x, y)$, en la imagen de salida $g(x, y)$.

Un procedimiento sencillo para filtrar una imagen $f(x, y)$ mediante un filtro mediano a través de una ventana de tamaño $n \times n$ para obtener la imagen filtrada de salida $g(x, y)$ es el siguiente:

**Pseudocódigo 4.4**

Algoritmo de filtrado mediano para imágenes

Procedimiento mediano para filtrar imagen $f(x, y)$ de dimensiones $n \times m$ píxeles con máscara de $n \times n$ píxeles:

Entrada: imagen en niveles de gris $f(x, y)$.

Salida: imagen filtrada $g(x, y)$.

```
Definir tamaño h × h de la máscara de filtrado.
for y=1 and  x=1
  | g(x,y)=0;
end
for y=1:m
  | for x=1:n
  |   | for i=1:h
  |   |   | for j=1:h
  |   |   |   | h(x,y)=f(x+i+1-h,y+j+i-h);
  |   |   | end
  |   | end
  |   | Ordenar de menor a mayor valores dentro de máscara h(x,y);
  |   | Este procedimiento da como resultado una lista ordenada de valores lo(k);
  |   | Seleccionar valor mediano vm como vm=lo(k/2+1);
  |   | g(x,y)=vm;
  | end
end
```

**• • • Ejemplo 4.12**

Considere la imagen $f(x, y)$ mostrada en la figura 4.13a contaminada con ruido sal y pimienta con $l = 0.2$. Realizar un programa en **MATLAB** para filtrar la imagen de entrada $f(x, y)$ a través del filtro mediano.

**Solución**

La función principal del filtro mediano es forzar a puntos con intensidades distintas a ser más como sus vecinos. En la figura 4.13a se muestra una imagen contaminada con ruido sal y pimienta, con $l = 0.2$. En las figuras 4.13b-4.13c se muestra el efecto de la aplicación del filtro mediano con tamaños de $3 \times 3$ y $5 \times 5$, a la imagen ruidosa de la figura 4.13a, pimienta.

En la figuras 4.13d-4.13e se muestra el efecto de aplicar el filtro promedio aritmético de tamaños $3 \times 3$ y $5 \times 5$ sobre la misma imagen. Note, por un lado, cómo al aumentar el tamaño de la ventana del filtro mediana: 1) reduce más el efecto del ruido y 2) la superioridad del mismo filtro mediano, sobre el filtro promedio aritmético, cuando la imagen viene contaminada con ruido sal y pimienta.

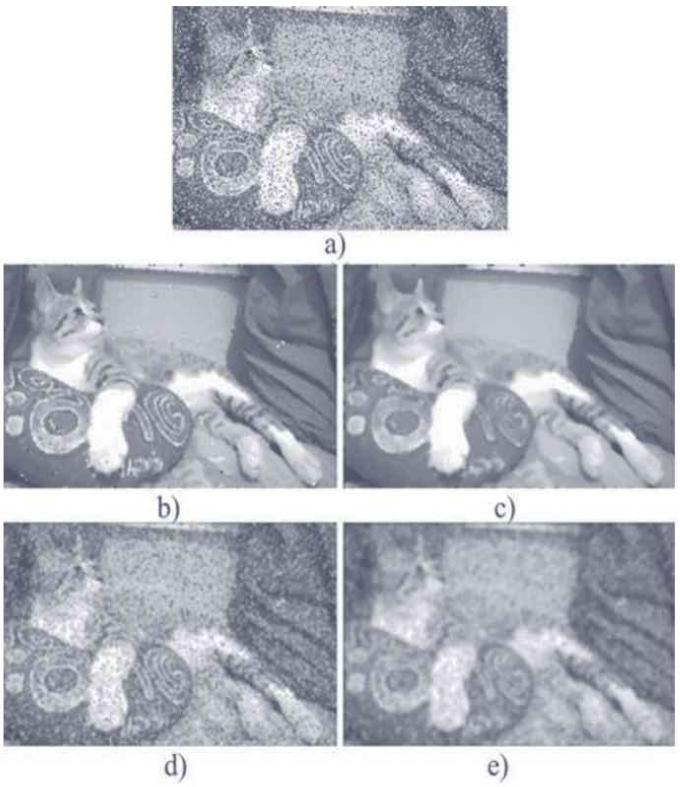



**Figura 4.13**: Efecto de aplicar el filtro mediano sobre una imagen contaminada con ruido sal y pimienta.

La figura 4.13a muestra la imagen contaminada con ruido sal y pimienta con $l = 0.2$.

En la figura 4.13b se presenta la imagen filtrada con un filtro mediano de tamaño $3 \times 3$.

La figura 4.13c contiene la imagen filtrada con un filtro mediano de tamaño $5 \times 5$.

La figura 4.13d ilustra la imagen filtrada con un filtro promedio aritmético $3 \times 3$.

Por último, en la figura 4.13e se muestra la imagen filtrada con un filtro promedio aritmético de tamaño $5 \times 5$.

En el sitio Web de la presenta obra se encuentra el programa cap4_programaejemplo12.m con el código fuente **MATLAB** que resuelve este ejemplo.

**Procedimiento para ejecutar el programa del ejemplo 4.12**

Entrada: imagen en niveles de gris $f(x, y)$ de la figura 4.13a (cap4_GatoManchasRuido.png ).

Salida: imagen filtrada $g(x, y)$.

El archivo de la imagen de prueba: cap4_GatoManchasRuido.png, así como el código fuente en **MATLAB** del programa cap4_programaejemplo12.m se encuentran disponibles en el sitio Web de esta obra.

Descargar todos los archivos en la misma carpeta, cuya trayectoria en disco duro es definida y seleccionada por el usuario.

### 4.4.4 Mejorado de contraste en imágenes

Algunas veces las imágenes adquiridas por los sistemas de visión del robot pueden venir con poco contraste. Esto se ve reflejado en el histograma de la imagen, el cual puede aparecer comprimido, cargado a la izquierda, al centro o a la derecha. En la figura 4.14a se observa una imagen bien contrastada. Su histograma se muestra en la figura 4.14e. Note cómo en este caso, los niveles de gris se encuentran distribuidos a lo largo del eje horizontal del histograma, esto habla de una imagen bien contrastada.

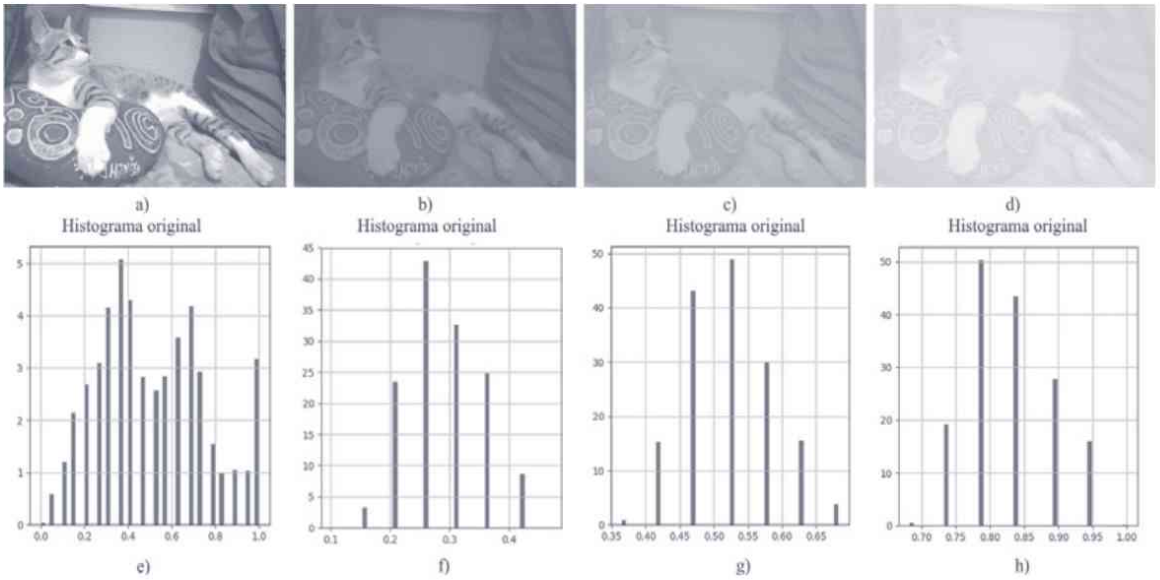



**Figura 4.14**: Imagen bien contrastada (a) y su histograma (e). Versiones (f-e) poco contrastadas de la imagen original (a), los correspondientes histogramas en (f-h).

Por otro lado, en las figuras 4.14c, 4.14d y 4.14e se muestra la misma imagen, pero con mucho menos contraste; en el primer caso cargado a la izquierda, en el segundo cargado al centro y en el tercero, cargado a la derecha. En las figuras 4.14f, 4.14g y 4.14h se muestran los histogramas correspondientes. Note cómo en los tres casos la imagen, como ya se dijo, aparece poco contrastada. En aplicaciones robóticas, al igual que el ruido, el bajo contraste también puede afectar el desempeño del sistema detector o identificador de objetos, ocasionando errores de posicionamiento en el robot.

A continuación se describe una técnica muy usada para mejorar el contraste de una imagen captada poco contrastada. Se trata de la técnica de contrastado por ensanchamiento o apertura.

**Definición 4.8**: **Mejora del contraste lineal**

En el dominio o espacio de los píxeles, un ensanchador o mejora de contraste lineal es una función que toma como entrada el valor de intensidad $v_f$ de cada píxel con coordenadas $(x, y)$ de la imagen de entrada $f(x, y)$, produciendo como salida un valor transformado $v_g$ sobre la imagen de salida $g(x, y)$, a través de la siguiente transformación:

$$g(x,y) \quad = \quad v_g = \text{trunc}\left( (v_f(x,y) - na_{min}) \frac{nd_{max} - nd_{min}}{na_{max} - na_{min}} + nd_{min} \right) \tag{4.10}$$

donde:

- $na_{\min}$ es el nivel de gris mínimo detectado en la imagen o en el histograma.
- $na_{\max}$ es el nivel de gris máximo detectado en la imagen o en el histograma.
- $nd_{\min}$ es el nivel de gris mínimo deseado en la imagen de salida.
- $nd_{\max}$ es el nivel de gris máximo deseado en la imagen de salida.

**Ejemplo 4.13**

Considere la imagen de $8 \times 8$ con 8 niveles de gris, mostrada en la figura 4.15a.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 3 | 4 | 4 | 3 | 4 | 3 | 3 |
| 4 | 6 | 6 | 6 | 5 | 6 | 5 | 3 |
| 4 | 6 | 4 | 4 | 4 | 3 | 6 | 3 |
| 3 | 6 | 4 | 3 | 3 | 4 | 5 | 4 |
| 3 | 5 | 3 | 3 | 3 | 4 | 5 | 3 |
| 4 | 5 | 4 | 3 | 4 | 3 | 6 | 4 |
| 4 | 5 | 6 | 5 | 6 | 5 | 6 | 4 |
| 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 |

$\Rightarrow$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 2 | 0 | 2 | 0 | 0 |
| 2 | 7 | 7 | 7 | 4 | 7 | 4 | 0 |
| 2 | 7 | 2 | 2 | 2 | 0 | 7 | 0 |
| 0 | 7 | 2 | 0 | 0 | 2 | 4 | 2 |
| 0 | 4 | 0 | 0 | 0 | 2 | 4 | 0 |
| 2 | 4 | 2 | 0 | 2 | 0 | 7 | 2 |
| 2 | 4 | 7 | 4 | 7 | 4 | 7 | 2 |
| 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 |

**Figura 4.15**: La imagen de la izquierda de $8 \times 8$ con 8 niveles de gris. La imagen de la derecha ampliada en su contraste a través de la aplicación de la ecuación (4.10).

Al observar esta imagen, el lector puede darse cuenta que $na_{min}$=3 y $na_{max} = 6$. Suponiendo que los valores mínimo y máximos deseados para la imagen de salida son, respectivamente, $nd_{min} = 0$ y $nd_{max} = 7$, uno se puede dar cuenta que la ecuación (3.10) toma la siguiente forma:

$$g(x,y) \quad = \quad v_g = \text{trunc}\left((v_f(x,y) - 3)\,\frac{7-0}{6-3} + 0\right) = \text{trunc}\left(\frac{7}{3}\,(v_f(x,y) - 3)\,.\right) \qquad (4.11)$$

Al aplicar esta ecuación, por ejemplo, al primer píxel con nivel de gris 3 de la imagen de la figura 3.16 (a) se tiene que:

$$v_g(x,y) \quad = \quad \text{trunc}\left(\frac{7}{3}\,(v_f(x,y) - 3)\right) = \text{trunc}\left(\frac{7}{3}(3-3)\right) = 0.$$

De igual forma, al aplicar la ecuación 3.11 al tercer píxel con nivel de gris 4 de la misma imagen, se tiene que:

$$v_g(x,y) = \text{trunc}\left(\frac{7}{3}\,(v_f(x,y) - 3)\right) = \text{trunc}\left(\frac{7}{3}\,(4-3)\right) = 2.$$

Si se procede de esta forma con todo el resto de píxeles de la imagen de la derecha en la figura 4.15, el lector puede verificar que se tiene como resultado la imagen izquierda mostrada de la figura 4.15. Un procedimiento sencillo para modificar el contraste de una imagen $f(x,y)$ para obtener la imagen de salida $g(x,y)$ con otro contraste es la siguiente:

**Pseudocódigo 4.5**

Algoritmo de contrastado para imágenes

Procedimiento contrastado de imagen $f(x,y)$ de dimensiones $n \times m$:

Entrada: imagen en niveles de gris $f(x,y)$.

Salida: imagen $g(x,y)$ con contraste modificado.

Escoger valores deseados $nd_{max}$ y $nd_{min}$;
Recorrer histograma $h(r_i)$ de imagen $f(x,y)$ y encontrar valores $na_{max}$ y $na_{min}$.

```
for y=1:m
    for x=1:n
        v_g=trunc((v_f (x,y)-na_min ) (nd_max-nd_min)/(na_max-na_min )+nd_min );
        h(x,y)=v_g;
    end
end
```

Mostrar imagen contrastada g(x,y);

**• • • Ejemplo 4.14**

Considere la imagen en niveles de gris mostrada en la figura 4.16a. Realice un programa en **MATLAB** para barrer la imagen de la esquina superior izquierda, de izquierda a derecha, hacia abajo hasta la esquina inferior derecha; entonces se aplica la ecuación (4.10) a todos los píxeles de la imagen, de tal forma que se obtenga la imagen con mejor contraste de la figura 4.16b.

**Solución**

Una situación más realista es la imagen en niveles de gris mostrada en la figura 4.16a. Al barrer la imagen de la esquina superior izquierda, de izquierda a derecha, hacia abajo hasta la esquina inferior derecha, uno puede percatarse que $na_{min} = X$ y $na_{max} = Y$. Si se desea que para la imagen de salida de nuevo $nd_{min} = 0$ y $nd_{max} = 255$, entonces se aplica la ecuación (4.10) a todos los píxeles de la imagen 4.16a tal que la imagen con mejor contraste se muestra en la figura 4.16b.

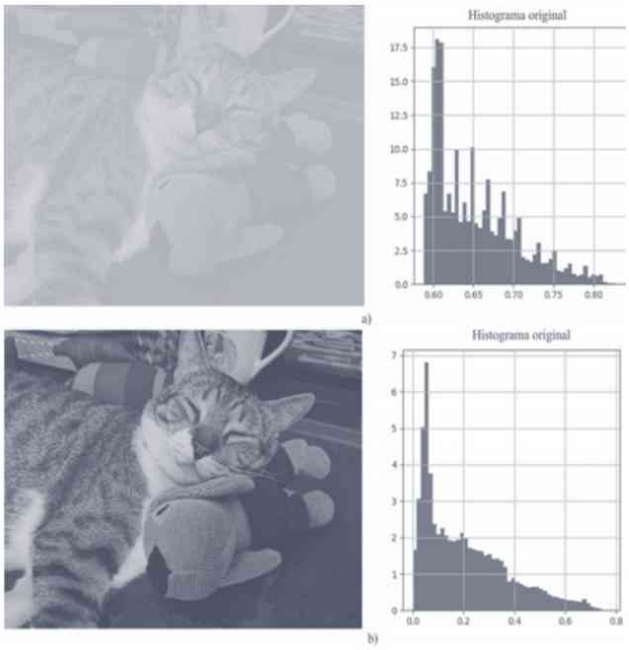



**Figura 4.16**: Imagen (a) poco contrastada de $416 \times 312$ píxeles y 256 niveles de gris con un contraste limitado entre los niveles de 150 a 230 y su histograma. Imagen (b) contrastada por la aplicación de la ecuación (4.10) sobre la imagen (a) y su histograma.

En el sitio Web de la presenta obra se encuentra el programa cap4_programaejemplo14.m con el código fuente MATLAB que resuelve este ejemplo.

**Procedimiento para ejecutar el programa del ejemplo 4.14**

- Entrada: imagen en niveles de gris $f(x, y)$ de la figura 4.16a (cap4_GatoManchasPocoContraste.png).
- Salida: imagen filtrada $g(x, y)$.
- El archivo de la imagen de prueba: cap4_GatoManchasPocoContraste.png, así como el código fuente en MATLAB del programa cap4_programaejemplo14.m se encuentran disponibles en el sitio Web de esta obra.
- Descargar todos los archivos en la misma carpeta, cuya trayectoria en disco duro es definida y seleccionada por el usuario.

## 4.5 Resumen

En este capítulo se describió de manera muy general el problema del tratamiento digital de imágenes para que un robot sea capaz de detectar y determinar la identidad de los diferentes objetos que se pueden encontrar dentro de su espacio de trabajo, para que sea capaz de realizar una tarea útil con ellos.

1. Se presentó un conjunto de conceptos y definiciones útiles para fácil lectura y asimilación del material.
2. Se dieron técnicas para reducir el ruido en imágenes, tales como:

   Filtro promedio aritmético.

Filtro promedio gaussiano.

Filtro mediano.

3. Fueron presentadas técnicas para mejorar su contraste.

## 4.6 Problemas propuestos

En esta sección el lector encontrará una serie de problemas a manera de ejercicio que le permitirán, por un lado, reafirmar el conjunto de conceptos vistos en este capítulo. Por el otro lado, le permitirán aprender nuevos conceptos y a poner en práctica sus habilidades de programación en el manejo de imágenes.

4.6.1 Considere cada una de las imágenes mostradas en la figura 4.17 en niveles de gris $f(x, y)$ (descargar del sito Web de la presente obra las imágenes referidas). Diseñar y poner en operación un programa en MATLAB que permita al usuario obtener su correspondiente histograma.


**Figura 4.17**: Imágenes de prueba para el ejercicio 4.6.1 (descargarlas del sitio Web de esta obra).

4.6.2 Diseñar y poner en operación un programa en MATLAB tal que permita al usuario agregarle ruido gaussiano, dado por la ecuación (4.4) a las imágenes en niveles de gris $f(x, y)$ mostradas en la figura 4.18. Probar con varios valores de $\sigma$.

4.6.3 La figura 4.18 ilustra varias imágenes en niveles de gris f(x,y). Diseñar un programa en MATLAB que permita al usuario agregarle ruido tipo sal y pimienta, dado por la ecuación (4.2). Probar con varios valores de $l$.

4.6.4 En referencia a las imágenes contaminadas con ruido gaussiano que se presentan en la figura 4.18, diseñar un programa en MATLAB que permita aplicar un filtro promedio aritmético (subsección 4.4.1). Probar con varios tamaños de ventana.

4.6.5 Con respecto a las imágenes de la figura 4.18, las cuales se encuentran en niveles de gris $f(x, y)$ y contaminadas con ruido gaussiano, diseñar un programa en MATLAB que permita aplicar un filtro promedio gaussiano, según lo explicado en la subsección 4.4.2. Probar con varios tamaños de ventana y con diversos valores de $\sigma^2$.

4.6.6 Considere nuevamente las imágenes en niveles de gris $f(x, y)$ contaminadas con ruido sal y pimienta con varios valores de $l$ que se muestran en la figura 4.18. Diseñar un programa MATLAB que permita al usuario aplicarle un filtro mediano, según lo explicado en la sección 4.4.2. Probar con varios tamaños de ventana.

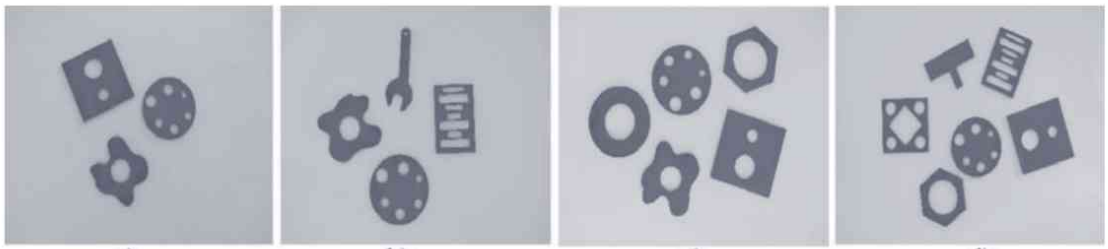



**Figura 4.18**: Imágenes de prueba para los ejercicios 4.6.2 al 4.6.6.

4.6.7 Sea la figura 4.19 que exhibe 3 imágenes en niveles de gris $f(x, y)$ con poco contraste. Diseñar y poner en operación un programa en MATLAB que permita al usuario aplicarle la técnica de mejora de contraste explicada en la subsección 4.4.4.


**Figura 4.19**: a) Imagen con contraste de 40-100; b) Imagen con contraste de 100 a 160 y c) Imagen con contraste de 180-240.

Capítulo 5

# Reconocimiento de objetos

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | **1** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | **1** | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | **1** | 0 | 1 | 1 | 1 | **0** | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Descripción del capítulo

En este capítulo se estudia cómo una vez acondicionada una imagen, esta puede ser segmentada en sus diferentes componentes. Primeramente, se habla acerca de la complejidad del problema y cómo puede ser resuelto mediante el conocido paradigma del umbralado de imágenes. Se describe la técnica de umbralado manual y automático. Se describe la operación de dos métodos muy populares, el de Otsu y de mínimo error. Más adelante, se estudian varios métodos para el postratamiento de imágenes segmentadas para la reducción de ruido residual debido a un pobre umbralado. Enseguida, se estudian varios métodos para el etiquetado de regiones conectadas en imágenes binarias. Posteriormente, se revisan varios rasgos útiles para describir propiedades geométricas y topológicas de los objetos segmentados. En particular se revisa el factor de compacidad, los invariantes geométricos a traslaciones, rotaciones y cambios de tamaño, así como el número de Euler como descriptores de la geometría y estructura topológica de los objetos. Después, se describe un conjunto de técnicas para detectar la presencia de objetos y para su identificación en escenas a partir de imágenes o secuencias de las mismas. Más adelante se describe la operación de varios detectores y clasificadores, entre ellos los basados en el cálculo de distancias y los estadísticos bayesianos. Al final, se presenta un conjunto de problemas propuestos para que el lector ponga en acción sus habilidades y reafirme los diferentes conceptos.

Los siguientes temas son abordados:


Segmentación de imágenes.


Post-filtrado de imágenes.


Descripción de objetos.


Detección y reconocimiento de objetos.

# 5.1 Introducción

PARA que un robot en un ambiente dado pueda realizar una tarea determinada, por ejemplo, navegar entre objetos o manipular uno en específico, es necesario que el robot sea capaz, a través de sus sensores, de detectar e identificar los objetos en dicha escena a partir de una o más imágenes. Una vez que la imagen de entrada ha sido tratada en forma digital, como se explicó en el capítulo anterior, esta debe ser segmentada; esto es, dividida en regiones con un significado lo más cercano al de los diferentes objetos en la escena observada por la cámara. Para lograr esto, en este capítulo se explica la operación de un conjunto de métodos basados en el paradigma de segmentación por umbralado y crecimiento de regiones para el segmentado automático de imágenes.

Debido a que las regiones segmentadas pueden incluir ruido parásito, este debe ser eliminado o reducido para un mejor éxito en etapas posteriores. Este capítulo describe cómo el ruido residual puede reducirse o eliminarse; también se estudia cómo las diferentes regiones segmentadas o aisladas en una imagen pueden ser marcadas con una etiqueta. Esto se hace a través de un método de etiquetado de componentes conectados. Asimismo se revisará la operación de algunos métodos representativos.

Enseguida a cada uno de los objetos segmentados y etiquetados se les calcula un conjunto de rasgos de tipo geométrico o topológico que permiten describir sus propiedades de tipo global o local.

Una vez que los objetos han sido segmentados y descritos en una imagen, estos pueden ser detectados, identificados, incluso ubicados dentro de una imagen y en consecuencia, dentro del espacio de trabajo del robot hacia los diferentes objetos que este eventualmente deba evadir o manipular. Para esto, en este capítulo se describe un conjunto de métodos y técnicas. Primeramente, se describen dos técnicas basadas en el cálculo de distancias, el de distancia mínima o euclidiano y el de Mahalanobis. En segundo lugar, se describe un conjunto de métodos más robustos que los basados en el cálculo de distancias. Se trata de los llamados detectores y clasificadores de tipo bayesiano. En todos los casos se presentan ejemplos para que el lector interesado asimile y afiance más fácilmente las definiciones y conceptos expuestos.

## 5.2 Segmentación de imágenes

Para que un robot sea capaz de detectar y reconocer objetos dentro de su espacio de trabajo requiere aislarlos unos de otros; esto lo debe hacer a partir de las imágenes adquiridas de sus cámaras. A este proceso de aislamiento de objetos en una imagen se le conoce como segmentación de imágenes. La segmentación en una imagen puede definirse como el proceso que consiste en subdividir una imagen en sus regiones, componentes, partes u objetos. Es importante mencionar que la imagen a segmentar probablemente ya fue pre-procesada por medio de alguna de las técnicas de filtrado o mejorado de contraste descritas en el capítulo 4.

En esta sección se mostrará al lector la complejidad del problema de la segmentación de una imagen, después se analizarán dos enfoques para la segmentación: el basado en umbralado y el fundamentado en crecimiento de regiones. Cualquiera que sea la definición que se adopte sobre segmentación, la idea básica consiste en aislar en la imagen los diferentes objetos a reconocer, respecto al fondo y entre ellos mismos.

Está claro que el resultado de este proceso puede determinar el desempeño del sistema completo. Una mala segmentación, en general, provocará errores de detección o reconocimiento de los objetos en la imagen. Una buena segmentación, por el contrario, provocará que el sistema entregue resultados adecuados.

Una manera de definir el proceso de segmentación de acuerdo con [17] es: "La segmentación es el proceso de dividir una imagen $f(x, y)$ en $n$ regiones de píxeles: $R_1,\ R_2, \cdots,\ R_n$, de forma que cada una de estas regiones o sub-imágenes engloben un objeto o una parte del mismo". La segmentación es, pues, el proceso que consiste en agrupar píxeles en regiones: $R_1,\ R_2, \cdots,\ R_n$, tales que:

a) $\cup_{i=1}^{n} R_i \subseteq f(x, y)$, $R_i$, $i = 1, 2, \cdots, n$ es conectada.

b) $R_i \cap R_j = \varnothing$. $\forall\ i, j$ con $i \neq j$.

c) $P(R_i)$ =VERDADERO, $i = 1, 2, \cdots,\ n$. Cada $R_i$ satisface un conjunto de propiedades. Esto significa que para cada elemento en $R_i$ comparten el mismo conjunto de propiedades.

d) $P(R_i \cup R_j) =$ FALSO para $i \neq j$. Esto significa que los píxeles que pertenecen a regiones adyacentes, al ser tomadas como una sola, no satisfacen la propiedad. Si no es el caso, entonces dichas regiones deberán ser consideradas como una sola región.

Dos preguntas que surgen respecto a esta manera de ver el proceso de partición, son:

1) ¿Cómo realizar dicha partición?
2) ¿Qué predicados o conjuntos de propiedades se deben tomar en cuenta para obtener la partición deseada?

Ambas preguntas están fuertemente ligadas. La primera tiene que ver con la selección de un método para obtener la partición, la segunda con las características o propiedades que deberán satisfacer los píxeles de la imagen para ser agrupados en regiones. Para entender esto mejor, consideremos la imagen mostrada en la figura 5.1a. De esta imagen se puede ver que el objeto de interés (el objeto hexagonal con un hueco) tiene una coloración grisácea oscura con tendencia a negro bastante contrastante con respecto a la del fondo (gris claro). Esto sugiere que el nivel de gris de un píxel es una característica suficiente para poder separar al objeto del fondo. En la figura 5.1b se puede apreciar el resultado al colorear de blanco los píxeles del objeto hexagonal con un hueco, respecto a los píxeles del fondo coloreados completamente de negro. Más adelante veremos cómo se llevó a cabo este proceso. Por supuesto, no todas las imágenes satisfacen esta condición y, por tanto, el predicado de segmentación puede constar de varias características, en lo general por la complejidad de la imagen de entrada.

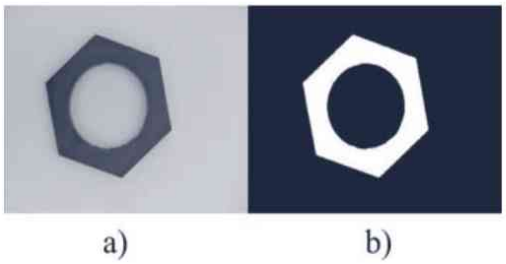



**Figura 5.1**: Imagen (a) es un objeto hexagonal con un hueco contra un fondo contrastante. Imagen (b) corresponde a su segmentación, al usar como característica el nivel de gris de los píxeles de la imagen.

### 5.2.1 El problema de la segmentación de imágenes

No es difícil darse cuenta de que el número de maneras o formas en que una imagen puede ser segmentada tiende a crecer con sus dimensiones. Para apreciar esto, supongamos que la imagen en cuestión tiene $2 \times 2$ píxeles y que los únicos valores que un píxel puede tener es 0 o 1, o sea binario. El número de segmentaciones posibles, al tomar en cuenta el paradigma antes descrito, en este caso es de $\binom{2}{1} + \binom{2}{2} + \cdots + \binom{2}{2} = 16$ combinaciones. Las correspondientes 16 posibles segmentaciones son las siguientes:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \cdots, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Ahora, para el caso general de una imagen de $n \times n$ píxeles, donde sus píxeles pueden tomar uno de $l - 1$ valores (con $l$ cualquier entero positivo $l \geq 2$), el número de posibles segmentaciones crece exponencialmente en concordancia con la expresión: $(l-1)^{(n \times n)}$.

La búsqueda de la solución deseada (la segmentación deseada) crece dramáticamente conforme $n$ crece, indicándonos que, en lo general, el problema de encontrar la segmentación deseada es $np$. Solo en limitadas circunstancias la segmentación automática de una imagen se puede realizar con éxito. La segmentación automática de imágenes es, en general, una de las tareas más difíciles de realizar en análisis de imágenes. Tiene una influencia determinante en el correcto desempeño del sistema de detección o reconocimiento de objetos.

En la literatura especializada se han obtenido más de mil técnicas diferentes para el segmentado de una imagen [10]. El problema, sin embargo, sigue sin tener una solución integral. Como regla general, si se quiere segmentar satisfactoriamente una imagen, se sabe lo que se quiere aislar del fondo. Por ejemplo, si de una imagen de un terreno tomada desde un avión se quisieran aislar los objetos correspondientes a los coches, uno debería primero tratar de segmentar de la imagen las regiones correspondientes a las carreteras o las calles. Una vez hecho esto, se procedería a aislar las regiones correspondientes a los vehículos si es que aparecen en la imagen.

En aplicaciones donde muchas de las condiciones pueden ser controladas, por ejemplo, la iluminación, el que los objetos no se toquen o traslapen entre ellos facilita el proceso de segmentación. En la mayoría de los casos, en navegación autónoma en entornos abiertos, esto no es así. En estos casos, la etapa de segmentación debería ser obviada. En lo que resta

de esta sección se describirán algunas técnicas para la segmentación de imágenes que han demostrado su eficiencia en diversas aplicaciones.

### 5.2.2 Segmentación de imágenes por umbralado

Históricamente, las primeras técnicas para la segmentación de imágenes utilizaban uno o más umbrales. La idea básica consiste en convertir una imagen en niveles de gris, digamos $f(x, y)$ en una imagen binaria $b(x, y)$, tratando de que el objeto u objetos de interés queden separados del fondo de la imagen. Para que cualquiera de estos métodos sea efectivo, se requiere que los objetos de interés presenten suficiente contraste con respecto al fondo y que se conozca el rango de intensidad, ya sea de los objetos o del fondo. Si se supone que el objeto a segmentar es más claro que el fondo, entonces, de acuerdo con [17]:

$$b(x,y) = \begin{cases} l-1 & \text{si } f(x,y) \geq u \\ 0 & \text{en otro caso} \end{cases} \tag{5.1}$$

Si se sabe que el nivel de gris de la superficie del objeto se encuentra en el rango $[u_1, u_2]$, entonces:

$$b(x,y) = \begin{cases} l-1 & \text{si } u_1 \leq f(x,y) \leq u_2 \\ 0 & \text{en otro caso} \end{cases} \tag{5.2}$$

En el caso general de objetos con niveles de intensidad diferentes en sus superficies, se puede usar la siguiente formulación:

$$b(x,y) = \begin{cases} l-1 & \text{si } f(x,y) \in C \\ 0 & \text{en otro caso} \end{cases} \tag{5.3}$$

En este caso, $C$ es el conjunto de valores de intensidad de la superficie de cada objeto.

Con base en lo anterior, tenemos la siguiente definición:

**Definición 5.1: Algoritmo de umbralado**

Un algoritmo de umbralado es una función que recibe como entrada una imagen $f(x, y)$ en $l$ niveles de gris, y a través de un umbral $u$ entrega como salida una imagen en dos niveles de gris $b(x, y)$.

En la presentación de los métodos para la segmentación de imágenes, en el desarrollo de esta sección, se hará uso de la siguiente notación:

- $r$ es el nivel de gris de un píxel con coordenadas $(x, y)$.
- $l$ es el número de niveles de gris de la imagen $f(x, y)$ con $0 \leq r \leq (l-1)$.
- $p_r$ es el número de píxeles con el nivel de gris $r$.
- $n_T = p_0 + p_1 + \cdots + p_r + \cdots + p_{(l-1)}$ es el número total de píxeles.
- $p(r) = \frac{p_r}{n_T}$ es el histograma normalizado de niveles de gris de $f(x, y)$, tal que $\sum_{r=0}^{l-1} p(r) = 1.0$
- $u$ es el umbral a determinar. El objetivo consiste en encontrar el valor óptimo de $u, u^*$ al maximizar o minimizar una función criterio.
- $C_1 = \frac{p_0}{p(C_1)}, \cdots, \frac{p_u}{p(C_1)}$ es el conjunto de píxeles cuyos niveles de gris están en $[0, u]$.
- $C_1 = \frac{p_(u+1)}{p(C_2)}, \cdots, \frac{p_{(l-1)}}{p(C_2)}$ es el conjunto de píxeles cuyos niveles de gris están en $[u+1, l-1]$, donde $p(C_1) = \sum_{r=0}^{u} p_r$, $p(C_2) = \sum_{r=u}^{l-1} p_r$ y $p(C_1)+p(C_2) = 1.0$.

### 5.2.3 Selección manual del umbral

El valor del umbral $u \in [0, l-1]$ para umbralar una imagen $f(x, y)$ puede ser seleccionado en forma manual o automática. En forma manual, el usuario normalmente observa el histograma de la imagen y selecciona un valor de umbral con el cual interactivamente observa el resultado obtenido.

**Ejemplo 5.1**

En las figuras 5.2a y 5.2b se observa una imagen y su correspondiente histograma. En las figuras 5.2c-5.2f se puede apreciar el resultado de aplicar la ecuación (5.1), para varios valores del umbral $u$=70, 100, 130 y 160, a la imagen de la figura 5.2a. Nótese cómo, conforme el valor del umbral $u$ aumenta, en la imagen resultante aparece más información relativa a los objetos. Al mismo se agrega más ruido, como se puede ver en la imagen de la figura 5.2e. Para mejorar el umbral de la imagen de entrada $f(x, y)$, se requiere de un método para encontrar de manera automática el valor de $u$.

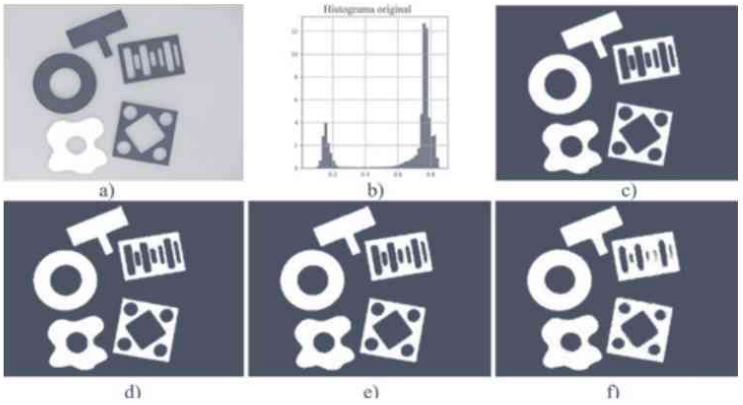



**Figura 5.2**: Imagen (a) con cinco objetos. Su histograma correspondiente (b). Resultados en (c-f) de umbralar la imagen a través de cuatro valores de $u$=70, 100, 130 y 160.

### 5.2.4 Selección automática del umbral

El procedimiento descrito en la sección anterior, para la selección manual del umbral de segmentación, no es útil en el caso general donde las condiciones de iluminación pueden cambiar en cada instante. En este caso, el umbralado automático se hace necesario. En las siguientes secciones se analizará un par de métodos para el umbralado automático de imágenes, esto es para encontrar, a partir de la información de la imagen de entrada $f(x, y)$, el umbral $u$ que mejor segmente a dicha imagen. Para simplificar la presentación de los métodos, se supondrá que los objetos de interés son oscuros contra un fondo claro. Esto permite determinar que los niveles de intensidad con valor por abajo de un umbral $u$ pertenecen al objeto de interés, mientras que los niveles de intensidad por arriba de dicho umbral pertenecen al fondo.

### 5.2.5 Método de Otsu

Este método, aunque es uno de los primeros, continúa usándose en muchas aplicaciones para umbralar automáticamente una imagen. Para su correcto funcionamiento, el método de Otsu [21] supone que los píxeles de la imagen $f(x, y)$ a binarizar pueden ser separados a través de un umbral $u$ (a determinar), en dos clases: $C_1$, la clase del objeto u objetos de interés, y $C_2$ la clase de los píxeles del fondo. El método de Otsu se fundamenta en la técnica del análisis

discriminante, al maximizar alguna medida que permita separar clases: la de los objetos y la del fondo. Una de estas medidas, de acuerdo con el trabajo de Otsu, es la siguiente:

$$J_1(u) = \frac{P_1(u)P_2(u)\left(\mu_1(u)-\mu_2(u)\right)^2}{P_1(u)\sigma_1^2(u)+P_2(u)\sigma_2^2(u)} \tag{5.4}$$

donde:

$$P_1(u) = P_r(C_1) = \sum_{r=0}^{u} p(r) \tag{5.5a}$$

$$P_2(u) = P_r(C_2) = \sum_{r=L-1}^{L-1} p(r) = 1 - P_1(u) \tag{5.5b}$$

$$\mu_1(u) = \sum_{r=0}^{u} rP_r(r|C_1) = \frac{1}{P_1(u)} \sum_{r=0}^{u} rp(r) \tag{5.5c}$$

$$\mu_2(u) = \sum_{r=u+1}^{L-1} rP_r(r|C_2) = \frac{1}{P_2(u)} \sum_{r=u+1}^{L-1} rp(r) \tag{5.5d}$$

$$\sigma_1^2(u) = \sum_{r=0}^{u} \left(r-\mu_1(u)\right)^2 P_r(r|C_1) = \frac{1}{P_1(u)} \sum_{r=0}^{u} \left(r-\mu_1(u)\right)^2 p(r) \tag{5.5e}$$

$$\sigma_2^2(u) = \sum_{r=u+1}^{L-1} \left(r-\mu_2(u)\right)^2 P_r(r|C_2) = \frac{1}{P_2(u)} \sum_{r=u+1}^{L-1} \left(r-\mu_2(u)\right)^2 p(r). \tag{5.5f}$$

Para maximizar el criterio dado por la ecuación (5.4), las medias $\mu_1$ y $\mu_2$ de las dos clases $C_1$ y $C_2$ deberían de estar bastante bien separadas y las varianzas $\sigma_1^2$ y $\sigma_2^2$ deberían de ser lo más pequeñas posibles. Si esto no sucede, el valor del umbral obtenido pudiera no coincidir con el deseado. De igual forma, una imagen con un fondo muy grande, comparado con el objeto u objetos en la imagen, pudiera también dar lugar a valores de umbral que produzcan resultados indeseados.

El valor óptimo $u^*$ puede encontrarse al buscar en el rango $[0,\ l-1]$ el valor de $u$ que maximice la ecuación (5.4). Esto es:

$$u^* = \arg \max_{0 \le u \le l-1} J_1(u). \tag{5.6}$$

Un procedimiento sencillo para segmentar una imagen $f(x,y)$ a través del método de Otsu y obtener una imagen binaria de salida $b(x,y)$ es el siguiente:

**Pseudocódigo 5.1**

Algoritmo de segmentación de imágenes por el método Otsu

Entrada: imagen en niveles de gris $f(x, y)$.
Salida: imagen de salida $b(x, y)$.

```
b(x,y)=zeros(n,m);
for y=1:m
    for x=1:n
        Calcular histograma h(r_i) de imagen f(x,y);
    end
end
Calcular umbral óptimo u* mediante ecuaciones (5.4)-(5.6)) a partir de h(r_i);
for y=1:m
    for x=1:n
        b(x,y) = { l-1   si f(x,y) ≥ u*
                 { 0     en otro caso
    end
end
```

## 5.2.6 Método del mínimo error

En [20] J. Kittler y J. Illingworth proponen el siguiente criterio para determinar el umbral $u$, que permite umbralar una imagen a partir de su histograma:

$$J_{KI}(u) = 1 + 2\,[\,P_1(u)\ln(\sigma_1(u) + P_2(u)\ln(\sigma_2(u))\,] - 2\,[\,P_1(u)\ln(P_1(u) + P_2(u)\ln(P_2(u)\,]. \tag{5.7}$$

En su análisis, Kittler e Illingworth asumen que las densidades de probabilidad, además de que son divididas las clases $C_1$ y $C_2$ a través del umbral $u$, siguen un comportamiento gaussiano conforme a los parámetros dados por las ecuaciones (5.5a) a (5.5f). El umbral óptimo $u^*$ se obtiene, en este caso, como:

$$u^* = \arg \min_{(0 \leq u \leq l-1)} J_{KI}(u). \tag{5.8}$$

**Ejemplo 5.2**

En la figura 5.3 se muestra la segmentación de imágenes. En la primera columna (figuras 5.3a-5.3d) se observan cuatro imágenes con un número variable de objetos: 3, 4, 5 y 6, respectivamente. Los cuatro valores de umbral $u$, calculados a través de las ecuaciones (5.5a) y (5.5f), fueron, respectivamente: 121, 116, 119 y 121.

Las figuras 5.3e- 5.3h muestran las versiones binarias correspondientes. Note que las imágenes fueron correctamente umbraladas, según lo esperado. Esto se debió a que el histograma de las cuatro imágenes es bimodal. La ventaja principal del método de Otsu es que no hace ninguna suposición acerca de las densidades $P_1(u)$ y $P_2(u)$. Como se puede ver en (5.4) a (5.6), este método asume que dichas densidades pueden ser descritas únicamente en términos de sus medias y varianzas y en lo general, no necesariamente es cierto.

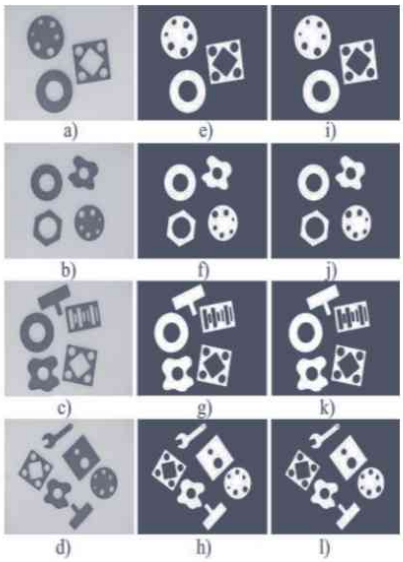



**Figura 5.3**: (a-d) cinco imágenes; (e-h) versiones binarias a través del método de segmentación de Otsu; y $(i-l)$ versiones binarias a través del método de Kittler e Illingworth.

Una de las principales desventajas de este método es la suposición de que el histograma de la imagen es bimodal, esto es, que los píxeles de la imagen pueden ser clasificados en solo dos clases. Para más de dos clases de píxeles en la imagen, el método debe ser modificado de manera que varios umbrales puedan ser definidos de tal forma que permitan maximizar la varianza dentro de la clase y minimizar la varianza entre clases.

**Ejemplo 5.3**

Para las imágenes de la figura 5.3a-5.3d, los cuatro valores de umbral calculados a través de las ecuaciones (5.7) y (5.8) fueron, respectivamente: 70, 72, 65 y 69. En las figuras 5.3i-5.3l se muestran las versiones binarias correspondientes.

Una de las desventajas de la formulación dada por (5.7) es el hecho de que, en muchas ocasiones, los valores de $\ln(P_1(u))$, $\ln(P_2(u))$, $\ln(\sigma_1(u))$ y $\ln(\sigma_2(u))$ pueden ser muy pequeños, incluso de cero. Esto redundará en problemas en el cálculo de uno o más de los términos de $J_{KI}(u)$.

Una manera de reducir este problema es alisar la información dada por el histograma de la imagen antes de los cálculos. Esto se puede lograr al usar el siguiente filtro, para un $p_r$, el valor filtrado se puede obtener como:

$$p_r^{'} = \frac{p_{r-2} + p_{r-1} + p_r + p_{r+1} + p_{r+2}}{b} \tag{5.9}$$

con $0 < b < 5$ se puede ver que cuando $b = 5$, el filtro actúa como un filtro paso-bajo clásico. Cuando el filtro $0 < b < 5$ tiende a enfatizar la información del histograma.

## 5.3 Postfiltrado imágenes segmentadas

En ocasiones, después de que una imagen ha sido umbralada, esta requiere que se le apliquen dos procesos posteriores. Uno de ellos consiste en un postfiltrado, con el objetivo de reducir o eliminar la influencia del ruido causado por un pobre umbralado. El otro consiste en etiquetar las regiones de píxeles aisladas y separadas en la imagen resultante del proceso umbralado.

En esta sección se estudiará, por un lado, un conjunto de técnicas para la eliminación de la imagen de pequeñas regiones conectadas en el fondo y que no son objetos. Por otro lado, se revisará también un conjunto de métodos para la eliminación de hoyos parásitos en la imagen umbralada.

Finalmente, se revisará la operación de una técnica para el etiquetado de componentes conectados de píxeles. Recordemos que a través del proceso de umbralado, todos los píxeles de la imagen umbralada $b(x, y)$ aparecen etiquetados como "0s" o como "1s". Si en la imagen umbralada aparecen varias regiones aisladas entre ellas, es necesario que a cada una se le asigne una etiqueta numérica diferente, con el fin de asignar rasgos descriptivos por región de manera individual para su posterior detección o identificación.

### 5.3.1 Reducción de ruido en imágenes segmentadas

Como ya se dijo al comienzo de esta sección, en ocasiones la imagen umbralada puede venir acompañada de regiones parásitas muy pequeñas resultado de un deficiente umbralado. De igual forma, las regiones separadas del fondo a través del proceso de umbralado pueden venir acompañadas de pequeños huecos, causa también del deficiente umbralado.

En ambos casos, el ruido remanente debe ser eliminado de la imagen umbralada o al menos, reducirlo al máximo con el fin de maximizar el desempeño del sistema de detección o identificación de objetos en el entorno del robot.

**Ejemplo 5.4**

En la figura 5.4a se muestra una imagen $b(x, y)$ de $10 \times 10$ píxeles, la cual se supone fue umbralada a dos niveles de gris, como se describió en la sección 5.2. Suponga que los "1s" y los "0s" marcados en negritas y subrayado en la imagen son ruido causado por un deficiente umbralado y deben ser eliminados de la imagen $b(x, y)$ de alguna manera.

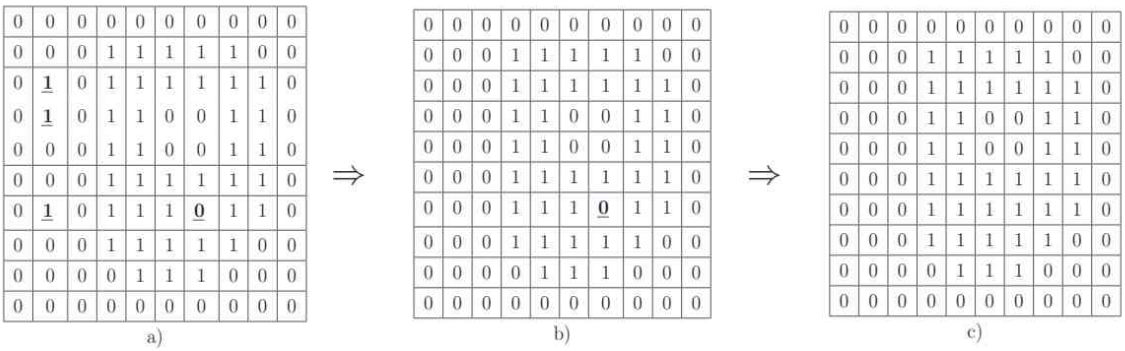



**Figura 5.4**: La imagen binaria 5.4a contiene una región conectada con un hueco parásito y dos pequeñas regiones parásitas. Imagen 5.4b después de aplicarle el método de eliminación de pequeñas componentes parásitas con un $T = 3$. Note cómo los "1" no deseados son transformados en "0s". Imagen 5.4c después de aplicarle el método de eliminación de hoyos parásitos con un $T = 2$. Note cómo los "0s" no deseados son transformados en "1s".

### 5.3.2 Método heurístico para eliminar regiones

Un método heurístico para la eliminación de pequeñas regiones parásitas aisladas, puede ser el siguiente:

- Seleccione un valor $T$ como variable de tamaño de las regiones a eliminar.
- Para cada región conectada en la imagen $b(x, y)$, calcule su tamaño $t$ en píxeles y si este tamaño $t < T$, entonces elimine de la imagen dicha región conectada.

Al aplicar el método antes descrito a la imagen de la figura 5.4a, se obtiene la imagen menos ruidosa mostrada en la figura 5.4b.

Una manera de eliminar los pequeños huecos parásitos en la imagen de la figura 5.4b es a través del siguiente método heurístico:

1) Seleccione un valor $T$ como variable de tamaño de los hoyos a eliminar.
2) Para cada hoyo en la imagen $b(x, y)$, calcule su tamaño $t$ en píxeles y si este tamaño $t < T$, entonces elimine de la imagen dicho hueco.

Al aplicar el método arriba descrito a la imagen de la figura 5.4b se obtiene la imagen menos ruidosa mostrada en la figura 5.4c.

**Nota:** En el caso de los dos métodos descritos anteriormente, la selección de valor $T$ es crítica, si el valor de $T$ seleccionado es menor al ideal, algunas regiones u hoyos no serán eliminados de la imagen de entrada. Por otro lado, si el valor de $T$ es mayor, algunos huecos de la imagen, que no deberán de ser eliminados, podrían serlo.

### 5.3.3 Etiquetado de componentes conectados

Una vez que una imagen $f(x, y)$ ha sido umbralada a través de alguno de los métodos descritos y que el ruido ha sido eliminado de la imagen $b(x, y)$, sus componentes conectados pueden ser etiquetados. Los píxeles en una región conectada forman una región que puede representar un objeto dado a reconocer.

El etiquetado de componentes conectados (ECC) puede ser visto como una segunda etapa en el proceso de segmentado de una imagen. Un algoritmo de ECC es secuencial por naturaleza, debido a que la operación de encontrar componentes conectados (CC) es de tipo global. Si se sabe que en la imagen hay un solo objeto, entonces el algoritmo de etiquetado de CC puede ser innecesario. Por el contrario, si en la imagen hay varios objetos y si su número, posición y orientación requieren ser obtenidos, entonces un algoritmo de etiquetado CC es necesario.

**Definición 5.2: Algoritmo de etiquetado**

Un algoritmo de etiquetado de componentes conectados en una función recibe una imagen de umbralada $b(x, y)$ y produce, a través de un conjunto de reglas, una imagen de salida $e(x, y)$ con regiones etiquetadas, donde cada $n$-ésima etiqueta pertenece a $N$.

Un algoritmo como el descrito en la definición anterior encuentra todos los CC en la imagen y origina una etiqueta única para todos los píxeles en un mismo componente.

**Ejemplo 5.5**

La figura 5.5 muestra una imagen muy sencilla y sus correspondientes CC etiquetados.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

a)

$\Rightarrow$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 3 | 3 | 3 | 0 | 0 | 4 | 4 | 4 | 0 | 0 |
| 0 | 3 | 3 | 3 | 3 | 0 | 4 | 4 | 4 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

b)

**Figura 5.5**: Imagen binaria (a) y sus respectivos CC etiquetados (b).

Hay varios tipos de algoritmos para etiquetado CC. Enseguida se describe uno de los métodos más representativos.

### 5.3.4 Método de dos pasadas que utiliza TE

Este algoritmo realiza dos pasadas sobre la imagen de entrada $f(x, y)$ y una tabla de equivalencias TE para etiquetar una imagen dada de entrada. Durante la primera pasada este método asigna etiquetas a las regiones de acuerdo con el tipo de conectividad seleccionado.

Cuando se presente la situación de asignar dos etiquetas diferentes al mismo píxel $p$, la etiqueta más pequeña le es atribuida, almacenando en TE la equivalencia entre ambas etiquetas. Después de la primera pasada, las clases de equivalencia son determinadas mediante la cerradura transitiva sobre el conjunto de etiquetas almacenadas en la tabla (para más detalles, referirse a [17]).

Sin pérdida de generalidad, supóngase el caso de 4-conectividad. Además, también que la imagen es barrida píxel a píxel, de arriba abajo y de izquierda a derecha. Sea $p$ un píxel de la imagen de entrada en cualquier etapa del barrido y sean $a$ e $i$, respectivamente, los píxeles arriba y a la izquierda de $p$ como se ilustra en la figura 5.6.

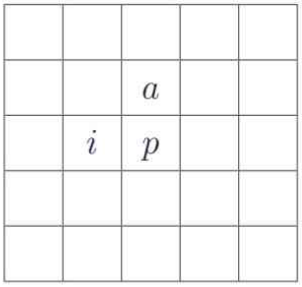



**Figura 5.6**: Píxel $p$ analizado con base en sus vecinos arriba $(a)$ y a su izquierda $(i)$ en una vecindad tipo 4.

La naturaleza de un barrido como el descrito asegura que, al llegar a $p$, $a$ e $i$ ya han sido visitados si son píxeles de tipo objeto. En forma más detallada, el método secuencial de dos pasadas procede como sigue:

1. Barrer la imagen de arriba abajo y de izquierda a derecha:

   a) Si $p$ es un píxel de fondo, esto es $p = 0$, desplazarse a la siguiente posición.

   b) Si $p$ es un píxel de objeto, esto es $p = 1$, entonces:

i) Si solo $a$ o $i$ es de objeto, entonces asignar a $p$ la etiqueta de dicho píxel.

ii) Si ambos, $a$ e $i$, son de tipo objeto y poseen la misma etiqueta, entonces asignar a $p$ la etiqueta de cualquiera de ellos.

iii) Si ambos, $a$ e $i$, son de tipo objeto pero con etiquetas diferentes, entonces asignar a $p$ la etiqueta más pequeña y almacenar en la tabla de equivalencia TE el hecho de que estas dos etiquetas son equivalentes.

iv) De otra manera, asignar a $p$ una nueva etiqueta y ponerla en TE.

2. Si hay más píxeles en la imagen, entonces regresar al paso 1.

3. Resolver la tabla de equivalencias TE.

4. Encontrar la etiqueta más pequeña para cada conjunto de etiquetas en TE.

5. Barrer de nuevo la imagen y reemplazar cada etiqueta por la más pequeña en su conjunto de equivalencia.

**Ejemplo 5.6**

La figura 5.7a muestra una imagen binaria de $6 \times 6$ píxeles, resultado de haber umbralado una imagen en niveles de gris a través de alguno de los métodos descritos en la sección 5.2. La imagen no es muy complicada, pero ayuda a entender la aplicación del método secuencial antes descrito.

En las figuras 5.7b-5.7j se muestra la aplicación de los pasos del método descrito en el etiquetado de los tres componentes de la figura 5.7a; cada uno de los pasos sobre esta imagen es enseguida explicado:

1. En la imagen 5.7b se muestra en negritas el primer píxel con valor "1" etiquetado por el método al aplicar sobre él la regla b)-iv). Al mismo tiempo, su etiqueta de "1" es almacenada en TE; ver la figura 5.8a.

2. En la figura 5.7c se muestra de nuevo en negritas el segundo píxel con valor “1”, etiquetado por la aplicación sobre él las reglas b)-iv). También, su etiqueta de “2” es almacenada en TE; ver la figura 5.8b.

3. En la figura 5.7d se muestra el tercer píxel etiquetado por la aplicación, ahora con las reglas b)-i).

4. En la figura 5.7e) se muestra el cuarto píxel etiquetado por la aplicación con las reglas b)-i).

5. En la figura 5.7f) se muestra el quinto píxel etiquetado por la aplicación sobre él de las reglas b)-i).

6. En la figura 5.7g) se muestra el sexto píxel etiquetado por la aplicación, ahora con las reglas b)-iv). También, su etiqueta de “3” es almacenada en TE; ver la figura 5.8c.

7. En la figura 5.7h se muestra el séptimo píxel etiquetado por la aplicación con las reglas b)-iv). También, su etiqueta de “4” es almacenada en TE; ver la figura 5.8d.

8. En la figura 5.7i se muestra el octavo píxel etiquetado por la aplicación con las reglas b)-iii). También, la equivalencia entre las etiquetas “2” y “4” es almacenada en TE; ver la figura 5.8e.

9. Se continúa barriendo la imagen y como ya no hay píxeles que etiquetar, se procede con el paso 3 del método que resuelve las equivalencia en TE; ver la figura 5.8f.

10. Una vez resueltas las equivalencias se procede con el paso 4 del método descrito que entrega las etiquetas más pequeñas para cada conjunto de las mismas, para nuestro ejemplo: min(1,1)=1, min(2,4)=2 y min(3,3)=3.

11. Finalmente, se procede con el paso 5 del método y se barre de nuevo la imagen (en este caso la imagen de la figura 5.7i) y se reemplaza cada etiqueta por la más pequeña en su conjunto de equivalencia. Como se puede apreciar, el resultado que aparece plasmado en la figura 5.7j.

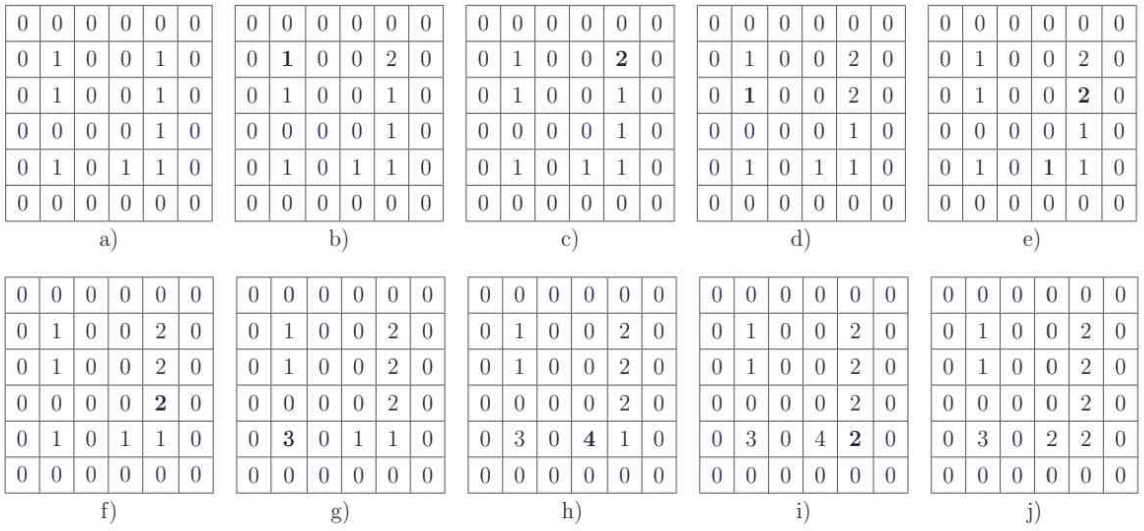



**Figura 5.7**: Imagen (a) de $6 \times 6$. Imágenes (b)-(j) aplicación de los diferentes pasos del método de etiquetado secuencial de dos pasadas anteriormente descrito.

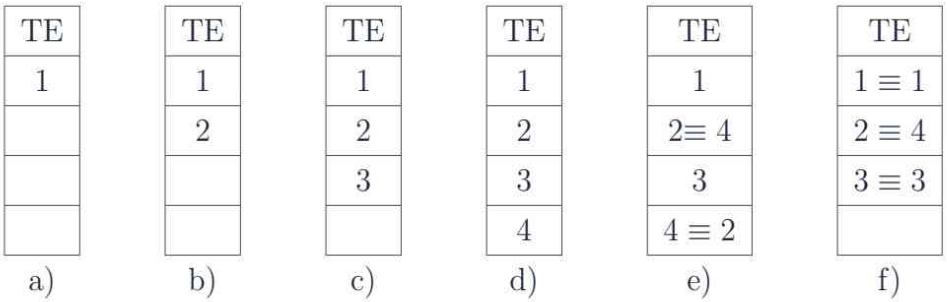
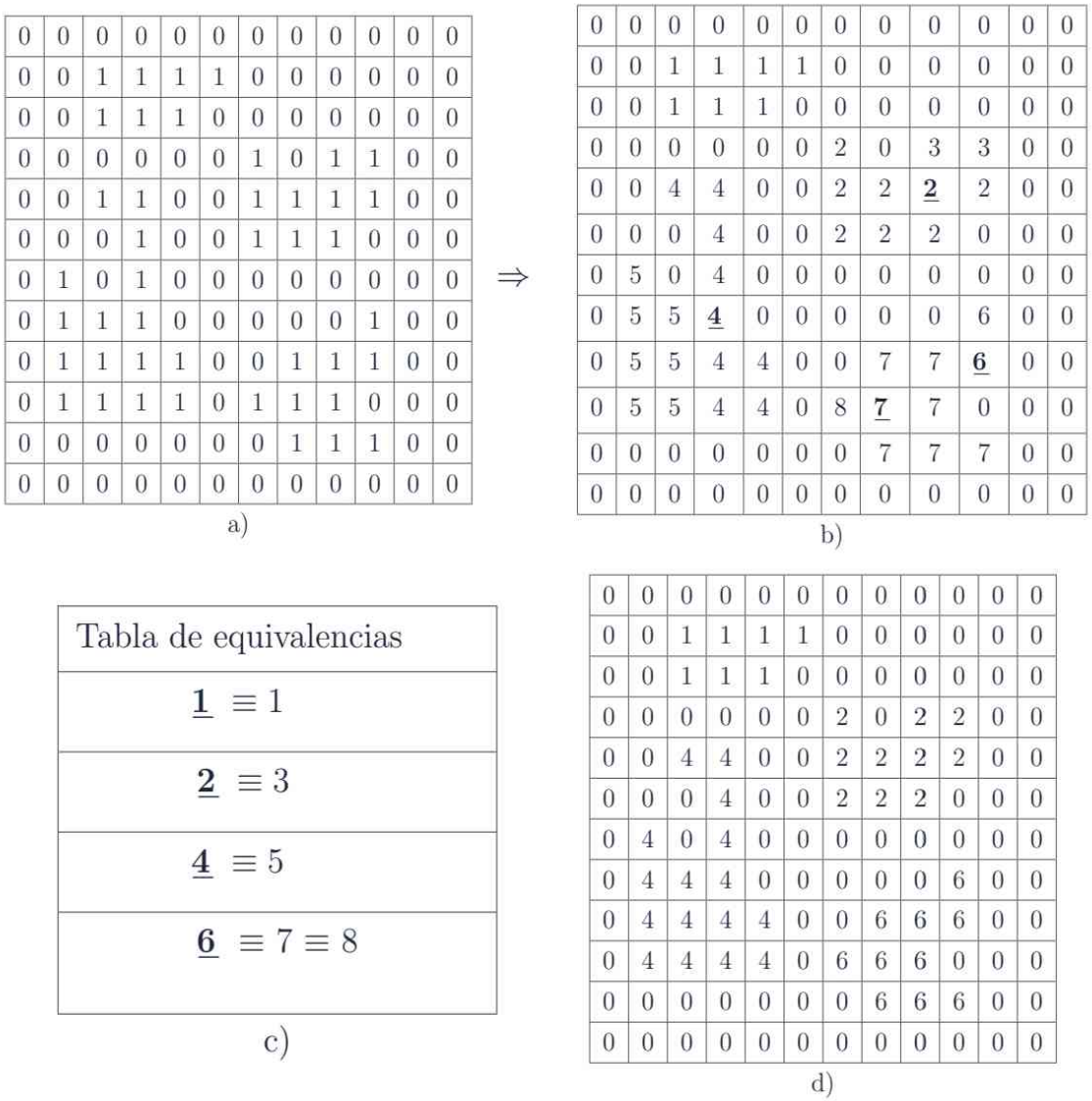



**Figura 5.8**: Imágenes (a)-(d) almacenamiento de las etiquetas asignadas a las regiones encontradas en la aplicación de los pasos del algoritmo secuencial de dos pasadas. Imagen (e) almacenamiento de la equivalencia entre etiquetas. Imagen (f) resolución de equivalencias.

**Ejemplo 5.7**

En la figura 5.9a se muestra una imagen de $12 \times 12$ píxeles con 4 regiones conectadas. En las figuras 5.9b, 5.9c y 5.9d se muestran, respectivamente, el resultado de la primera pasada del método de etiquetado de componentes conectados antes descrito (pasos 1 a 2), la tabla de equivalencias (paso 3) y el resultado final de la segunda pasada (pasos 4 y 5).

Se puede realizar una pasada extra sobre la imagen con el fin de hacer que las etiquetas asignadas sigan un orden secuencial.



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

a)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 | 2 | 2 | **2** | 2 | 0 | 0 |
| 0 | 0 | 0 | 4 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 5 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 5 | **4** | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 |
| 0 | 5 | 5 | 4 | 4 | 0 | 0 | 7 | 7 | **6** | 0 | 0 |
| 0 | 5 | 5 | 4 | 4 | 0 | 8 | **7** | 7 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

b)

| Tabla de equivalencias |
|---|
| **1** $\equiv 1$ |
| **2** $\equiv 3$ |
| **4** $\equiv 5$ |
| **6** $\equiv 7 \equiv 8$ |

c)

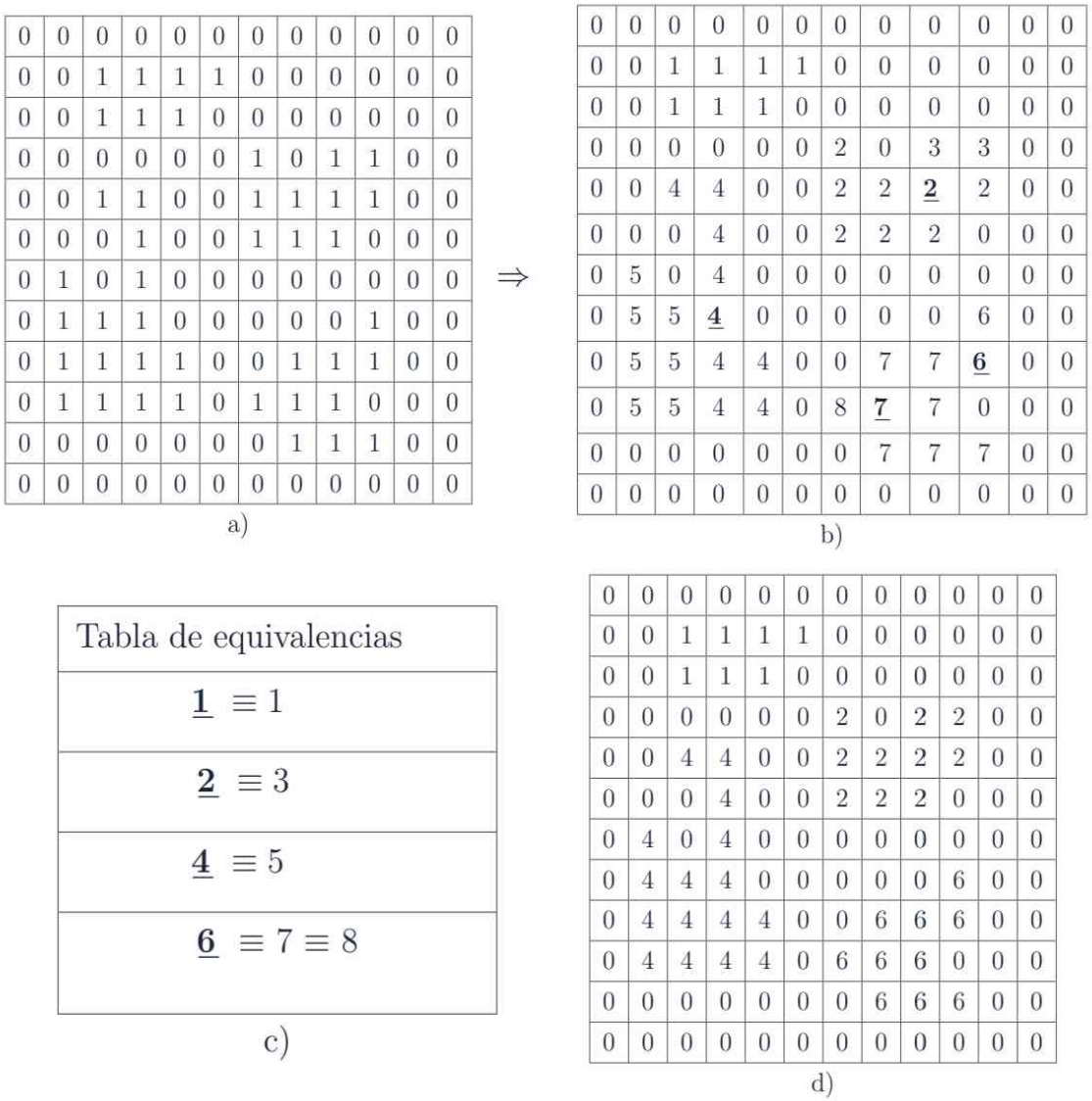
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 4 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 |
| 0 | 4 | 4 | 4 | 4 | 0 | 0 | 6 | 6 | 6 | 0 | 0 |
| 0 | 4 | 4 | 4 | 4 | 0 | 6 | 6 | 6 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 | 6 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

d)

**Figura 5.9**: Una imagen binaria e imágenes (b)-(d) resultados parciales del etiquetado.

La técnica de umbralado y el método de etiquetado de componentes conectados pueden ser combinados para separar las regiones de cada uno de los objetos en una imagen. Cada una de las regiones etiquetadas puede ser, entonces, descrita en términos de rasgos; a su vez estos pueden usarse para formar descripciones como las expresadas por la ecuación (5.1).

Se puede realizar una pasada extra sobre la imagen con el fin de hacer que las etiquetas asignadas sigan un orden secuencial.



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

a)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 3 | 3 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 | 2 | 2 | **2** | 2 | 0 | 0 |
| 0 | 0 | 0 | 4 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 5 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 5 | **4** | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 |
| 0 | 5 | 5 | 4 | 4 | 0 | 0 | 7 | 7 | **6** | 0 | 0 |
| 0 | 5 | 5 | 4 | 4 | 0 | 8 | **7** | 7 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

b)

| Tabla de equivalencias |
|---|
| **1** $\equiv 1$ |
| **2** $\equiv 3$ |
| **4** $\equiv 5$ |
| **6** $\equiv 7 \equiv 8$ |

c)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 0 |
| 0 | 0 | 0 | 4 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 0 | 4 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 |
| 0 | 4 | 4 | 4 | 4 | 0 | 0 | 6 | 6 | 6 | 0 | 0 |
| 0 | 4 | 4 | 4 | 4 | 0 | 6 | 6 | 6 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 | 6 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

d)

**Figura 5.9**: Una imagen binaria e imágenes (b)-(d) resultados parciales del etiquetado.

La técnica de umbralado y el método de etiquetado de componentes conectados pueden ser combinados para separar las regiones de cada uno de los objetos en una imagen. Cada una de las regiones etiquetadas puede ser, entonces, descrita en términos de rasgos; a su vez estos pueden usarse para formar descripciones como las expresadas por la ecuación (5.1).

Un aspecto importante es que estas descripciones pueden finalmente ser utilizadas para entrenar clasificadores que reconozcan los objetos correspondientes a partir de imágenes de los mismos.

A continuación se presenta otro ejemplo ilustrativo.

**Ejemplo 5.8**

En las figuras 5.10a y 5.10b se muestran dos imágenes binarias con 3 y 5 objetos, respectivamente. En las figuras 5.10c y 5.10d se muestran las regiones etiquetadas a través del método secuencial descrito; las regiones aparecen coloreadas con otros tonos de gris para enfatizar la diferenciación entre etiquetas asignadas a las regiones.

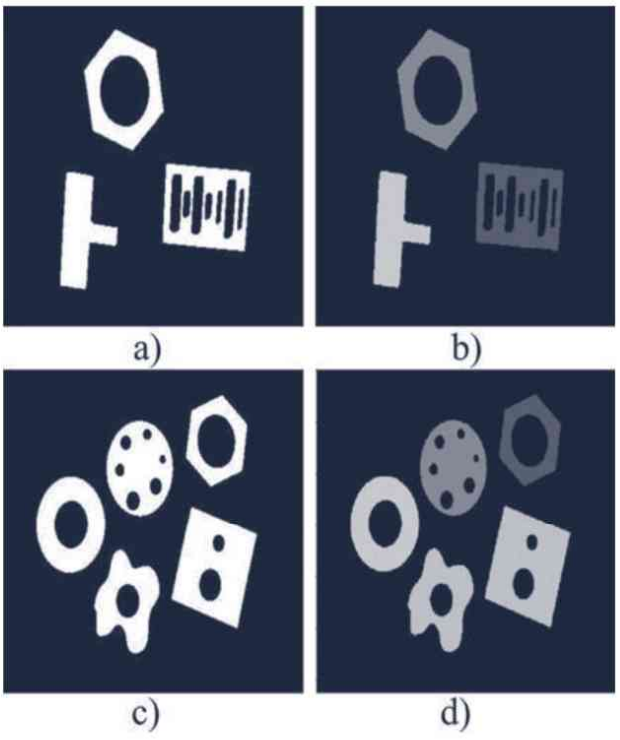



**Figura 5.10**: Dos imágenes binarias (a) y (b), una con tres objetos y otra con cinco. Figuras (c) y (d) con sus correspondientes imágenes etiquetadas y enfatizados en color por la aplicación del método secuencial descrito en esta sección.

# 5.4 Descripción de objetos

Una vez que los diferentes objetos en una imagen han sido segmentados a través de un mecanismo de umbralado, como ya se describió en la sección 5.2, y se etiquetaron mediante un procedimiento de etiquetado de componentes conectados, explicado en la sección 5.3, se procede a describirlos en términos de un conjunto de características que permiten describir propiedades geométricas y de forma de dichos objetos.

Un objeto segmentado y etiquetado dado $O_i,\ i = 1, 2, \cdots, p$, con $p$ el número de posibles objetos que pueden ser detectados y/o identificados, suele ser representado por un vector de $n$ rasgos:

$$(\, r_1 \quad r_2 \quad \cdots \quad r_n \,)^T. \tag{5.10}$$

El primer rasgo $r_1$ puede caracterizar el color del objeto, el segundo rasgo $r_2$ su textura, el tercero $r_3$ su número de huecos, y así sucesivamente hasta el $n$-ésimo rasgo.

En las siguientes subsecciones estudiaremos un conjunto de rasgos de tipo geométrico que permiten describir propiedades del objeto como su área, perímetro y qué tan compacto es. También se estudiará un conjunto de rasgos que permiten caracterizar propiedades topológicas de los objetos, como son si tienen o no huecos y su número o su llamado número de Euler.

## 5.4.1 Rasgos descriptivos y sus propiedades

Ya vimos que un rasgo descriptivo $r_j,\ j = 1, 2, \cdots, n$ de un objeto $O_i$ permite describir un aspecto relacionado con la geometría o la estructura topológica de dicho objeto. Para que un rasgo sea útil para la detección o reconocimiento del objeto $O_i$, debe poseer un conjunto de propiedades deseables. Algunas de estas propiedades son las siguientes:

**Poder de discriminación**. El rasgo $r_j$ debe permitir discriminar entre objetos de distintas clases; debería de dar valores numéricos diferentes para objetos de diversas clases.

**Invariabilidad**. Un rasgo debe ser invariante; esto es, con cambios numéricos pequeños para objetos de la misma clase.

**No correlacionado**. Un rasgo debería de presentar la menor correlación posible con otros rasgos.

**Rapidez**. Un rasgo debería de poder calcularse en tiempos aceptables. Este requisito puede llegar a ser determinante para ciertas aplicaciones.

**Economía**. Un rasgo debería de poder obtenerse con sensores económicos.

Un rasgo descriptor puede ser:

Una parte de $O_i$ con algunas propiedades especiales, por ejemplo, una línea o una región con una textura dada.

Una propiedad global del objeto completo o de una parte del mismo, por ejemplo, el área en píxeles encerrados por la región del objeto o el color de la superficie del mismo.

Al primer tipo de rasgo descriptivo en la literatura se le conoce como índice visual mientras que, al segundo se le designa rasgo objeto. Un índice visual refiere a una posición interesante en el objeto, como a un borde, línea, segmento de curva, esquina, etcétera. Por otro lado, un rasgo objeto es relacionado con mediciones geométricas o topológicas, como el área o número de Euler, obtenidos a partir de la región conectada de píxeles o el contorno del objeto.

### 5.4.2 Rasgos globales y locales

Desde un punto descriptivo, un rasgo puede pertenecer a una de dos categorías: global o local. En el primer caso, el rasgo describe una propiedad global del objeto $O_i$ como puede ser el tamaño del perímetro.

Un rasgo local por otro lado, describe a una parte del objeto por ejemplo, el área calculada a partir de una parte del mismo. En este libro estudiaremos únicamente rasgos descriptivos de tipo global. Sin pérdida de generalidad, este tipo de rasgos son suficientes para los fines explicativos de esta obra.

### 5.4.3 Factor de compacidad y su cálculo

Este rasgo fue uno de los primeros utilizados por la comunidad para describir en forma global una propiedad geométrica de un objeto. Viene dado como una función del perímetro y el área del mismo. Más en detalle, el factor compacidad (FC) de objeto $O_i$, con área $A$ y perímetro $P$ viene dado como:

$$\text{FC} = \frac{P^2}{4\pi A}. \tag{5.11}$$

Se puede demostrar que el valor de FC presenta un valor mínimo para un objeto circular. De hecho, en el caso continuo, sabiendo que para un círculo $P = 2\pi r$ y $A = \pi r^2$, el valor mínimo de FC para un objeto circular es:

$$\text{FC} = \frac{P^2}{4\pi A} = \frac{(2\pi r)^2}{4\pi(\pi r^2)} = \frac{4\pi^2 r^2}{4\pi^2 r^2} = 1.0. \tag{5.12}$$

En el caso discreto, el área $A$ de una región conectada de píxeles R, es el número de estos que la componen. De la misma forma, el perímetro $P$ de la región $R$, es el número de píxeles en su frontera. Finalmente, el contorno $c(R)$ de $R$ es el conjunto de píxeles del perímetro $P$ de la región $R$.

El caso discreto, el valor de FC tenderá a ser menor, conforme el objeto sea más compacto; y tenderá a ser mayor conforme el objeto sea menos compacto o más irregular.

El FC sin normalizar a la forma de un círculo se calcula como sigue:

$$\text{FC} = \frac{P^2}{A}. \tag{5.13}$$

**Ejemplo 5.9**

Considérense, por ejemplo, los cuatro objetos mostrados en la figura 5.11. Todos los objetos tienen la misma área de 16 píxeles, marcados en gris, pero diferente forma. Los píxeles de cada objeto aparecen en un tono de gris más cargado.

El respectivo FC para cada uno de estos cuatro objetos viene dado, respectivamente, como: Objeto 1: $\frac{12^2}{16} = 9.0$, Objeto 2: $\frac{14^2}{16} = 12.25$, Objeto 3: $\frac{16^2}{16} = 16.0$, y Objeto 4: $\frac{16^2}{16} = 16$.

En el caso de los tres primeros objetos, note cómo el FC de los tres crece conforme el objeto se va haciendo menos compacto. Note el caso de los objetos tres y cuatro que a pesar de

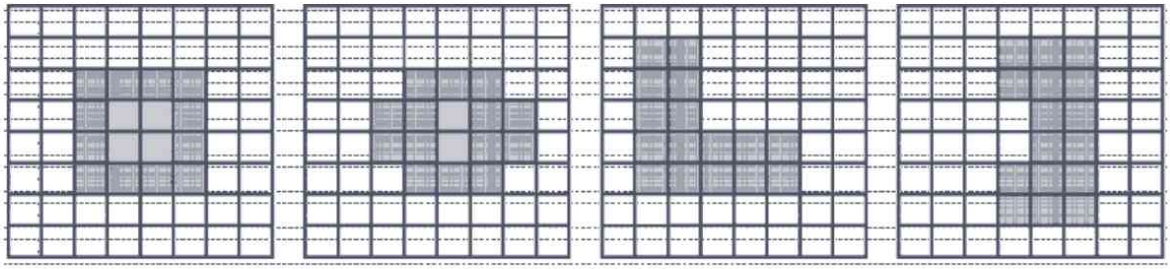


**Figura 5.11**: Cuatro objetos de diferente forma.

que tienen diferente forma, poseen el mismo FC. Esto nos hace ver que el FC no es un descriptor único para cada objeto. Desde el punto de vista de la diferenciación de objetos a través del FC se puede apreciar una clara diferencia entre los tres primeros, sin embargo, una evidente no diferenciación entre los objetos tres y cuatro. Esto sugiere que el FC no puede ser utilizado en lo general para diferencias entre formas.

Un procedimiento sencillo para el cálculo del factor de compacidad FC de una región binaria $R$, $k$-conectada es el siguiente:

**Pseudocódigo 5.2**

Procedimiento cálculo de factor de compacidad FC de una región binaria $R$

Entrada: imagen binaria $b(x, y)$ de $n \times m$ píxeles conteniendo una región $R$.
Salida: compacidad FC de la región $R$.

Establecer las variables $P = 0$ y $A = 0$.
Algoritmo para calcular el factor de compacidad FC de una región binaria $R$.

```
for y=1:m
    for x=1:n
        if p==1
            A=A+1;
        end
        if p==1 and p ∈ c(R)
            P=P+1
        end
    end
end
```

Calcular el factor de compacidad mediante la ecuación (5.11).

Otra forma de calcular el factor de compacidad FC de un objeto se obtiene a partir del llamado perímetro de contacto entre los píxeles del mismo:

$$P_c = \frac{(T_n - P)}{2}. \tag{5.14}$$

En este caso, $T$ es el número de lados de la celda (píxel) de base, $T$=3 para celdas de tipo triangular, $T$=4 para el caso de celdas rectangulares y $T$=6 para el caso de celdas hexagonales, $n$ es el número de píxeles del objeto, y $P$ es el número de lados en la frontera del mismo.

Por ejemplo:

- El valor de $n$ para las cuatro formas de la figura 5.11 es 16.
- El perímetro $P$ para las cuatro formas de la figura 5.11 es, respectivamente: objeto 1: $P$=16, objeto 2: $P$=18, objeto 3: $P$=20, y objeto 4: $P$=20.
- El perímetro de contacto $P_c$ de una forma compuesta de píxeles es igual a la suma de las longitudes de los segmentos comunes a dos celdas.
- El perímetro de contacto $P_c$ para las cuatro formas de la figura 5.11 es, respectivamente: objeto 1: $P_c = 24$, objeto 2: $P_c = 23$, objeto 3: $P_c = 22$, y objeto 4: $P_c = 22$.

En su trabajo de 1997, Bribiesca [15] define el factor de compacidad FC a partir del perímetro de contacto $P_c$ de un objeto como:

$$FC = P_c. \tag{5.15}$$

Para hacer invariante este rasgo a traslaciones, rotaciones y cambios de escala, Bribiesca propone la siguiente formulación para el factor de compacidad normalizado (FCN):

$$\text{FCN} = \frac{(\text{FC} - \text{FC}_{min})}{(\text{FC}_{max} - \text{FC}_{min})}. \tag{5.16}$$

En este caso: $\text{FC}_{min} = n - 1$ representa el valor mínimo que puede tomar FC y $\text{FC}_{max} = \frac{T_n - 4\sqrt{n}}{2}$ es el valor máximo que puede tomar.

- El $\text{FC}_{min}$ y $\text{FC}_{max}$ para las cuatro formas de la figura 5.11 es, respectivamente: $\text{FC}_{min} = n - 1 = 16 - 1 = 15$ y $\text{FC}_{max} = \frac{T_n - 4\sqrt{n}}{2} = \frac{4(16) - 4(16)}{2} = 24$.

**Ejemplo 5.10**

El factor de compacidad normalizado en términos de la ecuación (5.13) para las cuatro formas de la figura 5.11 es, respectivamente:

- Objeto 1: FCN $= \frac{24-15}{24-15} = 1.0$
- Objeto 2: $P_c = \frac{23-15}{24-15} = 0.8888$
- Objeto 3: $P_c = \frac{22-15}{24-15} = 0.7777$
- Objeto 4: $P_c = \frac{22-15}{24-15} = 0.7777$.

Note cómo ahora las formas más compactas toman valores cercanos de FCN a 1.0, mientras que formas menos compactas toman valores más pequeños. De hecho, se puede demostrar que el valor de FCN varía entre 0.0 y 1.0.

### 5.4.4 Rasgos descriptivos con momentos geométricos

Los rasgos descriptivos basados en los momentos geométricos han sido usados por décadas en la caracterización de objetos, tanto en dos como en tres dimensiones. En este capítulo repasaremos los momentos geométricos como descriptores de la forma de un objeto en el caso 2-dimensional. Estudiaremos primeramente, los llamados momentos geométricos estándar. A partir de estos, enseguida estudiaremos los momentos centrales, los cuales ya son invariantes a traslaciones en el plano. Después, con base en estos momentos, veremos algo de la teoría relativa a los momentos invariantes a rotaciones y cambios de tamaño.

Finalmente, veremos cómo a partir de todos estos rasgos se puede derivar un grupo de invariantes ante grupos de transformaciones imagen, más generales, como son las traslaciones combinadas con rotaciones y cambios de tamaño. En el caso de una función continua bidimensional $f(x, y)$, el momento de orden $(p + q)$ se define como:

$$M_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q \, f(x, y) dx dy, \text{ para } p, q = 0, 12, \cdots \tag{5.17}$$

Una característica del conjunto infinito de momentos $M_{pq}$ generados es que describe de forma única cada función $f(x, y)$.

En el caso de una imagen digital, la doble integral suele ser reemplazada por una doble suma, dando como resultado:

$$M_{pq} = \Sigma_x \Sigma_y x^p y^q \ f(x,y), \text{ para } p,q = 0, 12, \cdots \tag{5.18}$$

En el caso de una imagen binaria $b(x,y)$ con valores en $\{0,\ 1\}$, la ecuación (5.18) toma la forma:

$$M_{pq} = \Sigma_{(\,x,\,y\,)\in R}\ x^p y^q, \text{ para } p,q = 0, 12, \cdots \tag{5.19}$$

donde $R$ denota la región del objeto.

### 5.4.5 Invariantes a traslaciones: momentos centrales

Se puede demostrar que los momentos $M_{pq}$ no son invariantes a transformaciones de tipo imagen como traslaciones, esto es a transformaciones del tipo:

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}. \tag{5.20}$$

En este caso $x$ y $y$ son las coordenadas del objeto original, $[\,a,\ b\,]^T$ es el vector de desplazamiento, mientras que $x^{'}$ y $y^{'}$ son las coordenadas del objeto transformado.

Para lograr que los $M_{pq}$ sean invariantes ante transformaciones del tipo dado por la ecuación (5.20), se proponen los llamados momentos centrales, denotados como $\mu_{pq}$, los cuales en el caso discreto para una región $R$ compuesta por $M_{00}$ píxeles, se definen como:

$$\mu_{pq} = \Sigma_{(\,x,\,y\,)\in R}\ \Sigma\,(\,x-\bar{x}\,)^p\,(\,y-\bar{y}\,)^q\,, \quad \text{para } p,q = 0,1,2,\cdots \tag{5.21}$$

donde:

$$\bar{x} = \frac{M_{10}}{M_{00}} = \frac{M_{01}}{M_{00}} \tag{5.22}$$

son las coordenadas del llamado centroide del objeto. En particular, los momentos centrales hasta de orden tres, esto es, los momentos para los cuales $(p+q) \leq 3$ vienen dados como:

$$\begin{aligned}
\mu_{00} &= M_{00} & \mu_{11} &= M_{11} - \bar{y}M_{10} \\
\mu_{10} &= 0 & \mu_{30} &= M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2 M_{10} \\
\mu_{01} &= 0 & \mu_{12} &= M_{12} - 2\bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2 M_{10} \\
\mu_{20} &= M_{20} - \bar{x}M_{10} & \mu_{21} &= M_{21} - 2\bar{x}M_{11} + \bar{y}M_{20} + 2\bar{x}^2 M_{01} \\
\mu_{02} &= M_{02} - \bar{y}M_{01} & \mu_{03} &= M_{03} - 3\bar{y}M_{02} + 2\bar{y}^2 M_{01}.
\end{aligned}$$

**• Ejemplo 5.11**

Demostrar que cualquiera de los invariantes centrales es invariante ante traslaciones en el plano.

**Solución**

Con la finalidad de demostrar que la propiedad de cualquier invariante central se mantiene ante traslaciones en el plano, considere la figura 5.12a donde se muestra un objeto compuesto por seis píxeles. Sin pérdida de generalidad, tomemos cualquiera de los 10 momentos centrales; por ejemplo, considere el siguiente momento $\mu_{20}$.

Este momento para el objeto mostrado en la figura 5.12a, tiene un valor de:

$$\begin{aligned} M_{20} &= \Sigma_{(x,y)\in R}\Sigma\, x^2y^0 = 1^2+2^2+3^2+1^2+2^2+3^2 = 28. \\ M_{10} &= \Sigma_{(x,y)\in R}\Sigma\, x^1y^0 = 1^1+2^1+3^1+1^1+2^1+3^1 = 12. \\ \bar{x} &= \frac{M_{10}}{M_{00}} = \frac{\Sigma_{(x,y)\in R}\ \Sigma\, x^1y^0}{\Sigma_{(x,y)\in R}\ \Sigma\, x^0y^0} = \frac{1^1+2^1+3^1+1^1+2^1+3^1}{1^0+2^0+3^0+1^0+2^0+3^0} = \frac{12}{6} = 2. \\ \mu_{20} &= M_{20} - \bar{x}M_{10} = 28 - 2(12) = 4. \end{aligned}$$

El mismo momento para el objeto trasladado mostrado en la figura 5.12b vale:

$$\begin{aligned} M_{20} &= \Sigma_{(x,y)\in R}\Sigma\, x^2y^0 = 4^2+5^2+6^2+4^2+5^2+6^2 = 154. \\ M_{10} &= \Sigma_{(x,y)\in R}\Sigma\, x^1y^0 = 4^1+5^1+6^1+4^1+5^1+6^1 = 30. \\ \bar{x} &= \frac{M_{10}}{M_{00}} = \frac{\Sigma_{(x,y)\in R}\ \Sigma\, x^1y^0}{\Sigma_{(x,y)\in R}\ \Sigma\, x^0y^0} = \frac{4^1+5^1+6^1+4^1+5^1+6^1}{4^0+5^0+6^0+4^0+5^0+6^0} = \frac{30}{6} = 5. \\ \mu_{20} &= M_{20} - \bar{x}M_{10} = 154 - 5(30) = 4. \end{aligned}$$

Note cómo efectivamente el valor $\mu_{20}$ se mantiene invariante ante traslaciones dadas por la ecuación (5.20).

El lector demostrará que cualquier momento central $\mu_{pq}$ es invariante a traslaciones en el plano dadas por la ecuación (5.20), ver problema 5.7.7 en la página 228.

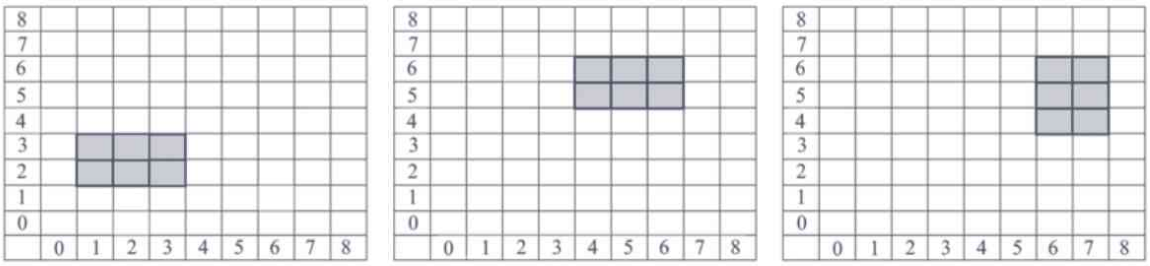



**Figura 5.12**: Imagen (a) un objeto compuesto por seis píxeles. Imagen (b) el mismo objeto desplazado hacia la derecha y hacia arriba. Imagen (c) el mismo objeto desplazado y rotado 90°.

### 5.4.6 Invariantes a rotaciones

Las rotaciones que un objeto puede experimentar en el plano son causadas por transformaciones del tipo:

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\operatorname{sen}(\theta) \\ \cos(\theta) & \operatorname{sen}(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.23}$$

En este caso $x$ y $y$ son las coordenadas del objeto original, $\theta$ es el ángulo de rotación y $x^{'}$ y $y^{'}$ son las coordenadas del objeto transformado.

La derivación de momentos invariantes ante rotaciones de la forma (5.23) no es fácil de obtener. Los primeros dos invariantes a traslaciones y rotaciones propuestos en [18], vienen dados por las siguientes expresiones:

$$\phi_1 = \mu_{20} + \mu_{02} \tag{5.24a}$$

$$\phi_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \tag{5.24b}$$

$$\phi_3 = (\mu_{30} - 3\mu_{12})^2 + 3(\mu_{21} - \mu_{03})^2 \tag{5.24c}$$

$$\phi_4 = (\mu_{30} + \mu_{12})^2 + 3(\mu_{21} + \mu_{03})^2 \tag{5.24d}$$

$$\begin{aligned} \phi_5 = {} & (\mu_{30} - \mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} - \mu_{03})^2] \\ & + (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \end{aligned} \tag{5.24e}$$

$$\begin{aligned} \phi_6 = {} & (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \\ & + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03}) \end{aligned} \tag{5.24f}$$

$$\phi_7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2]$$

$$+ ( 3\mu_{21} - \mu_{03} ) ( \mu_{21} + \mu_{03} ) [ 3 ( \mu_{30} + \mu_{12} )^2 - ( \mu_{21} + \mu_{03} )^2 ] . \qquad (5.24g)$$

**Nota:** Se puede mostrar que los momentos $\phi_i,\ i = 1, 2, \cdots$ son invariantes ante rotaciones y traslaciones.

**Ejemplo 5.12**

Suponga que el objeto mostrado en la figura 5.12a es trasladado y rotado 90° a la derecha o a la izquierda, como se ilustra en la figura 5.12c. El primer momento invariante a traslaciones y rotaciones de Hu $\phi_1$ para el objeto mostrado en la figura 5.12a, vale:

$$\begin{aligned} \mu_{20} &= M_{20} - \bar{x}M_{10} = 28 - 2(12) = 4. \\ \mu_{02} &= M_{02} - \bar{y}M_{01} = 39 - 2.5(15) = 1.5. \\ \phi_1 &= \mu_{20} + \mu_{02} = 4 + 1.5 = 5.5. \end{aligned}$$

El mismo momento pero ahora para el objeto mostrado en la figura 5.12c, vale:

$$\begin{aligned} \mu_{20} &= M_{20} - \bar{x}M_{10} = 255 - 6.5(39) = 1.5. \\ \mu_{02} &= M_{02} - \bar{y}M_{01} = 154 - 5(30) = 4.0. \\ \phi_1 &= \mu_{20} + \mu_{02} = 1.5 + 4.0 = 5.5. \end{aligned}$$

Note cómo efectivamente el valor $\phi_1$ se mantiene invariante ante traslaciones y rotaciones dadas por las ecuaciones (5.20) y (5.23).

### 5.4.7 Invariantes a cambios de escala

Los cambios de escala (de tamaño) que un objeto puede experimentar dentro del espacio de un robot son causados por las transformaciones de coordenadas de la forma:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} . \qquad (5.25)$$

Los invariantes a cambios de escala pueden obtenerse al dividir cada momento por un factor de normalización que cancele el efecto de escalamiento. Este factor de normalización, generalmente viene dado por el tamaño del objeto en cuestión.

Si $f^{'}(x^{'},\ y^{'})$ es la imagen $f(x,y)$, después de un escalamiento en cada eje por un factor $\alpha$ a través de la aplicación dada por la ecuación (5.26), entonces se obtiene lo siguiente: $f^{'}(x^{'},\ y^{'}) = f(\alpha x, \alpha y) = \alpha f(x,y)$; con $x^{'} = \alpha x$, $y^{'} = \alpha y$. En consecuencia:

$$M^{'}_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^{'p} y^{'q} f(x^{'}, y^{'})\ dx^{'} dy^{'} \tag{5.26}$$

$$M^{'}_{pq} = \alpha^{p+q+2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^{p} y^{q} f(x,y)\ dxdy \tag{5.27}$$

por lo tanto, $M^{'}_{pq} = \alpha^{p+q+2} = M_{pq}$, de manera similar $\mu^{'}_{pq} = \alpha^{p+q+2}\mu_{pq}$. En particular, $\mu^{'}_{00} = \alpha^2\mu_{00}$, de donde se puede ver que:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu^{\gamma}_{00}},\quad \gamma = \frac{p+q}{2} + 1,\quad p+q = 2,3,\cdots \tag{5.28}$$

Se deja al lector demostrar este hecho (ver problema 5.7.8, en la página 228).

### 5.4.8 Invariantes traslación, rotación y escalamiento

Dentro del espacio de trabajo del robot, el objeto a ser manipulado puede experimentar una combinación de transformaciones; usualmente cambios de posición (traslaciones en el plano), giros (rotaciones en el plano) y cambios de tamaño (escalamientos), estos últimos ocasionados, por ejemplo, por el posicionamiento del captor con respecto al objeto.

Una transformación tipo imagen que combina cambios de posición, rotaciones y escalamientos viene dada por la siguiente expresión:

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \alpha \begin{bmatrix} \cos(\theta) & -\operatorname{sen}(\theta) \\ \cos(\theta) & \operatorname{sen}(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}. \tag{5.29}$$

donde, como ya vimos, $[a,b]^T$ es el vector de desplazamiento, $\theta$ el ángulo de rotación y $\alpha$ el factor de escalamiento.

Al reemplazar simplemente los momentos centrales $\mu_{pq}$ en las expresiones para invariantes a traslaciones y rotaciones, dados por las ecuaciones (5.24a) a (5.24g), por los invariantes a cambios de escala $\eta_{pq}$, resultan los famosos siete invariantes de Hu a transformaciones dadas por la ecuación (5.29):

$$\phi_1 = \eta_{20} + \eta_{02} \tag{5.30a}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{5.30b}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{5.30c}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + 3(\eta_{21} + \eta_{03})^2 \tag{5.30d}$$

$$\begin{aligned}\phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned} \tag{5.30e}$$

$$\begin{aligned}\phi_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned} \tag{5.30f}$$

$$\begin{aligned}\phi_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].\end{aligned} \tag{5.30g}$$

Un procedimiento sencillo de la obtención de los invariantes de Hu en una región binaria es:

**Pseudocódigo 5.3**

Cálculo de los 7 invariantes de Hu de una región binaria $R$.

Entrada: imagen binaria $b(x, y)$ de dimensiones $n \times m$ píxeles.
Salida: siete invariantes de Hu: $\phi_1,\ \phi_2, \cdots,\ \phi_7$.

```
for y=1:m
    for x=1:n
        if p==1
            Calcular 10 primeros momentos geométricos M_pq, ecuación (5.19).
        end
    end
end
```

Calcular los 10 momentos centrales $\mu_{pq},\ (p+q) \leq 3$ mediante la ecuación (5.21).
Calcular los 7 momentos centrales normalizados $\eta_{pq},\ 2 \leq (p+q) \leq 3$, ecuación (5.28).
Calcular las 7 invariantes de Hu mediante la ecuación (5.30a-5.30g).

**Ejemplo 5.13**

Considere los dos objetos mostrados en las figuras 5.13a y 5.13e, así como sus tres transformaciones tipo imagen ilustradas, respectivamente en las figuras 5.13b-5.13d y 5.13f-5.13h. La figura 5.14 muestra los valores del primer invariante $\phi_1$ para cada uno de los dos objetos y sus correspondientes cuatro transformaciones.

Note cómo a pesar de los cambios de posición, rotación y de tamaño, el valor de la invariante para los dos objetos se mantiene constante. Esto deja ver que los invariantes de Hu pueden ser utilizados como poderosos rasgos para describir la forma del objeto ante transformaciones de posición, rotación y tamaño.

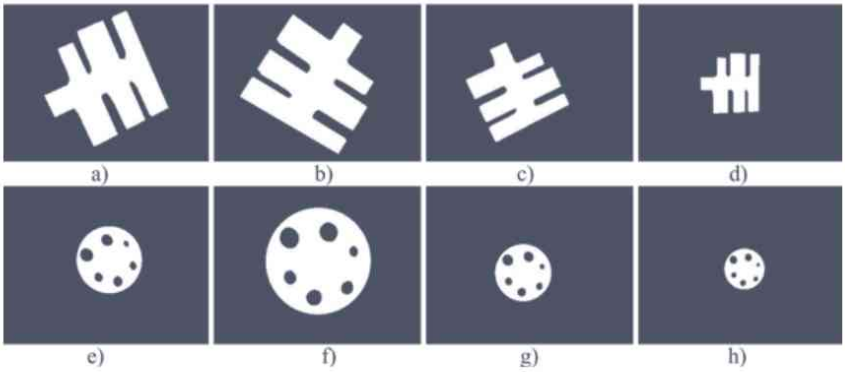



**Figura 5.13**: Imágenes (a) y (e) dos objetos, uno sin huecos y otro con seis huecos. Imágenes (b-d) tres versiones transformadas del objeto mostrado en (a). Imágenes (f-h) tres versiones transformadas del objeto mostrado en (e).

| Objeto | Versión transformada 1 | Versión transformada 2 | Versión transformada 3 |
|---|---|---|---|
|  |  |  |  |
| 0.1956 | 0.1957 | 0.1978 | 0.1975 |
|  |  |  |  |
| 0.1832 | 0.1840 | 0.1842 | 0.1833 |

**Figura 5.14**: Dos objetos y sus valores del invariante de Hu $\phi_1$ y el correspondiente a tres versiones transformadas de cada objeto.

### 5.4.9 Posición del objeto en el plano de trabajo

Una vez que un objeto ha sido identificado es necesario que sea ubicado dentro del espacio de trabajo del robot para que este último pueda realizar una tarea con dicho objeto, por ejemplo, tomarlo. Para esto, es necesario conocer en la imagen las coordenadas del centro del objeto (centroide) y del ángulo del llamado eje principal del objeto.

La posición del centroide del objeto $R$ dentro de su imagen captada por el sensor del robot viene dada por sus coordenadas $(\bar{x}, \bar{y})$. Estas, como ya vimos, se obtienen a través de los momentos estándar $M_{00}, M_{10}$ y $M_{01}$ como sigue:

$$\bar{x} = \frac{M_{10}}{M_{00}} \tag{5.31a}$$

$$\bar{y} = \frac{M_{01}}{M_{00}} \tag{5.31b}$$

las coordenadas del centroide $(\bar{x}, \bar{y})$ son la intersección de las líneas paralelas $x = \bar{x}$ y $y = \bar{y}$, respectivamente a los ejes $x$ e $y$.

Note que el centroide define una posición única con respecto al objeto, que puede ser usada como punto de referencia para describir la posición del objeto dentro del campo visual del captor del robot.

**Ejemplo 5.14**

Considere al objeto binario compuesto por 9 píxeles mostrado en la figura 5.15.

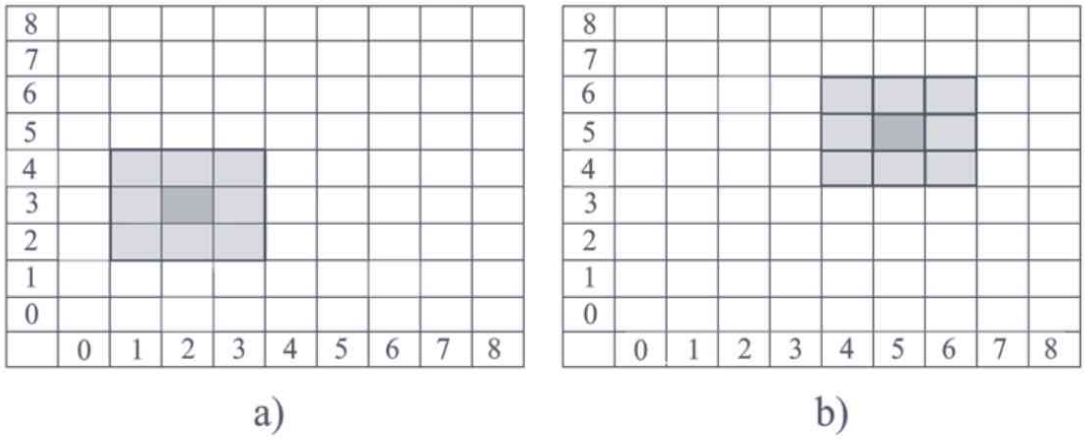



**Figura 5.15**: Imagen (a) un objeto compuesto por nueve píxeles con centro en (2,3). Imagen (b) el mismo objeto, pero con centro en (5,5). En cada caso, el centro del objeto aparece marcado en un tono de gris más oscuro.

Al aplicar las ecuaciones (5.31a) y (5.31b) a los nueve píxeles de este objeto se puede ver que la posición de su centroide viene dada como sigue:

$$\begin{aligned}\bar{x} &= \frac{M_{10}}{M_{00}} \\ &= \frac{1+2+3+1+2+3+1+2+3}{9} = \frac{18}{9} = 2\end{aligned}$$

$$\begin{aligned}\bar{y} &= \frac{M_{01}}{M_{00}} \\ &= \frac{4+4+4+3+3+3+2+2+2}{9} = \frac{27}{9} = 3.\end{aligned}$$

Posición que coincide con la observada en la imagen.

Suponga que ahora el objeto es desplazado a la posición mostrada en la figura 5.15. La nueva posición del centroide vale ahora:

$$\begin{aligned}\bar{x} &= \frac{M_{10}}{M_{00}} \\ &= \frac{4+5+6+4+5+6+4+5+6}{9} = \frac{45}{9} = 5\end{aligned}$$

$$\begin{aligned}\bar{y} &= \frac{M_{01}}{M_{00}} \\ &= \frac{6+6+6+5+5+5+4+4+4}{9} = \frac{45}{9} = 5.\end{aligned}$$

Posición que, de nuevo, coincide con la observada en la imagen.

El ángulo del eje principal del objeto que pasa por su centroide y que concentra la masa de dicho objeto de la figura 5.15 puede ser obtenido a través de los momentos centrales como sigue.

Primero se calcula el ángulo auxiliar:

$$\phi = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right). \tag{5.32}$$

Es de notar que el rango del ángulo $\phi$ se encuentra en el rango $-\frac{\pi}{4} \leq \phi \leq \frac{\pi}{4}$.

En la figura 5.16 se presenta una imagen con cinco objetos binarios y alargados con sus ejes principales pasando por su respectivo centroide. En la figura 5.17 se ilustra cómo el ángulo del eje principal mayor $\theta$ puede ser determinado a partir de los segundos momentos y el ángulo $\phi$. El ángulo del eje principal de menor inercia puede ser usado como eje de referencia única para describir la orientación del objeto dentro del campo visual (rotación en el plano). Nótese que $\theta$ no garantiza por sí solo una orientación única debido a la existencia de la ambigüedad en 180°. Esta ambigüedad puede ser resuelta al usar el signo de los momentos de orden tres.

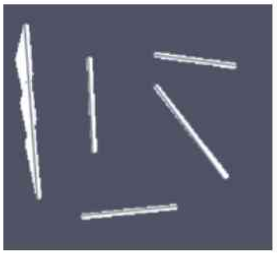


**Figura 5.16:** Objetos binarios.

| $\mu_{11}$ | $\mu_{20} - \mu_{02}$ | $\phi$ | $\theta$ |
|---|---|---|---|
| 0 | - | 0 | $\frac{\pi}{2}$ |
| + | - | $-\frac{\pi}{4} < \phi < 0$ | $\frac{\pi}{4} < \phi < \frac{\pi}{2}$ |
| + | 0 | 0 | $\frac{\pi}{4}$ |
| + | + | $0 < \phi < \frac{\pi}{4}$ | $0 < \theta < \frac{\pi}{4}$ |
| 0 | 0 | 0 | 0 |
| - | + | $-\frac{\pi}{4} < \phi < 0$ | $-\frac{\pi}{4} < \theta < 0$ |
| - | 0 | 0 | $-\frac{\pi}{4}$ |
| - | - | $0 < \phi < \frac{\pi}{4}$ | $-\frac{\pi}{2} < \theta < -\frac{\pi}{4}$ |

**Figura 5.17**: Orientación del eje mayor principal.

### 5.4.10 Rasgos estructurales: característica de Euler

La característica de Euler $E$ ha sido ampliamente usada en el campo de la visión por ordenador para el reconocimiento de objetos en 2D y en 3D (véase, por ejemplo, [23]). Esta característica, como se sabe, permite relacionar el número de componentes del objeto ($nc$) y el número de hoyos ($nh$) del mismo objeto de la siguiente manera:

$$E = nc - nh. \tag{5.33}$$

El número de Euler de un objeto o una imagen puede ser calculado de muchas maneras. Una de ellas consiste en combinar operaciones de procesamiento de imágenes muy sencillas, como sigue. Dada una imagen binaria $b(x, y)$ con uno o más objetos con y sin huecos:

1) Calcular el número de componentes conectados (CC). Esto nos da el número de componentes $nc$.
2) Negar la imagen $b(x, y)$. Esto da como resultado la imagen complemento $\tilde{b}(x, y)$.
3) Calcular el número de CC sobre $\tilde{b}(x, y)$. Esto nos da el número de componentes $NC + 1$. El 1 representa la región del fondo de la imagen.
4) Obtener el valor $nh$ al restar al valor de CC calculado en el paso 3.
5) Calcular el número de Euler de la imagen simplemente al aplicar la ecuación (5.33).

**Ejemplo 5.15**

Suponga la imagen binaria mostrada en la figura 5.18a. Al aplicar el paso número 1 del procedimiento descrito, a través del método explicado en la subsección 5.3.4: $nc = 4$. Al negar la imagen $b(x, y)$ se obtiene la imagen invertida $\tilde{b}(x, y)$, mostrada en la figura 3.33 (b). Al aplicar el paso 3, se obtiene $nc = 4$. Al aplicar el paso 4, se obtiene que $nh = 3$. Finalmente, al aplicar el paso 5, se obtiene, como número de Euler $E$ para la imagen de la figura 3.33 (a), $E = nc - nh = 4 - 3 = 1$, valor que coincide con el calculado directamente mediante la ecuación (5.33).

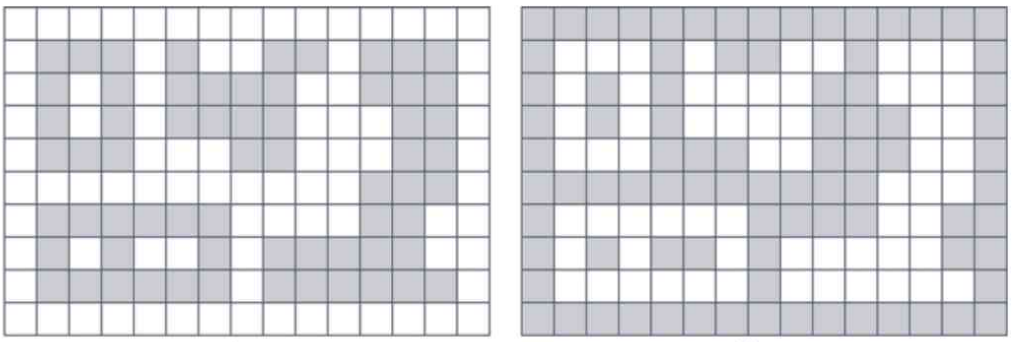



**Figura 5.18**: Imagen binaria usada para el ejemplo 5.15.

Otra manera para calcular el número de Euler de un objeto (imagen binaria) utiliza la siguiente formulación:

$$E = \#\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} - \#\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{5.34}$$

donde $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ y $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ son dos máscaras, que al ser pasadas por la imagen binaria en cuestión, entregan como resultado un "1" o un "0", según los valores que son encontrados a su paso en el recorrido. El píxel de referencia de la máscara es localizado en la esquina superior izquierda. Para la máscara $\begin{bmatrix} \mathbf{1} & 0 \\ 0 & 0 \end{bmatrix}$ el píxel de referencia sería el píxel marcado en negritas; lo mismo sería para las segundas máscaras: $\begin{bmatrix} \mathbf{1} & 1 \\ 1 & 0 \end{bmatrix}$. La notación $\#\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$ indica el número de veces que la máscara ha sido encontrada en la imagen al recorrerla.

### Ejemplo 5.16

Considere la imagen binaria mostrada en la figura 5.19a. Es la misma que la de la figura 5.18a. Al recorrer las máscaras en la figura 5.19b aparecen marcadas con un "×" las posiciones donde la máscara $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ es encontrada en el recorrido. En la misma imagen, aparecen marcadas con una "•" las posiciones donde ahora la máscara $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ es encontrada en el recorrido. Luego entonces, $\#\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = 6$ y $\#\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = 5$, por tanto, de la ecuación (5.34) $E = \#\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} - \#\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = 6 - 5 = 1$, que coincide con el valor esperado.

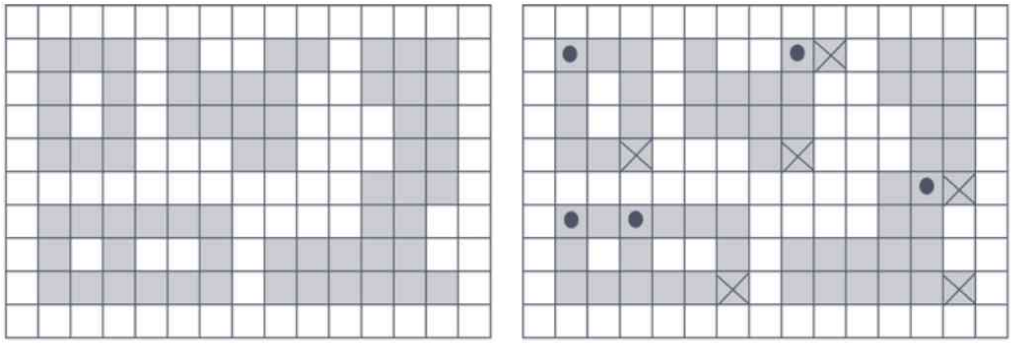



**Figura 5.19**: Imagen binaria usada para el ejemplo 5.16.

### Ejemplo 5.17

Consideremos el caso de las dos imágenes umbraladas mostradas en la figura 5.20, la cuales contienen como se puede ver tres y cinco objetos, respectivamente. En el caso de la imagen de la izquierda, el objeto hexagonal tiene un hueco; el segundo abajo a su izquierda cero huecos, mientras que el tercero abajo a su derecha tiene siete huecos.

Para la imagen de la derecha, por otro lado, como se puede ver tres objetos tienen un hueco, uno más tiene dos huecos, mientras que el último tiene seis huecos. Al aplicar la ecuación (5.34) a estas dos imágenes, el lector puede verificar que se obtienen los siguientes números de Euler:

Imagen (a): $E$=-5.

Imagen (b): $E$=-6.


**Figura 5.20**: Imagen (a) binaria con tres objetos. Imagen (b) binaria con cinco objetos.

### 5.4.11 Descripción de los objetos

Una vez que los objetos han sido aislados unos de otros y del fondo, como ya vimos en la sección anterior, se les puede calcular rasgos descriptores como el facto de compacidad, los momentos de Hu o el número de Euler; se les asigna un vector descriptor el cual podrá ser usado para detectar su presencia en una imagen, determinar su número o identificarlos.

Esto permitirá al robot ubicar a los objetos y realizar una tarea con ellos, por ejemplo, tomar uno en particular y ponerlo dentro de un contenedor especificado por la tarea.

**Ejemplo 5.18**

Consideremos el conjunto de seis objetos umbralados mostrados en la figura 5.21. En la figura 5.22 se ilustran su respectiva primera invariante de Hu y número de Euler.

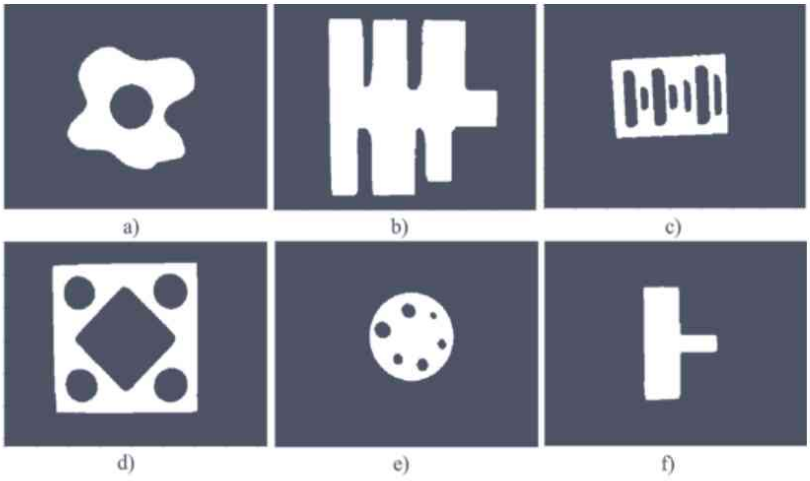



**Figura 5.21**: (a)-(f) Seis objetos binarios con diferente forma y diferente número de huecos.

| Objeto | $\phi_1$ | $E$ |
|---|---|---|
| 1 | 0.228647 | 0 |
| 2 | 0.194198 | 1 |
| 3 | 0.290548 | -6 |
| 4 | 0.356483 | -4 |
| 5 | 0.181847 | -5 |
| 6 | 0.246813 | 1 |

**Figura 5.22**: Primera invariante de Hu y número de Euler como descriptores de seis objetos.

Con base a la figura 5.22 podemos formar las descripciones de los seis objetos como sigue:

$$\boldsymbol{o}_1 = \begin{pmatrix} 0.228647 \\ 0 \end{pmatrix}, \quad \boldsymbol{o}_2 = \begin{pmatrix} 0.194198 \\ 1 \end{pmatrix}, \quad \boldsymbol{o}_3 = \begin{pmatrix} 0.290548 \\ -6 \end{pmatrix}$$

$$\boldsymbol{o}_4 = \begin{pmatrix} 0.356483 \\ -4 \end{pmatrix}, \quad \boldsymbol{o}_5 = \begin{pmatrix} 0.181847 \\ -5 \end{pmatrix}, \quad \boldsymbol{o}_6 = \begin{pmatrix} 0.246813 \\ 1 \end{pmatrix}.$$

En lo general, un objeto $\boldsymbol{o}_i$ puede ser descrito en términos de $n$ rasgos mediante un vector como sigue:

$$\boldsymbol{x} = [r_1 \quad \cdots \quad r_n]^T. \tag{5.35}$$

En la siguiente sección utilizaremos este tipo de descripciones para detectar la presencia de objetos específicos en una imagen, así como para reconocerlos.

## 5.5 Detección y reconocimiento

Una vez que los objetos dentro de una imagen han sido descritos como se refirió en la sección anterior, estas descripciones pueden ser usadas para detectar su presencia o reconocerlos. De esta manera un robot estaría en posibilidades de llevar a cabo una tarea especificada.

En este apartado estudiaremos un conjunto de clasificadores muy sencillos pero útiles en el campo de la robótica. Primeramente, hablaremos de los clasificadores de distancia mínima. Enseguida, se presenta una extensión a los llamados clasificadores de Mahalanobis que permiten afrontar algunas deficiencias posiblemente presentes en los clasificadores de distancia mínima.

Finalmente, se presenta una familia de clasificadores que utilizando el teorema de Bayes permite resolver problemas de clasificación más complejos.

### 5.5.1 Clasificadores de cálculo de distancias

Uno de los enfoques más utilizados para la clasificación de patrones se basa en el cálculo de distancias. Sin pérdida de generalidad, una distancia es una medida entre dos puntos. Así, si las clases de pertenencia de los diferentes patrones se representan como puntos en un espacio $n$-dimensional de rasgos, la clase de pertenencia de un patrón dado $\boldsymbol{x}$ se determina como aquella clase $C^k$, $k = 1 \quad, \cdots, p$, para la cual la distancia entre $\boldsymbol{x}$ y $C^k$ sea la más pequeña.

Los métodos basados en este enfoque fueron los primeramente utilizados por la comunidad para resolver problemas relacionados con la clasificación de patrones. De manera muy general, la operación de un clasificador basado en el cálculo de una distancia se muestra en la figura 5.23. Como se puede apreciar de esta figura, una descripción $\boldsymbol{x}$ dada por la ecuación (5.35) de un objeto percibido, obtenida a partir de los captores conectados a un robot, se compara con todas y cada una de las descripciones $\boldsymbol{z}^1, \boldsymbol{z}^2, \cdots, \boldsymbol{z}^k, \cdots, \boldsymbol{z}^p$ almacenadas en la base de datos del cerebro (ordenador) del robot.

Después del análisis respectivo, el sistema de clasificación del robot responderá normalmente con el índice de clase más cercano al patrón de entrada. Esto lo hará al encontrar el mínimo de las $p$ distancias calculadas.

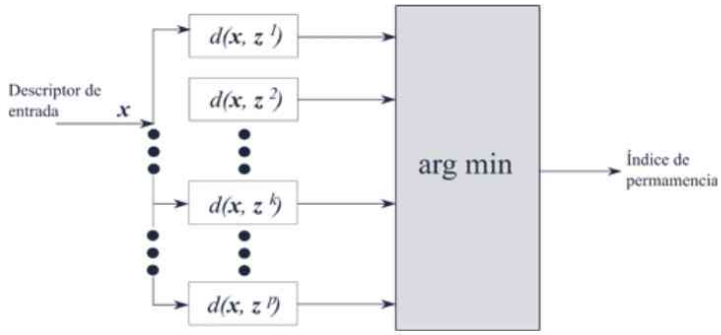



**Figura 5.23**: Diagrama de operación de los métodos basados en el cálculo de distancias.

A manera de ejemplo, en la figura 5.24 se ilustra el proceso antes mencionado con las descripciones de tres objetos sencillos. Como se puede ver, el sistema emite como salida el índice correspondiente a la segunda clase. Como alternativa, el sistema puede también responder con el nombre del objeto (barco) o incluso con la foto de este almacenada en la base de datos.

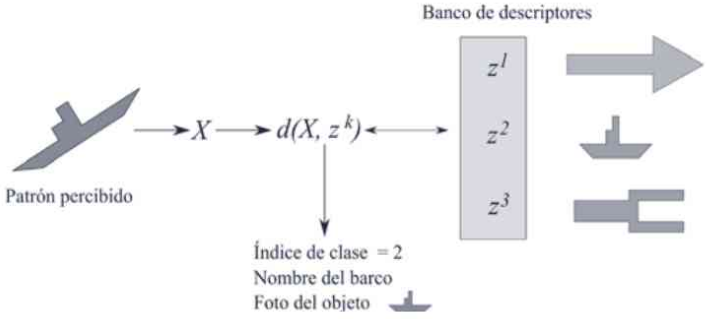



**Figura 5.24**: Ejemplo de operación de un método basado en el cálculo de distancias.

De la discusión anterior se puede apreciar que la operación de clasificación consiste en encontrar el índice de la clase $C^k$, $k = 1, 2, \cdots, p$ de pertenencia del objeto de entrada, descrito en términos de un vector de rasgos como el dado por la ecuación (5.35). De aquí en adelante, usaremos de manera indistinta los términos reconocedor y clasificador para designar la misma operación.

### 5.5.2 Clasificador de distancia mínima

Este clasificador, como ya se dijo, es el más utilizado en la literatura del reconocimiento de patrones. Parte del hecho de que las clases de patrones son linealmente separables (i. e., entre individuos de dos clases es posible encontrar una superficie de separación). El clasificador de distancia mínima supone que la pertenencia de cada patrón se conoce de antemano y que además se sabe el número de clases en las cuales el patrón puede ser clasificado.

Supónganse $p$ clases $C^k,\ k = 1, 2, \cdots, p$, personalizadas por $p$ representantes, es decir: $\boldsymbol{z}^k,\ k = 1, 2, \cdots, p$. Cada representante es un vector de la forma $\boldsymbol{z}^k = (\mu_1, \mu_2, \cdots, \mu_n)^T$.

Un patrón de entrada $\boldsymbol{z} = (r_1, r_2, \cdots, r_n)^T$ será clasificado en la clase $C^k,\ k = 1, 2, \cdots, p$ cuando se cumpla la condición que:

$$\underset{k}{\text{argmin}}\ d(\boldsymbol{x}, \boldsymbol{z}^k) \tag{5.36}$$

donde $d(\boldsymbol{x}, \boldsymbol{z}^k)$ es normalmente la distancia euclidiana:

$$d(\boldsymbol{x}, \boldsymbol{z}^k) \quad = \quad [\,\Sigma_{i=1}^{n}(r_i - z_i)^2\,]^{\frac{1}{2}} = \|\boldsymbol{x} - \boldsymbol{z}^k\|_2. \tag{5.37}$$

Esta distancia es calculada entre el vector descriptor de entrada $\boldsymbol{x}$ y cada uno de los representantes $\boldsymbol{z}^k,\ k = 1, 2, \cdots, p$ de cada clase $C^k,\ k = 1, 2, \cdots, p$.

**Ejemplo 5.19**

Las distancias entre el vector $\boldsymbol{x} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ y los vectores representantes: $\boldsymbol{z}^1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ y $\boldsymbol{z}^2 = \begin{bmatrix} 7 \\ 6 \end{bmatrix}$ son, respectivamente:

$$\begin{aligned} d_1(\boldsymbol{x}, \boldsymbol{z}^1) &= [\,(3-1)^2 + (4-1)^2\,]^{\frac{1}{2}} = 3.6055. \\ d_2(\boldsymbol{x}, \boldsymbol{z}^2) &= [\,(3-7)^2 + (4-6)^2\,]^{\frac{1}{2}} = 4.4721. \end{aligned}$$

**Comentario:** La distancia euclidiana es un caso especial de la distancia de Minkowsky:

$$d_{MI}(\boldsymbol{x}, \boldsymbol{z}^k) \quad = \quad [\,\Sigma_{i=1}^{n}(x_i - z_i^k)^{MI}\,]^{\frac{1}{MI}} = \|\boldsymbol{x} - \boldsymbol{z}^k\|_{MI}. \tag{5.38}$$

De acuerdo con la figura 5.23, note cómo el clasificador de distancia mínima requiere del cálculo de todas y cada una de las distancias entre el vector de entrada $\boldsymbol{x} = (r_1, r_2, \cdots, r_n)^T$ y los vectores descriptivos $\boldsymbol{z}^1, \boldsymbol{z}^2, \cdots, \boldsymbol{z}^k, \cdots, \boldsymbol{z}^p$. Por supuesto, si $p$ es muy grande, esta puede

ser una de las desventajas de los métodos basados en el cálculo de distancias. Este problema puede ser fácilmente resuelto al implementar la distancia euclidiana en una plataforma paralela de procesamiento.

### 5.5.3 Diseño del clasificador de distancia mínima

El diseño de un clasificador de distancia mínima, en particular de tipo euclidiano, incorporará cinco etapas [37]. Cada una de estas se detalla a continuación:

1. **Determinación del número de clases**. Esta etapa se realiza normalmente en forma manual. Prácticamente en todos los casos se sabe en cuántas clases los patrones que componen el universo de muestras pueden ser clasificados. Si no es así, se deberá recurrir a técnicas de agrupamiento (acumulación) automática [37].

2. **Elección y prueba de los rasgos descriptivos**. Esta etapa es fundamental para el buen desempeño del clasificador. En la selección de los rasgos descriptores se debe satisfacer al menos la condición de que por medio de la combinación de los rasgos se logre, por un lado, la diferenciación entre elementos de clases distintas y por otro la clasificación de patrones de una clase dada en esa propia clase.

3. Evaluación de la prueba determinista. Es bien sabido que el clasificador euclidiano es determinístico por naturaleza. Cada clase $C^k$ del universo de trabajo $U$ queda representada por un vector único $\boldsymbol{z}^k$, $k = 1, 2, \cdots, p$, denominado prototipo o representante de la clase. En el diseño de un clasificador euclidiano es necesario verificar el cumplimiento de esta hipótesis para cada clase dentro del universo de trabajo del clasificador. Si esta condición no se cumple al menos para una de las clases, se debería proceder a usar otro clasificador. Para la verificación de la hipótesis mencionada, para cada una de las clases se supone que de alguna manera se han obtenido $m$ muestras o realizaciones $\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^m$ del mismo vector de rasgos para la misma clase. Una manera de verificar la hipótesis determinista para una clase $C_k$, $i = 1, 2, \cdots, n$, consiste en analizar su matriz de covarianza $MC(C_k)$, definida como:

$$
\begin{aligned}
MC(C^k) &= E\left\{[\,\boldsymbol{x}-E[\boldsymbol{x}]\,]^T \times [\,\boldsymbol{x}-E[\boldsymbol{x}]\,]\right\} \\
&= E\begin{bmatrix} (x_1-\bar{x}_1)^2 & (x_1-\bar{x}_1)(x_2-\bar{x}_2) & \cdots & (x_1-\bar{x}_1)(x_n-\bar{x}_n) \\ (x_2-\bar{x}_2)(x_1-\bar{x}_1) & (x_2-\bar{x}_2)^2 & \cdots & (x_2-\bar{x}_2)(x_n-\bar{x}_n) \\ \vdots & \vdots & \cdots & \vdots \\ (x_n-\bar{x}_n)(x_1-\bar{x}_1) & (x_n-\bar{x}_n)(x_2-\bar{x}_2) & \cdots & (x_n-\bar{x}_n)^2 \end{bmatrix}. \quad (5.39)
\end{aligned}
$$

En este caso $E$ es el operador esperanza matemática y $E(x_j) = \bar{x}_j$. Se sabe que la matriz de covarianza es simétrica y que en su diagonal principal contiene las varianzas de cada rasgo de la clase, mientras que el resto de los elementos se corresponden con las covarianzas de los mismos rasgos.

Otro parámetro necesario para verificar la hipótesis determinista del clasificador de distancia mínima es la media aritmética de la clase $\bar{\mu}^k$, en este caso particular coincide con la estimación de la media estadística. Si se disponen de $M$ muestras (vectores) para cada clase $\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^M$, con los siguientes elementos dados por $\boldsymbol{x}^j = (r_j 1, r_j 2, \ldots, r_j n)^T$ :

$$\bar{\mu}^k = (\bar{\mu}_1, \bar{\mu}_2, \cdots, \bar{\mu}_n)^T. \quad (5.40)$$

Cada componente $\bar{\mu}_i$ de $\bar{\mu}^k$ se calcula como:

$$\bar{\mu}_i = \frac{1}{M}\sum_{j=1}^{M} r_{ij}. \quad (5.41)$$

Obsérvese que este vector de tamaño $n$ coincide con $\boldsymbol{z}^k$, $k = 1, 2, \cdots, p$ representante de la clase $C^k, k = 1, 2, \cdots, p$. Un criterio experimental para validar la hipótesis determinista es que la desviación estándar de cada rasgo no se vaya más allá de 10 % de su correspondiente promedio, esto es:

$$\frac{\sigma_{ki}}{\bar{\mu}_{ki}}, \quad k = 1, 2, \cdots, N; \quad i = 1, 2, \cdots, n. \quad (5.42)$$

Puede suceder que una o más de las desviaciones estándar rebasen el 10 % estipulado en la ecuación (5.42). En este caso si los representantes de las clases se encuentran bien separados entre ellos, como se muestra en la figura 5.25, entonces la hipótesis será aceptada. Respecto a la figura se puede ver que, aunque los cúmulos para las muestras "•" y "+" presentan desviaciones estándar más grandes que la observada para el cúmulo de muestras tipo "*", su separación es grande. Esto permite aceptar la prueba determinística.

$$
\begin{aligned}
MC(C^k) &= E\left\{[\,\boldsymbol{x}-E[\boldsymbol{x}]\,]^T \times [\,\boldsymbol{x}-E[\boldsymbol{x}]\,]\right\} \\
&= E\begin{bmatrix} (x_1-\bar{x}_1)^2 & (x_1-\bar{x}_1)(x_2-\bar{x}_2) & \cdots & (x_1-\bar{x}_1)(x_n-\bar{x}_n) \\ (x_2-\bar{x}_2)(x_1-\bar{x}_1) & (x_2-\bar{x}_2)^2 & \cdots & (x_2-\bar{x}_2)(x_n-\bar{x}_n) \\ \vdots & \vdots & \cdots & \vdots \\ (x_n-\bar{x}_n)(x_1-\bar{x}_1) & (x_n-\bar{x}_n)(x_2-\bar{x}_2) & \cdots & (x_n-\bar{x}_n)^2 \end{bmatrix}. \quad (5.39)
\end{aligned}
$$

En este caso $E$ es el operador esperanza matemática y $E(x_j) = \bar{x}_j$. Se sabe que la matriz de covarianza es simétrica y que en su diagonal principal contiene las varianzas de cada rasgo de la clase, mientras que el resto de los elementos se corresponden con las covarianzas de los mismos rasgos.

Otro parámetro necesario para verificar la hipótesis determinista del clasificador de distancia mínima es la media aritmética de la clase $\bar{\mu}^k$, en este caso particular coincide con la estimación de la media estadística. Si se disponen de $M$ muestras (vectores) para cada clase $\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^M$, con los siguientes elementos dados por $\boldsymbol{x}^j = (r_j 1, r_j 2, \ldots, r_j n)^T$ :

$$\bar{\mu}^k = (\bar{\mu}_1, \bar{\mu}_2, \cdots, \bar{\mu}_n)^T. \quad (5.40)$$

Cada componente $\bar{\mu}_i$ de $\bar{\mu}^k$ se calcula como:

$$\bar{\mu}_i = \frac{1}{M}\sum_{j=1}^{M} r_{ij}. \quad (5.41)$$

Obsérvese que este vector de tamaño $n$ coincide con $\boldsymbol{z}^k$, $k = 1, 2, \cdots, p$ representante de la clase $C^k, k = 1, 2, \cdots, p$. Un criterio experimental para validar la hipótesis determinista es que la desviación estándar de cada rasgo no se vaya más allá de 10 % de su correspondiente promedio, esto es:

$$\frac{\sigma_{ki}}{\bar{\mu}_{ki}}, \quad k = 1, 2, \cdots, N; \quad i = 1, 2, \cdots, n. \quad (5.42)$$

Puede suceder que una o más de las desviaciones estándar rebasen el 10 % estipulado en la ecuación (5.42). En este caso si los representantes de las clases se encuentran bien separados entre ellos, como se muestra en la figura 5.25, entonces la hipótesis será aceptada. Respecto a la figura se puede ver que, aunque los cúmulos para las muestras "•" y "+" presentan desviaciones estándar más grandes que la observada para el cúmulo de muestras tipo "*", su separación es grande. Esto permite aceptar la prueba determinística.

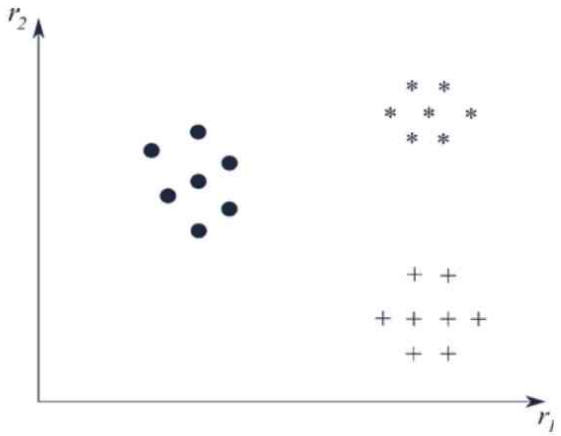



**Figura 5.25**: Ejemplo visual para el cual algunos rasgos no cumplen con el criterio dado por la ecuación (5.42), pero sí están bien separados entre ellos.

4. **Cálculo de los vectores prototipo**. Una vez demostrada la hipótesis determinista para todas las clases, el paso siguiente consiste en el cómputo de los vectores prototipos de cada clase. Estos son calculados como sigue:

$$\boldsymbol{z}^k = \frac{1}{m}\sum_{j=1}^{m}\boldsymbol{x}^j. \tag{5.43}$$

   Al conjunto de muestras $\boldsymbol{x}^1, 1, \boldsymbol{x}^1, 2, \cdots, \boldsymbol{x}(1, m)$, con $\boldsymbol{x}^{(k,j)} = (r_{kj1}, r_{kj2}, \cdots, r_{kjn})^T$, que se utilizan para el cálculo de los prototipos (entrenamiento del clasificador), se le denomina conjunto de entrenamiento.

5. **Prueba del clasificador**. Una vez determinado el número de clases, elegido y validado el conjunto de rasgos descriptores, aceptada la hipótesis determinista con los calculados prototipos, se procede a probar el desempeño del clasificador. En este caso se usan vectores de rasgos descriptores de los objetos, pero diferentes a los utilizados para entrenar el clasificador.

   En este caso, un vector descriptor $\boldsymbol{x} = (r_1, r_2, \cdots, r_n)^T$ es presentado al clasificador. Enseguida se calcula la distancia de este vector a todos y cada uno de los prototipos $\boldsymbol{z}^k$ de las $p$ clases. Finalmente, el patrón $\boldsymbol{x}$ es asignado a la clase $C$ para la cual, como ya vimos al comienzo de esta sección, se cumpla la ecuación (5.38).

**•• Ejemplo 5.20**

Supóngase el siguiente universo de patrones y sus correspondientes tres clases:

$$\left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}\right\} \in C_1$$

$$\left\{\begin{bmatrix}5\\5\end{bmatrix}, \begin{bmatrix}5\\6\end{bmatrix}, \begin{bmatrix}4\\5\end{bmatrix}, \begin{bmatrix}6\\5\end{bmatrix}, \begin{bmatrix}6\\4\end{bmatrix}\right\} \in C_2$$

$$\left\{\begin{bmatrix}9\\12\end{bmatrix}, \begin{bmatrix}10\\12\end{bmatrix}, \begin{bmatrix}10\\11\end{bmatrix}, \begin{bmatrix}9\\10\end{bmatrix}, \begin{bmatrix}11\\9\end{bmatrix}\right\} \in C_3$$

Calcular los prototipos o representantes de cada una de las tres clases y establecer los respectivos clasificadores.

**Solución**

De la figura 5.26a, el lector puede apreciar la visible separabilidad lineal entre los elementos de las tres clases. Esto nos permite asegurar que con este número de rasgos el problema de separabilidad lineal entre clases queda resuelto, indicándonos a la vez que los rasgos seleccionados de alguna manera cumplen con lo estipulado en la etapa 2 del procedimiento de diseño de un clasificador de distancia mínima.

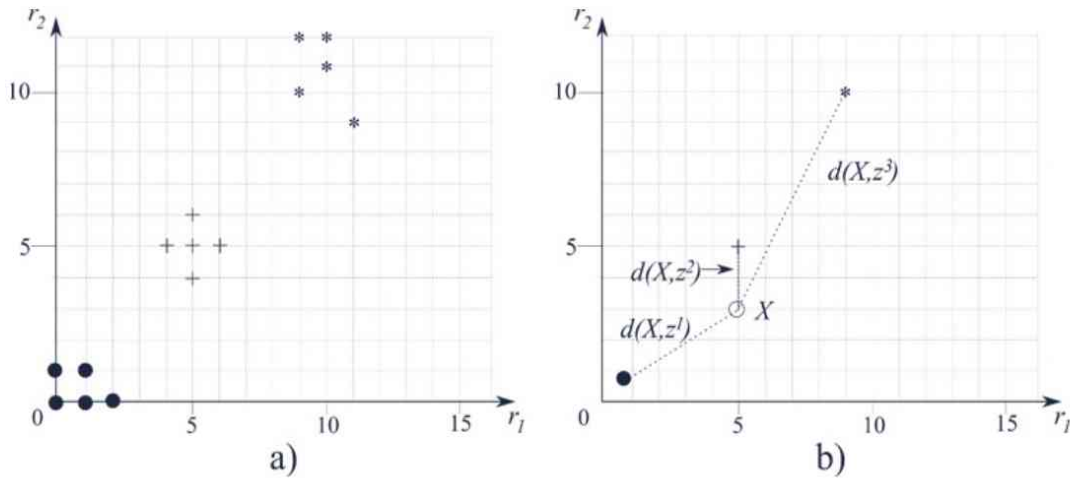



**Figura 5.26**: (a) Patrones pertenecientes a las tres clases del ejemplo, representados, respectivamente, como "•" y "+" y "*" en el plano de rasgos $r_1$ y $r_2$. (b) Distancia del patrón $\boldsymbol{x} = \begin{bmatrix}5\\3\end{bmatrix}$ a las tres clases: $C^1, C^2$ y $C^3$.

Por otro lado, para facilitar la explicación se supondrá por el momento que las tres clases satisfacen el criterio determinista dado por el paso 3 del procedimiento de entrenamiento para el clasificador de distancia mínima. Para solucionar el ejemplo planteado, se requiere primeramente del cálculo de los prototipos o representantes de cada una de las tres clases. Así entonces, de acuerdo con la ecuación (5.43), los prototipos para las tres clases son los siguientes:

$$\begin{aligned} z^1 &= \frac{1}{5}\left[\begin{bmatrix}0\\0\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix}+\begin{bmatrix}2\\1\end{bmatrix}\right]=\begin{bmatrix}0.8\\0.6\end{bmatrix} \\ z^2 &= \frac{1}{5}\left[\begin{bmatrix}5\\6\end{bmatrix}+\begin{bmatrix}4\\5\end{bmatrix}+\begin{bmatrix}5\\5\end{bmatrix}+\begin{bmatrix}6\\5\end{bmatrix}+\begin{bmatrix}5\\4\end{bmatrix}\right]=\begin{bmatrix}5.0\\5.0\end{bmatrix} \\ z^3 &= \frac{1}{5}\left[\begin{bmatrix}9\\12\end{bmatrix}+\begin{bmatrix}10\\12\end{bmatrix}+\begin{bmatrix}10\\11\end{bmatrix}+\begin{bmatrix}9\\10\end{bmatrix}+\begin{bmatrix}11\\9\end{bmatrix}\right]=\begin{bmatrix}9.8\\10.8\end{bmatrix}. \end{aligned}$$

**Prueba del clasificador**. Supóngase el siguiente vector $\boldsymbol{x} = \begin{bmatrix}5\\3\end{bmatrix}$ obtenido como una medición de la clase $C^2$ de objetos. Nótese que la nueva medición es diferente a las usadas para calcular los representantes de clase. De acuerdo con las ecuaciones (5.38) y (5.37):

$$\begin{aligned} d(\boldsymbol{x},\boldsymbol{z}^1) &= [(5.0-0.8)^2+(3.0-0.6)^2]^{\frac{1}{2}}=4.837 \\ d(\boldsymbol{x},\boldsymbol{z}^2) &= [(5.0-5.0)^2+(3.0-5.0)^2]^{\frac{1}{2}}=2.0 \\ d(\boldsymbol{x},\boldsymbol{z}^3) &= [(5.0-9.8)^2+(3.0-10.8)^2]^{\frac{1}{2}}=9.158. \end{aligned}$$

Debido a que el $\underset{k}{\text{argmin}}\ \{4.837, 2.507, 9.158\} = 2$, entonces la descripción de entrada $\boldsymbol{x} = [5 \quad 3]^T$ debe ser considerada como perteneciente a la clase $C^2$. Una manera gráfica de ver este resultado se muestra en la figura 5.26b, en la cual los representantes de las tres clases aparecen representados en sus respectivas coordenadas. En esta misma figura aparece representado el vector $\boldsymbol{x}$, donde se puede ver que su distancia más corta es hacia el prototipo número 2. Por tanto, el patrón de entrada $\boldsymbol{x}$ es asignado a la clase $C^2$.

### • • • Ejemplo 5.21

Suponga el conjunto de entrenamiento del ejemplo 5.20, verificar si este universo satisface la prueba determinista.

### Solución

De acuerdo con el ejemplo 5.20, los vectores promedio para cada una de las tres clases son:

$$\bar{\mu}_1 = \begin{bmatrix}0.8\\0.6\end{bmatrix},\ \bar{\mu}_2 = \begin{bmatrix}5.0\\5.0\end{bmatrix}\ \text{y}\ \bar{\mu}_3 = \begin{bmatrix}9.8\\10.8\end{bmatrix}.$$

Para poder comprobar si el universo propuesto satisface la prueba determinista, es necesario calcular la desviación estándar para cada rasgo. La desviación estándar $\sigma_{ki}$ para cada rasgo $r_i$ para cada clase $C^k$, puede ser calculada mediante la siguiente expresión:

$$\sigma_{ki} = \left[\frac{1}{M}\sum_{j=1}^{M}(r_{ij}-\bar{\mu}_{ij})^2\right]^{\frac{1}{2}}. \tag{5.44}$$

Prueba determinística para la clase $C_1$:

$$\begin{aligned}\sigma_{11} &= \left[\frac{1}{5}\sum_{j=1}^{5}(x_{1j}-0.8)^2\right]^2 \\ &= \left[\frac{1}{5}\left[(0-0.8)^2+(0-0.8)^2+(1-0.8)^2+(1-0.8)^2+(2-0.8)^2\right]\right]^2 \\ &= \left[\frac{1}{5}[0.64+0.64+0.04+0.04+1.44]\right]^2 = 0.7483\end{aligned}$$

$\frac{\sigma_{11}}{\bar{x}_{1j}} = \frac{0.7483}{0.8} = 0.9353$ no satisface la prueba determinista

$$\begin{aligned}\sigma_{12} &= \left[\frac{1}{5}\sum_{j=1}^{5}(x_{2j}-0.6)^2\right]^2 \\ &= \left[\frac{1}{5}\left[(0-0.6)^2+(0-0.6)^2+(1-0.6)^2+(1-0.6)^2+(1-0.6)^2\right]\right]^2 \\ &= \left[\frac{1}{5}[0.36+0.16+0.16+0.36+0.16]\right]^2 = 0.4898.\end{aligned}$$

Observe que $\frac{\sigma_{12}}{\bar{x}_{2j}} = \frac{0.4898}{0.6} = 0.8164$ no satisface la prueba determinista

Prueba determinística para la clase $C_2$:

$$\begin{aligned}\sigma_{21} &= \left[\frac{1}{5}\sum_{j=1}^{5}(x_{2j}-5)^2\right]^2 \\ &= \left[\frac{1}{5}\left[(5-5)^2+(4-5)^2+(5-5)^2+(6-5)^2+(5-5)^2\right]\right]^2 \\ &= \left[\frac{1}{5}[0+0+1+0+1]\right]^2 = 0.6324.\end{aligned}$$

$\frac{\sigma_{22}}{\bar{x}_{2j}} = \frac{0.6324}{5} = 0.1216$ no satisface la prueba determinista.

$$\begin{aligned}
\sigma_{22} &= \left[\frac{1}{5}\sum_{j=1}^{5}(x_{2j}-5)^2\right]^2 \\
&= \left[\frac{1}{5}\left[(6-5)^2+(5-5)^2+(5-5)^2+(5-5)^2+(4-5)^2\right]\right]^2 \\
&= \left[\frac{1}{5}[1+0+0+0+1]\right]^2 = 0.4898.
\end{aligned}$$

$\frac{\sigma_{22}}{\bar{x}_{2j}} = \frac{0.6324}{5} = 0.1216$ no satisface la prueba determinista.

Prueba determinística para la clase $C_3$:

$$\begin{aligned}
\sigma_{31} &= \left[\frac{1}{5}\sum_{j=1}^{5}(x_{3j}-9.8)^2\right]^2 \\
&= \left[\frac{1}{5}\left[(9-9.8)^2+(10-9.8)^2+(10-9.8)^2+(9-9.8)^2+(11-9.8)^2\right]\right]^2 \\
&= \left[\frac{1}{5}[0.64+0.04+0.04+0.64+1.44]\right]^2 = 0.7483.
\end{aligned}$$

$\frac{\sigma_{31}}{\bar{x}_{3j}} = \frac{0.7483}{9.8} = 0.0763$ satisface la prueba determinista.

$$\begin{aligned}
\sigma_{32} &= \left[\frac{1}{5}\sum_{j=1}^{5}(x_{2j}-10.8)^2\right]^2 \\
&= \left[\frac{1}{5}\left[(12-10.8)^2+(12-10.8)^2+(11-10.8)^2+(10-10.8)^2+(9-10.8)^2\right]\right]^2 \\
&= \left[\frac{1}{5}[1.44+1.44+0.04+0.64+3.24]\right]^2 = 1.1661
\end{aligned}$$

$\frac{\sigma_{32}}{\bar{x}_{2j}} = \frac{1.1661}{10.8} = 1.1079$ no satisface la prueba determinista.

En resumen, como se puede ver, solo las clases segunda y tercera en su segundo y primer rasgo, respectivamente, satisfacen la prueba determinista. Sin embargo, ya que los representantes de las tres clases se encuentran bien separados, la prueba determinista para el universo de patrones tratado puede ser aceptada.

### 5.5.4 Clasificador de Mahalanobis

Este clasificador basa su operación en el cómputo de la llamada distancia de Mahalanobis. La distancia de Mahalanobis entre un patrón: $\boldsymbol{x} = [r_1, r_2, \cdots, r_n]^T$ a una clase $C$ caracterizada mediante un vector prototipo promedio $\bar{\boldsymbol{x}}$ y matriz de covarianza $\Sigma(C)$, se define como:

$$d_M(\boldsymbol{x}, C) \quad = \quad (\boldsymbol{x} - \bar{\boldsymbol{x}})^T \Sigma^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}})^T. \tag{5.45}$$

Si se supone que la matriz $\sum(C)$ es diagonal, cosa que sucede solo en caso de que los elementos de vector descriptor sean estadísticamente independientes, entonces:

$$\sum \quad = \quad \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{bmatrix}. \tag{5.46}$$

Por tanto:

$$d_M(\boldsymbol{x}, C) \quad = \quad (\boldsymbol{x} - \bar{\boldsymbol{x}})^T \Sigma^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}})^T = \frac{(\, x_1 - \bar{x}_1 \,)^2}{\sigma_1^2} + \cdots + \frac{(\, x_1 - \bar{x}_n \,)^2}{\sigma_n^2} \tag{5.47}$$

En este caso:

$$\sigma_i^2 \quad = \quad \frac{1}{m} \sum_{j=1}^{m} (\, x_{ij} - \bar{x}_{ij} \,)^2 \,. \tag{5.48}$$

Note cómo en este caso particular en que la matriz de covarianza es diagonal, la distancia de Mahalanobis coincide con el cuadrado de la distancia euclidiana dada por la ecuación (5.38), pero ponderada por la varianza de cada rasgo.

Una vez entrenado el clasificador, un vector de entrada es puesto en la clase para la cual:

$$\underset{k}{\operatorname{argmin}}\, d_M(\boldsymbol{x}, C^k). \tag{5.49}$$

**Explicación intuitiva de la operación de la distancia de Mahalanobis**. Para entender la operación de la distancia de Mahalanobis, consideremos el problema de que un patrón descriptivo $\boldsymbol{x}$ pertenezca a una clase dada. Como primera aproximación supongamos que podemos calcular la distancia del vector $\boldsymbol{x}$ al centro $\boldsymbol{z}$ de la clase $C$. Sea esta distancia $d(\boldsymbol{x}, \boldsymbol{z})$. Ahora bien, entre más pequeña es la distancia $d$ entre $\boldsymbol{x}$ y $\boldsymbol{z}$, es más probable que $\boldsymbol{x}$ pertenezca a la clase $C$. Este hecho se puede visualizar en la figura 5.27a.

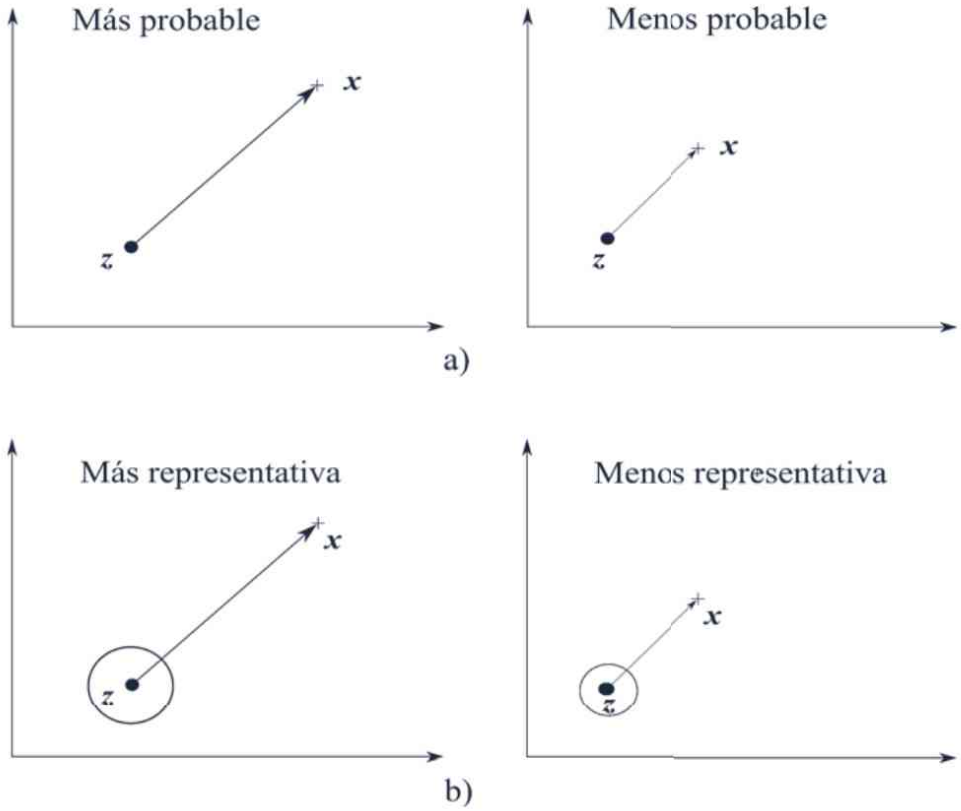



**Figura 5.27**: Dispersión de clase. Explicación intuitiva de la operación de la distancia de Mahalanobis respecto a la distancia euclideana. (a) Distancia entre dos puntos sin tomar en cuenta la desviación estándar de la clase $C$. (b) Distancia entre dos puntos al tomar en cuenta la desviación estándar de clase $C$.

Ahora bien, nos gustaría saber también qué tanto se dispersa la clase $C$ en el espacio (mucho o poco), de tal forma que podamos decir que una distancia $d$ entre $\boldsymbol{x}$ y $\boldsymbol{z}$ es más o menos representativa. Una manera sencilla de medir esto consiste en estimar la desviación estándar $\sigma$ entre los elementos de la clase $C$.

Luego, entonces si la distancia $d$ entre el punto $\boldsymbol{x}$ y el punto $\boldsymbol{z}$ es menor a $\sigma$ podemos concluir que es más probable que $\boldsymbol{x}$ pertenezca a $C$. Entre menos suceda esto, debemos concluir que $\sigma$ no debe ser clasificado en $C$. Este hecho se ilustra en la figura 5.27. Para entender todavía mejor lo visto consideremos los siguientes ejemplos numéricos.

Al igual que para el caso del clasificador euclidiano, enseguida se da un ejemplo numérico que permitirá al lector reforzar el conocimiento aprendido hasta el momento.

**• • • Ejemplo 5.22**

Supóngase el siguiente conjunto de entrenamiento y sus clases:

$$\left\{\begin{bmatrix}5\\8\end{bmatrix}, \begin{bmatrix}2\\5\end{bmatrix}, \begin{bmatrix}5\\5\end{bmatrix}, \begin{bmatrix}8\\5\end{bmatrix}, \begin{bmatrix}5\\2\end{bmatrix}\right\} \in C_1$$
$$\left\{\begin{bmatrix}13\\11\end{bmatrix}, \begin{bmatrix}12\\10\end{bmatrix}, \begin{bmatrix}13\\10\end{bmatrix}, \begin{bmatrix}14\\10\end{bmatrix}, \begin{bmatrix}13\\9\end{bmatrix}\right\} \in C_2$$

Establecer el entrenamiento del clasificador y realizar su respectiva prueba.

**Solución**

Estos dos conjuntos de patrones aparecen representados en la figura 5.28.

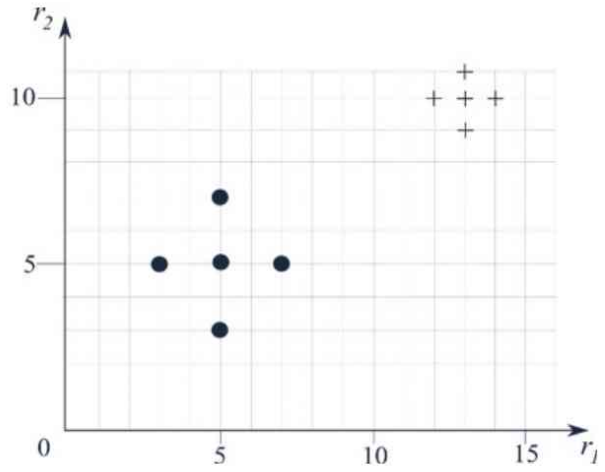



**Figura 5.28:** Patrones pertenecientes a las dos clases del ejemplo, representados, respectivamente, como"•" y "+" en el plano de rasgos $r_1$ y $r_2$.

**Entrenamiento del clasificador**. De acuerdo con la ecuación (5.46) para el entrenamiento del clasificador de Mahalanobis es necesario calcular los vectores de medias, así como las varianzas para cada clase.

Procedamos:

$$\boldsymbol{x}^1 = \frac{1}{5}\left[\begin{pmatrix}5\\8\end{pmatrix} + \begin{pmatrix}2\\5\end{pmatrix} + \begin{pmatrix}5\\5\end{pmatrix} + \begin{pmatrix}8\\5\end{pmatrix} + \begin{pmatrix}5\\2\end{pmatrix}\right] = \begin{bmatrix}5\\5\end{bmatrix}$$
$$\boldsymbol{x}^2 = \frac{1}{5}\left[\begin{pmatrix}13\\11\end{pmatrix} + \begin{pmatrix}12\\10\end{pmatrix} + \begin{pmatrix}13\\10\end{pmatrix} + \begin{pmatrix}14\\10\end{pmatrix} + \begin{pmatrix}13\\9\end{pmatrix}\right] = \begin{bmatrix}13\\10\end{bmatrix}.$$

**• • • Ejemplo 5.22**

Supóngase el siguiente conjunto de entrenamiento y sus clases:

$$\left\{\begin{bmatrix}5\\8\end{bmatrix}, \begin{bmatrix}2\\5\end{bmatrix}, \begin{bmatrix}5\\5\end{bmatrix}, \begin{bmatrix}8\\5\end{bmatrix}, \begin{bmatrix}5\\2\end{bmatrix}\right\} \in C_1$$

$$\left\{\begin{bmatrix}13\\11\end{bmatrix}, \begin{bmatrix}12\\10\end{bmatrix}, \begin{bmatrix}13\\10\end{bmatrix}, \begin{bmatrix}14\\10\end{bmatrix}, \begin{bmatrix}13\\9\end{bmatrix}\right\} \in C_2$$

Establecer el entrenamiento del clasificador y realizar su respectiva prueba.

**Solución**

Estos dos conjuntos de patrones aparecen representados en la figura 5.28.

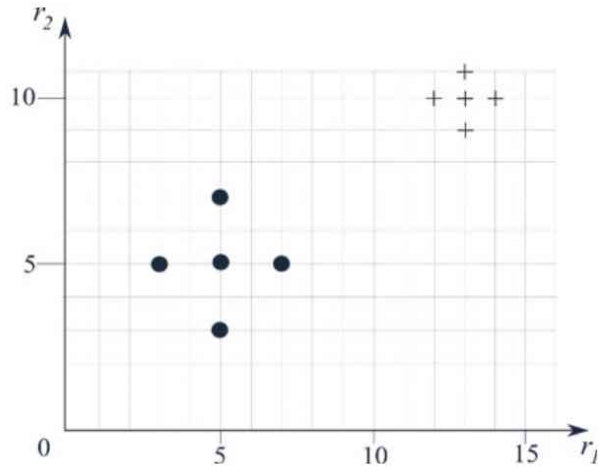



**Figura 5.28:** Patrones pertenecientes a las dos clases del ejemplo, representados, respectivamente, como"•" y "+" en el plano de rasgos $r_1$ y $r_2$.

**Entrenamiento del clasificador**. De acuerdo con la ecuación (5.46) para el entrenamiento del clasificador de Mahalanobis es necesario calcular los vectores de medias, así como las varianzas para cada clase.

Procedamos:

$$\boldsymbol{x}^1 = \frac{1}{5}\left[\begin{pmatrix}5\\8\end{pmatrix} + \begin{pmatrix}2\\5\end{pmatrix} + \begin{pmatrix}5\\5\end{pmatrix} + \begin{pmatrix}8\\5\end{pmatrix} + \begin{pmatrix}5\\2\end{pmatrix}\right] = \begin{bmatrix}5\\5\end{bmatrix}$$

$$\boldsymbol{x}^2 = \frac{1}{5}\left[\begin{pmatrix}13\\11\end{pmatrix} + \begin{pmatrix}12\\10\end{pmatrix} + \begin{pmatrix}13\\10\end{pmatrix} + \begin{pmatrix}14\\10\end{pmatrix} + \begin{pmatrix}13\\9\end{pmatrix}\right] = \begin{bmatrix}13\\10\end{bmatrix}.$$

De la figura 5.28, nótese que los patrones en cada clase aparecen distribuidos con la misma forma en las dos dimensiones pero con tamaños diferentes. Se debe esperar que la varianza para las dos clases es igual para cada rasgo pero diferente para las dos clases. Veamos:

$$\begin{aligned}
\sigma^2_{C^1} &= \tfrac{1}{5}\left\{\left[\binom{5}{8}-\binom{5}{5}\right]^2+\left[\binom{2}{5}-\binom{5}{5}\right]^2+\left[\binom{5}{5}-\binom{5}{5}\right]^2+\left[\binom{8}{5}-\binom{5}{5}\right]^2+\left[\binom{5}{2}-\binom{5}{5}\right]^2\right\} \\
&= \frac{1}{5}\left[\binom{0}{9}+\binom{9}{0}+\binom{0}{0}+\binom{9}{0}+\binom{0}{9}\right]=\begin{bmatrix}3.6\\3.6\end{bmatrix} \\
\sigma^2_{C^2} &= \tfrac{1}{5}\left\{\left[\binom{13}{11}-\binom{13}{10}\right]^2+\left[\binom{12}{10}-\binom{13}{10}\right]^2+\left[\binom{13}{10}-\binom{13}{10}\right]^2+\left[\binom{14}{10}-\binom{13}{10}\right]^2+\left[\binom{13}{9}-\binom{13}{10}\right]^2\right\} \\
&= \frac{1}{5}\left[\binom{0}{1}+\binom{1}{0}+\binom{0}{0}+\binom{1}{0}+\binom{0}{1}\right]=\begin{bmatrix}0.4\\0.4\end{bmatrix}.
\end{aligned}$$

Note que el tamaño entre la varianza entre la clase $C^1$ y la clase $C^2$ es exactamente de 9, esto es 3.6/0.4.

**Prueba del clasificador**. Supóngase el siguiente vector $\boldsymbol{x} = [10 \quad 8]^T$ obtenido como una medición ruidosa de la clase $C^1$ de objetos. Al aplicar las ecuaciones (5.47) y (5.49 ):

$$\begin{aligned}
d_M(\boldsymbol{x}, C^1) &= \frac{(10-5)^2}{3.6}+\frac{(8-5)^2}{3.6}=6.9444+2.5=9.4444 \\
d_M(\boldsymbol{x}, C^2) &= \frac{(10-13)^2}{0.43}+\frac{(8-10)^2}{0.4}=22.5+10=32.5 \\
\underset{j}{\text{argmin}}\ \{9.4444,\ 32.5\} &= 1
\end{aligned}$$

Debido a esto, el patrón de entrada $\boldsymbol{x} = [10 \quad 8]^T$ es correctamente clasificado en la clase $C^1$, que es lo que se esperaba. Si se usara el clasificador euclidiano, se obtendría lo siguiente:

$$\begin{aligned}
d_M(\boldsymbol{x}, \boldsymbol{z}^1) &= \left[(10-5)^2+(8-5)^2\right]^{\frac{1}{2}}=5.916 \\
d_M(\boldsymbol{x}, \boldsymbol{z}^2) &= \left[(10-13)^2+(8-10)^2\right]^{\frac{1}{2}}=3.6055 \\
\underset{j}{\text{argmin}}\ \{5.9160,\ 3.6055\} &= 2.
\end{aligned}$$

Note que en este caso el patrón de entrada es erróneamente clasificado en la clase $C^2$. Esto se debe, por un lado, a la cercanía entre el vector $[10 \quad 8]^T$ a la clase $C^2$ y por otro lado, al no considerar el efecto normalizador por las desviaciones estándar de las clases.

En conclusión, debemos considerar que si las clases presentan desviaciones estándar diferentes y evidentes, en lugar del clasificador basado en la distancia euclidiana se debería optar por el clasificador con base en la distancia de Mahalanobis.

### 5.5.5 Clasificador estadístico de Bayes

En esta sección estudiaremos uno de los clasificadores más utilizados por la comunidad científica para resolver problemas de clasificación de patrones. Se trata del clasificador estadístico de Bayes, cuya base de operación es el teorema de Bayes.

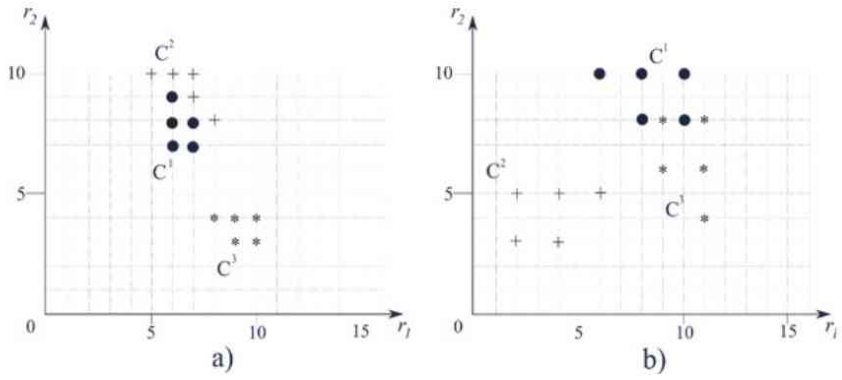



**Figura 5.29**: (a) Ejemplo de clases cercanas. (b) Ejemplo de clases separadas pero con dispersión significativa.

Como ya se trató en la subsección 5.5.3, el clasificador euclidiano es muy útil cuando las clases de patrones presentan distribuciones similares y una buena separación entre elementos de clases diferentes o cuando a pesar de estar cerca una de otras, sus desviaciones son pequeñas. En aquellos casos en los cuales los patrones de algunas clases aparecen muy cerca (figura 5.29a) o una dispersión significativa con respecto a su media (figura 5.29b), es conveniente abandonar la hipótesis determinística a cambio de una más de tipo estadístico. El primer caso se puede deber, entre otras cosas, a la alta correlación entre las clases o por los rasgos usados para describir dichas clases. El segundo caso puede deberse no solo a la correlación entre patrones o el tipo de rasgos usados, sino también a su alta variación.

En la figura 5.29a, nótese cómo debido a la cercanía entre elementos de la clase $C^1$ y $C^2$ ocurre un traslape aparente entre valores. Por otro lado, el traslape entre elementos de las clases $C^1$ y $C^3$ de la figura 5.29b, por el contrario, se debe a la alta dispersión entre los valores de dichas clases. En ambos casos, como ya se dijo, se debe adoptar la hipótesis estadística en lugar de la determinística.

En este capítulo se estudiará el conocido clasificador de Bayes, así como algunas de sus variantes. Se darán algunos ejemplos para su mejor comprensión y aplicación.

Enseguida estudiaremos el teorema de Bayes sobre el cual los clasificadores bayesianos fundamentan su operación.

**Teorema 5.1: Bayes**

El teorema de Bayes puede expresarse de varias formas. Una de estas es la siguiente. Si:

$p\left(C^k|\boldsymbol{x}\right)$ representa la probabilidad de que un patrón $\boldsymbol{x}$ pertenezca a la clase $C^k$.

$p\left(\boldsymbol{x}|C^k\right)$ significa la probabilidad de que dada la clase $C^k$, el valor de la variable aleatoria sea precisamente $\boldsymbol{x}$.

$p\left(C^k\right)$ es la probabilidad a priori de que se manifieste un elemento de la clase $C^k$.

$p\left(\boldsymbol{x}\right)$ significa la probabilidad a priori de que se manifieste un patrón $\boldsymbol{x}$.

Entonces:

$$p\left(C^k|\boldsymbol{x}\right) = \frac{p\left(\boldsymbol{x}|C^k\right) \times p\left(C^k\right)}{p\left(\boldsymbol{x}\right)}. \tag{5.50}$$

Nótese que el término $p(\boldsymbol{x})$ es común a todas las clases, por tanto puede eliminarse.

Para determinar la clase a la cual debe ser asignado un patrón de entrada $\boldsymbol{x}$, se puede usar tanto el término a la izquierda de la ecuación (5.50) como el término de la derecha.

En el primer caso, dado un patrón $\boldsymbol{x}$, su clase se puede determinar al realizar la siguiente operación:

$$\boldsymbol{x} \longrightarrow C^k \quad \text{si y solo si} \quad p\left(C^k|\boldsymbol{x}\right) > p\left(C^l|\boldsymbol{x}\right), \ \ \forall k \neq l, \ j = 1, 2, \cdots, p \tag{5.51}$$

En otras palabras, el patrón $\boldsymbol{x}$ es asignado a la clase $C^k$ para la cual su probabilidad a posteriori es máxima.

En el segundo caso, dado un patrón $\boldsymbol{x}$, su clase se puede determinar al llevar a cabo el siguiente cálculo:

$$\boldsymbol{x} \longrightarrow C^k \quad \text{si y solo si} \quad p\left(\boldsymbol{x}|C^k\right) \cdot p\left(C^k\right) > p\left(\boldsymbol{x}|C^l\right) \cdot p\left(C^l\right), \ \forall k \neq l, \ j = 1, 2, \cdots, p \tag{5.52}$$

En otras palabras, el patrón $\boldsymbol{x}$ es asignado a la clase $C^k$ para la cual su probabilidad a priori es máxima. Respecto a estas dos ecuaciones es pertinente observar lo siguiente:

1. Para usar la ecuación (5.51) es necesario calcular o al menos estimar la probabilidad de que el patrón $\boldsymbol{x}$ pertenezca a la clase $C^k$.

2. Para el caso de la ecuación (5.52) se parte de la hipótesis de que el patrón pertenece a la clase $C^k$, y el patrón en cuestión presente justamente el valor numérico $\boldsymbol{x}$.

3. Para estimar las funciones de probabilidad para el clasificador a posteriori dado por la ecuación (5.51), se requiere de procesos complejos de entrenamiento y, muchas veces, muy tardados. Para más detalles, referirse por ejemplo a [mara93].

El diseño de clasificadores a priori dados por la ecuación (5.52) consiste en la estimación estadística de las funciones de densidad de probabilidad de cada una de las $p$ clases: $p\left(\boldsymbol{x}|C^1\right), p\left(\boldsymbol{x}|C^2\right), \cdots, p\left(\boldsymbol{x}|C^p\right)$, a partir de un conjunto de muestras o realizaciones para cada clase.

En la siguiente sección se estudiará el clasificador bayesiano para el cual se asume que las distribuciones de probabilidad de sus clases siguen un comportamiento gaussiano.

**Clasificador bayesiano con distribución tipo gaussiana**. En la mayoría de los casos prácticos, para el caso de vectores descriptores con $n$ rasgos, las distribuciones de probabilidad de cada una de las $p$ clases de patrones presentan un comportamiento normal o gaussiano, expresado como sigue:

$$p\left(\boldsymbol{x}|C^k\right) = \frac{1}{(2\pi)^{\frac{n}{2}}|\sum_k|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\Sigma^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\right) \quad k=1,2,\cdots,p \tag{5.53}$$

en este caso $\Sigma_k$ es la matriz de correlación $C^k$.

Tomando en cuenta lo anterior, las funciones de discriminación de tipo bayesiano toman la forma:

$$p\left(C^k\right)\cdot p\left(\boldsymbol{x}|C^k\right) = \frac{p\left(C^k\right)}{(2\pi)^{\frac{n}{2}}|\sum_k|^{\frac{1}{2}}}\exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\Sigma^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\right) \quad k=1,2,\cdots,p \tag{5.54}$$

Al tomar logaritmos neperianos sobre esta expresión se obtiene:

$$\ln\left(p\left(C^k\right)\cdot p\left(\boldsymbol{x}|C^k\right)\right) = -\tfrac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\Sigma^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T + \ln\left(p\left(C^k\right)\right) - \tfrac{n}{2}\ln(2\pi) - \tfrac{n}{2}\ln\left(|\sum_k|\right) \tag{5.55}$$

Sin pérdida de generalidad, y para simplificar la discusión, supondremos que las $p$ clases son equiprobables a priori. La expresión de cada función de discriminación $fd_k$ después de eliminar los términos redundantes, queda como:

$$fd_k = \frac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\Sigma^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T - \frac{n}{2}\ln\left(|\Sigma_k|\right), \quad k=1,2,\cdots,p \tag{5.56}$$

Enseguida se discuten los casos que se pueden presentar [37]. En particular nos centraremos en aquellos que tienen una relación estrecha con los clasificadores basados en el cálculo de distancias. Los casos que se pueden presentar son tres, como sigue:

**Caso 1:** Las matrices de covarianza de las $p$ clases son todas diferentes, es decir: $\Sigma_1 \neq \Sigma_2 \neq \cdots \neq \Sigma_p$, y en general, sin elementos nulos. Las funciones discriminantes son no lineales de tipo cuadrático (i.e., todas las posibles combinaciones cuadráticas de los elementos del vector de rasgos se hacen manifiestas).

En este caso, las fronteras de decisión o clasificadores vienen representados como círculos, elipses, parábolas o hipérbolas. En la figura 5.30 se muestra un ejemplo de una función de decisión para dos clases $C^1$ y $C^2$ en el caso dos-dimensional de este tipo.

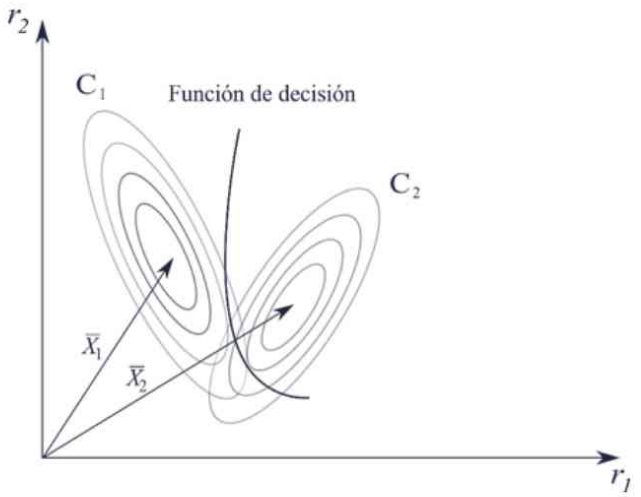



**Figura 5.30**: Ejemplos de función de decisión cuadrática.

**Caso 2:** Las matrices de covarianza de las $p$ clases son todas iguales, es decir: $\Sigma_1 = \Sigma_2 = \cdots = \Sigma_p$ y en general, sin elementos nulos. Este caso es el que más se presenta en la práctica. Se parte del supuesto de que el vector de rasgos presenta un comportamiento estadístico similar para todas las clases de objetos. En este caso se pueden eliminar todos los términos no lineales de la ecuación (5.56) quedando funciones discriminantes para cada clase del tipo:

$$fd_k(\boldsymbol{x}) = \boldsymbol{x}^T \Sigma_k^{-1} \bar{\boldsymbol{x}}_k - \frac{1}{2} \bar{\boldsymbol{x}}_k^T \Sigma_k^{-1} \bar{\boldsymbol{x}}_k, \quad k = 1, 2, \cdots, p \tag{5.57}$$

Para este caso en particular existen dos subcasos que se pueden presentar:

**Caso 2.1:** A parte de ser iguales, las matrices de covarianza son de la forma:

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_p = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 I \tag{5.58}$$

donde $I$ es la matriz identidad, y las matrices de covarianza son diagonales con valor diagonal $[\,\sigma^2 \cdots \sigma^2\,]$, lo cual implica que los rasgos son estadísticamente independientes, con la misma varianza; esto es que son homoestadísticos. Esta suposición nos lleva al hecho de que:

$$p\left(\boldsymbol{x}|C^k\right) \approx \Pi_{i=1}^n p\left(x_i|C^k\right) \tag{5.59}$$

Además, las probabilidades $p\left(x_1|C^k\right), p\left(x_2|C^k\right), \cdots, p\left(x_n|C^k\right)$ son muy fáciles de estimar a partir del conjunto de entrenamiento.

Debido a que $\Sigma_k^{-1} = \text{diagonal}\,[\,\frac{1}{\sigma^2}, \cdots, \frac{1}{\sigma^2}\,]$, al sustituir en (5.57) se tiene que las funciones de decisión tiene ahora la forma:

$$fd_k(\boldsymbol{x}) = \frac{1}{\sigma^2} \boldsymbol{x}^T \bar{\boldsymbol{x}}_k - \frac{1}{2\sigma^2} \bar{\boldsymbol{x}}_k^T \bar{\boldsymbol{x}}_k, \quad k = 1, 2, \cdots, p \tag{5.60}$$

Hay que recordar que $x_i$ se refiere al valor del atributo $r_i$ para la muestra $\boldsymbol{x}$. Siguiendo esta hipótesis, después de eliminar el factor de escalamiento $\sigma^2$, las funciones de decisión quedan finalmente como:

$$fd_k(\boldsymbol{x}) = \boldsymbol{x}^T \bar{\boldsymbol{x}}_k - \frac{1}{2} \bar{\boldsymbol{x}}_k^T \bar{\boldsymbol{x}}_k, \quad k = 1, 2, \cdots, p \tag{5.61}$$

En la figura 5.31 se muestra un ejemplo del tipo de funciones de decisión que se pueden generar a partir de la ecuación (5.60), para el caso de dos clases $C^1$ y $C^2$. En este caso las funciones de decisión son de tipo lineal y, por tanto, los clasificadores asociados son del mismo tipo.

No es difícil demostrar que las funciones de discriminación dadas por la expresión (5.61) coinciden con el clasificador euclidiano discutido en la sección 5.5.2, donde cada uno de los representantes $z^k,\ k = 1, 2, \cdots, p$ ha sido sustituido por el vector media $\bar{\boldsymbol{x}}_k$ de cada clase.

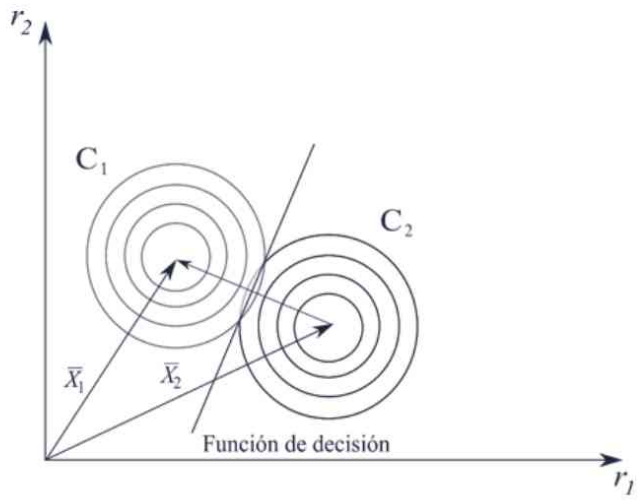



**Figura 5.31**: Ejemplo de función de decisión lineal.

**Caso 2.2:** A parte de ser iguales, las matrices de covarianza tienen la forma:

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_p = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{bmatrix}. \tag{5.62}$$

En otras palabras, los rasgos son estadísticamente independientes pero además, presentan varianzas distintas. Las funciones de discriminación toman ahora la forma:

$$fd_k(\boldsymbol{x}) \quad = \quad \boldsymbol{x}^T \Sigma_k^{-1} \bar{\boldsymbol{x}}_k - \frac{1}{2} \bar{\boldsymbol{x}}_k^T \Sigma_k^{-1} \bar{\boldsymbol{x}}_k, \quad k = 1, 2, \cdots, p \tag{5.63}$$

Al igual que en el caso anterior, no es difícil darse cuenta de que las funciones de decisión calculadas mediante la ecuación (5.63) coinciden con las obtenidas a través de la ecuación (5.46). Es decir, cuando las distribuciones de las clases siguen la ecuación (5.53), las funciones de decisión son equivalentes al cálculo de las $p$ distancias de Mahalanobis del patrón de entrada $\boldsymbol{x}$, respecto a las $p$ clases.

**Estimación de los parámetros de la matriz de covarianza**. Como ya se mencionó, el principal problema en el diseño de clasificadores a priori es la estimación de las funciones de densidad de probabilidad $p\left(\boldsymbol{x}|C^k\right),\ k = 1, 2, \cdots, p$ a partir de un conjunto de muestras o realizaciones, uno por clase. Para una clase dada $C^k$, y suponiendo que se dispone de $m$ muestras o realizaciones, primero se estima la esperanza matemática de la clase mediante el cálculo del vector promedio como:

$$\bar{\boldsymbol{x}}_k = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_n \end{bmatrix} = \begin{bmatrix} \frac{1}{m}\sum_{j=1}^{m} x_{1,j} \\ \frac{1}{m}\sum_{j=1}^{m} x_{2,j} \\ \vdots \\ \frac{1}{m}\sum_{j=1}^{m} x_{n,j} \end{bmatrix}. \tag{5.64}$$

Por otro lado, la matriz de covarianza de la clase $C^k$ puede estimarse como:

$$\Sigma_k = \frac{1}{m}\sum_{j=1}^{m} \boldsymbol{x} \cdot \boldsymbol{x}_j^T - \bar{\boldsymbol{x}}_k \cdot \bar{\boldsymbol{x}}_k^T. \tag{5.65}$$

Enseguida se da un ejemplo para ilustrar la operación y utilidad del clasificador bayesiano.

**•• Ejemplo 5.23**

Supóngase el siguiente universo de patrones y sus clases

$$\left\{ \begin{bmatrix} 6 \\ 7 \end{bmatrix}, \begin{bmatrix} 6 \\ 6 \end{bmatrix}, \begin{bmatrix} 7 \\ 6 \end{bmatrix}, \begin{bmatrix} 6 \\ 5 \end{bmatrix}, \begin{bmatrix} 7 \\ 5 \end{bmatrix} \right\} \in C_1$$

$$\left\{ \begin{bmatrix} 7 \\ 8 \end{bmatrix}, \begin{bmatrix} 8 \\ 8 \end{bmatrix}, \begin{bmatrix} 7 \\ 7 \end{bmatrix}, \begin{bmatrix} 8 \\ 7 \end{bmatrix}, \begin{bmatrix} 8 \\ 6 \end{bmatrix} \right\} \in C_2$$

Determinar la clase de pertenencia de los patrones: $\boldsymbol{x}^1 = \begin{bmatrix} 6.5 \\ 7 \end{bmatrix}$ y $\boldsymbol{x}^2 = \begin{bmatrix} 6.5 \\ 8 \end{bmatrix}$.

**Solución**

En la figura 5.32 se muestran el universo de patrones y sus respectivas clases.

**Entrenamiento del clasificador**. De acuerdo con lo discutido para el entrenamiento del clasificador a priori de Bayes, es necesario calcular los vectores de medias así como las matrices de covarianza de cada clase.

Procedamos de la siguiente manera:

$$\begin{aligned} \boldsymbol{x}^1 &= \frac{1}{5}\left[\begin{bmatrix}6\\7\end{bmatrix}+\begin{bmatrix}6\\6\end{bmatrix}+\begin{bmatrix}7\\6\end{bmatrix}+\begin{bmatrix}6\\5\end{bmatrix}+\begin{bmatrix}7\\5\end{bmatrix}\right]=\begin{bmatrix}6.4\\5.8\end{bmatrix} \\ \boldsymbol{x}^2 &= \frac{1}{5}\left[\begin{bmatrix}7\\8\end{bmatrix}+\begin{bmatrix}8\\8\end{bmatrix}+\begin{bmatrix}7\\7\end{bmatrix}+\begin{bmatrix}8\\7\end{bmatrix}+\begin{bmatrix}8\\6\end{bmatrix}\right]=\begin{bmatrix}7.6\\7.2\end{bmatrix}. \end{aligned}$$

De igual manera:

$$\begin{aligned} \Sigma_1 &= \tfrac{1}{5}\left[\begin{bmatrix}6\\7\end{bmatrix}[6\ \ 7]+\begin{bmatrix}6\\6\end{bmatrix}[6\ \ 6]+\begin{pmatrix}7\\6\end{pmatrix}[7\ \ 6]+\begin{bmatrix}6\\5\end{bmatrix}[6\ \ 5]+\begin{bmatrix}7\\5\end{bmatrix}[7\ \ 5]\right]-\begin{bmatrix}6.4\\5.8\end{bmatrix}[6.4\ \ 5.8] \\ &= \begin{bmatrix}41.2 & 37\\ 37 & 34.2\end{bmatrix}-\begin{bmatrix}40.96 & 37.12\\ 37.12 & 33.64\end{bmatrix}=\begin{bmatrix}0.24 & 0.12\\ -0.12 & 0.56\end{bmatrix} \\ \Sigma_2 &= \tfrac{1}{5}\left[\begin{bmatrix}7\\8\end{bmatrix}[7\ \ 8]+\begin{bmatrix}8\\8\end{bmatrix}[8\ \ 8]+\begin{bmatrix}7\\7\end{bmatrix}[7\ \ 7]+\begin{bmatrix}8\\7\end{bmatrix}[8\ \ 7]+\begin{bmatrix}8\\6\end{bmatrix}[8\ \ 6]\right]-\begin{bmatrix}7.6\\7.2\end{bmatrix}[7.6\ \ 7.2] \\ &= \begin{bmatrix}58 & 54.6\\ 54.6 & 52.4\end{bmatrix}-\begin{bmatrix}57.76 & 54.72\\ 54.72 & 51.84\end{bmatrix}=\begin{bmatrix}0.24 & 0.12\\ -0.12 & 0.56\end{bmatrix} \end{aligned}$$

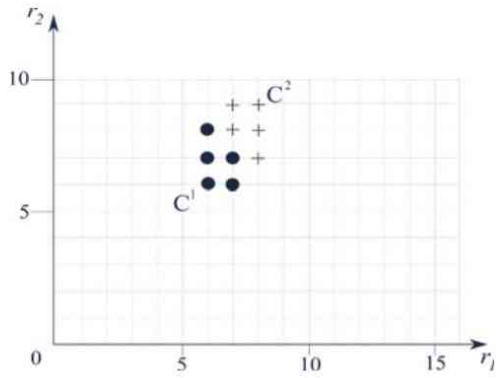



**Figura 5.32**: Patrones y sus clases para el ejemplo 5.23.

Note cómo las matrices de covarianza de las dos clases son iguales y las varianzas por rasgo son diferentes. En consecuencia, las funciones de discriminación pueden ser calculadas a través de la ecuación (5.57). Al hacer las sustituciones y cálculos respectivos se tiene que:

$$fd_1(\boldsymbol{x}) = [x_1\ \ x_2]\begin{bmatrix}0.24 & 0.12\\ -0.12 & 0.56\end{bmatrix}^{-1}\begin{bmatrix}6.4\\5.8\end{bmatrix}-\frac{1}{2}[6.4\ \ 5.8]\begin{bmatrix}0.24 & 0.12\\ -0.12 & 0.56\end{bmatrix}^{-1}\begin{bmatrix}6.4\\5.8\end{bmatrix}$$

$$
\begin{aligned}
&= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 4.66 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 6.4 \\ 5.8 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 6.4 & 5.8 \end{bmatrix} \begin{bmatrix} 4.66 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 6.4 \\ 5.8 \end{bmatrix} \\
&= 35.62x_1 + 18x_2 - 166.184.
\end{aligned}
$$

$$
\begin{aligned}
fd_2(\boldsymbol{x}) &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 0.24 & 0.12 \\ -0.12 & 0.56 \end{bmatrix}^{-1} \begin{bmatrix} 7.6 \\ 7.2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 7.6 & 57.2 \end{bmatrix} \begin{bmatrix} 0.24 & 0.12 \\ -0.12 & 0.56 \end{bmatrix}^{-1} \begin{bmatrix} 7.6 \\ 7.2 \end{bmatrix} \\
&= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 4.66 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 7.6 \\ 7.2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 7.6 & 7.2 \end{bmatrix} \begin{bmatrix} 4.66 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 7.6 \\ 7.2 \end{bmatrix} \\
&= 42.616x_1 + 22x_2 - 241.140.
\end{aligned}
$$

Si se supone que la probabilidad de aparición de un elemento de cualquiera de las dos clases $C^1$ y $C^2$ es la misma, esto es si $p(C^1) = p(C^2) = 0.5$, entonces un vector descriptor $\boldsymbol{x}$ será clasificado como:

$$
\begin{aligned}
\boldsymbol{x} \longrightarrow C^1 \quad &\text{si y solo si} \quad fd_1(\boldsymbol{x}) > fd_2(\boldsymbol{x}), \text{ esto es si } p\left(\boldsymbol{x}|C^1\right) > p\left(\boldsymbol{x}|C^2\right) \\
\boldsymbol{x} \longrightarrow C^2 \quad &\text{si y solo si} \quad fd_1(\boldsymbol{x}) < fd_2(\boldsymbol{x}), \text{ esto es si } p\left(\boldsymbol{x}|C^1\right) < p\left(\boldsymbol{x}|C^2\right).
\end{aligned}
$$

**Prueba del clasificador**: Para el vector de entrada $\boldsymbol{x}^1 = \begin{bmatrix} 6.5 \\ 7 \end{bmatrix}$:

$$
\begin{aligned}
fd_1(\boldsymbol{x}) &= 35.62\ (6.5) + 18\ (7) - 166.184 = 191.346 \\
fd_2(\boldsymbol{x}) &= 42.616\ (6.5) + 22\ (7) - 241.140 = 189.864.
\end{aligned}
$$

Debido a que $fd_1(\boldsymbol{x}) > fd_2(\boldsymbol{x})$, entonces el patrón $\boldsymbol{x}^1 = \begin{bmatrix} 6.5 \\ 7 \end{bmatrix}$ es clasificado en la clase $C^1$.

Para el vector de entrada $\boldsymbol{x}^1 = \begin{bmatrix} 6.5 \\ 8 \end{bmatrix}$:

$$
\begin{aligned}
fd_1(\boldsymbol{x}) &= 35.62\ (6.5) + 18\ (8) - 166.184 = 209.346 \\
fd_2(\boldsymbol{x}) &= 42.616\ (6.5) + 22\ (8) - 241.140 = 211.864.
\end{aligned}
$$

Debido a que $fd_1(\boldsymbol{x}) < fd_2(\boldsymbol{x})$, entonces el patrón $\boldsymbol{x}^1 = \begin{bmatrix} 6.5 \\ 8 \end{bmatrix}$ es clasificado en la clase $C^2$.

## 5.6 Resumen

En este capítulo vimos cómo los diferentes objetos en una escena pueden ser detectados e identificados a partir de una imagen de dicha escena. Con esto un robot será capaz de realizar una tarea dentro de su medio ambiente, por ejemplo, desplazarse de un punto a otro, ir y tomar un objeto dado. En lo particular:

1. Se estudió el problema de la segmentación y su complejidad. También se estudiaron dos técnicas de segmentado de imágenes a través de selección automática de un umbral, así como una técnica basada en el crecimiento de regiones.

2. Se revisaron dos técnicas para el postratamiento de imágenes segmentadas con el fin de eliminar la presencia de pequeños componentes ruidosos como píxeles y hoyos parásitos.

3. Se profundizó en el problema de etiquetado de componentes conectados como etapa posterior al umbralado de imágenes con el objetivo de asignar etiquetas distintas a cada una de las regiones previamente segmentadas. Se analizó en detalle un método secuencial muy usado en la literatura.

4. Se presentaron varios rasgos descriptivos globales de tipo geométrico y topológico. Entre los rasgos geométricos estudiamos al factor de compacidad, que mide qué tanto un objeto se aproxima a una forma circular. También se analizaron los famosos rasgos descriptivos de Hu, invariantes a transformaciones de imagen como traslaciones, rotaciones y cambios de tamaño. Se vio cómo estos rasgos pueden ser usados para determinar la posición de un objeto, su centro y el ángulo de su eje mayor; ambos útiles para que un robot sea capaz de ubicar un objeto reconocido y lo tome si así se desea. Posteriormente se estudió el número de Euler como descriptor de la estructura de un objeto; se analizaron dos métodos para su cálculo.

5. Finalmente, se profundizó en la operación de varios clasificadores para detectar la presencia de un objeto y para determinar su clase de pertenencia. En particular, se estudiaron los clasificadores con base al cálculo de distancia euclidiana y de Mahalanobis, así como varios clasificadores que utilizan el teorema de Bayes.

## 5.7 Problemas propuestos

En esta sección, el lector encontrará una serie de problemas a manera de ejercicio que le permitirán, por un lado, reafirmar el conjunto de conceptos vistos en este capítulo. Por el otro lado, podrá aprender nuevos conceptos y poner en práctica sus habilidades de programación en el manejo de imágenes.

5.7.1 Diseñar y poner en operación en **MATLAB** un programa que permita al usuario, dada una imagen en niveles de gris $f(x, y)$ como las que se muestran en la figura 5.33, seleccionar en forma manual a partir del histograma de la imagen el umbral $u$ que dará como resultado la imagen binaria correspondiente $b(x, y)$.


**Figura 5.33**: Imágenes de prueba de los problemas 5.7.1 al 5.7.6.

5.7.2 Programar en **MATLAB** el método de Otsu descrito en la subsección 5.2.5 y umbralar las imágenes de la figura 5.33.

Conteste y argumente la siguiente pregunta:

¿Qué umbrales generó el método de Otsu?

5.7.3 Desarrollar un procedimiento del método de Kittler-Illingworth en **MATLAB** descrito en la subsección 5.2.6 y umbralar las imágenes de la figura 5.33. En caso necesario, utilice el filtro dado en la ecuación (5.9).

¿Qué umbrales generó el método de Kittler-Illingworth?

¿Qué umbrales generó el método de Otsu?

5.7.4 Programar en **MATLAB** el método de etiquetado de componentes conectados descrito en la subsección 5.3.3. Para esto utilice una de las imágenes binarias generadas a través del método de Otsu programado en el problema 5.7.2.

5.7.5 Desarrollar el código fuente necesario en **MATLAB** para implementar los métodos de postfiltrado de imágenes descritos en la sección 5.3 para eliminar los huecos y componentes parásitos, generados por un pobre umbralado. Para esto umbrale manualmente las imágenes de la figura 5.33 de tal forma que visualice componentes y huecos aislados en las imágenes binarias correspondientes.

5.7.6 Programar un procedimiento en **MATLAB** para calcular los siete invariantes de Hu $\phi_1, \phi_2, \cdots, \phi_n$, según lo descrito en la subsección 5.4.8, para cada uno de los objetos mostrados en las cuatro imágenes de figura 5.33.

5.7.7 Demostrar que cualquier momento central $\mu_{pq}$ es invariante a traslaciones en el plano dadas por la ecuación (5.20), ver página 190.

5.7.8 Demostrar la ecuación (5.28), ver página 193.

5.7.9 Realizar en **MATLAB** un procedimiento para calcular los siete invariantes de Hu $\phi_1, \phi_2, \cdots, \phi_n$ según lo descrito en la subsección 5.4.8, para cada uno de los siguientes 10 objetos mostrados en la figura 5.34. Primero umbrale las imágenes a través del procedimiento diseñado en el problema 5.7.2.

5.7.10 Programar en **MATLAB** un procedimiento para calcular el número de Euler, de acuerdo con lo descrito en la subsección 5.4.1, para cada uno de los siguientes 10 objetos mostrados en la figura 5.34. Primero umbrale las imágenes por medio del procedimiento diseñado en el problema 5.7.2.

5.7.11 Implementar en **MATLAB** un procedimiento que determine el centroide y orientación (subsección 5.4.9) del eje mayor de los objetos mostrados en las cuatro imágenes de la figura 5.35. Primero umbrale las imágenes a través del procedimiento diseñado en el problema 5.7.2.

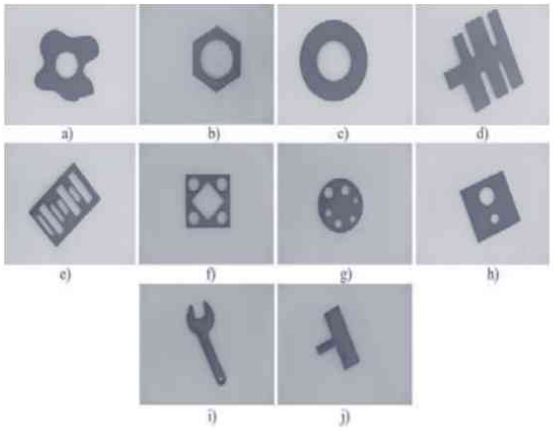


**Figura 5.34**: Imágenes de prueba de los problemas 5.7.9 al 5.7.10.

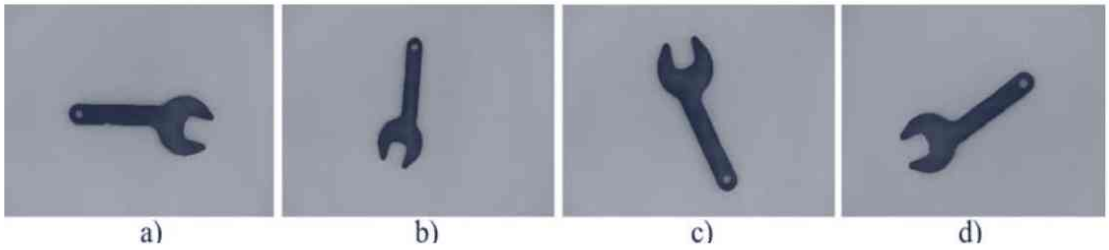



**Figura 5.35**: Imágenes de prueba del problema 5.7.11.

5.7.12 Suponga el siguiente conjunto de patrones mostrado en la figura 5.36. Aplicar el procedimiento estudiado en la subsección 5.5.2 para entrenar el clasificador de distancia mínima. Una vez entrenado, determinar la clase de pertenencia de los siguientes patrones: $\boldsymbol{x} = [8 \quad 9]^T$ y $Y = [12 \quad 4]^T$.

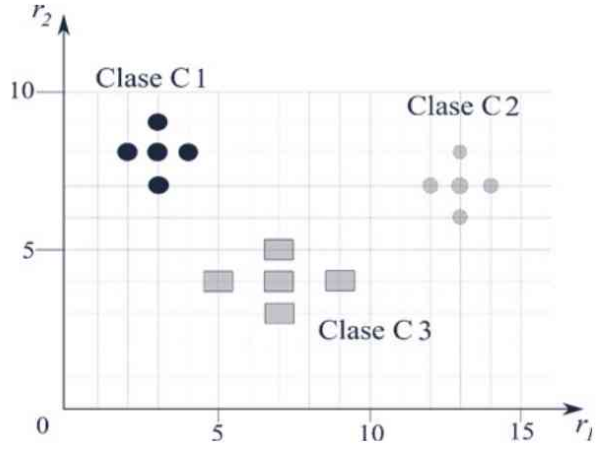



**Figura 5.36**: Distribución de patrones para el problema 5.7.12.

5.7.13 Sea el siguiente conjunto de patrones mostrado en la figura 5.37. Aplicar el procedimiento estudiado en la subsección 5.5.4 para entrenar el clasificador de distancia de Mahalanobis. Una vez entrenado, determinar la clase de pertenencia de los siguientes patrones: $\boldsymbol{x} = [10 \quad 9]^T$ y $Y = [10 \quad 10]^T$.

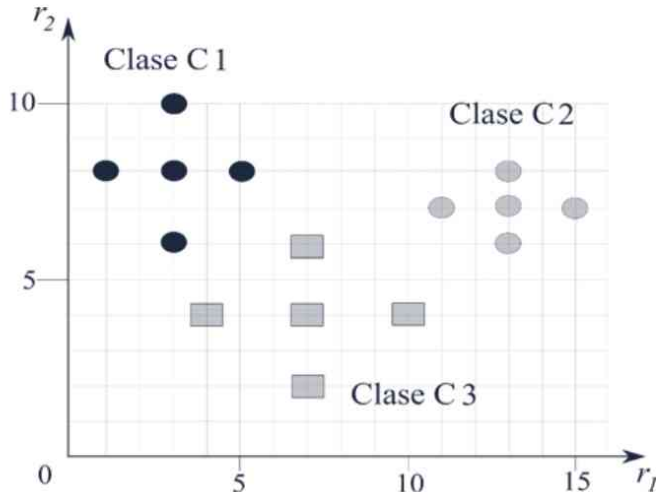



**Figura 5.37**: Distribución de patrones para el problema 5.7.13.

5.7.14 Suponga el siguiente conjunto de patrones mostrado en la figura 5.38. Aplicar el procedimiento estudiado en la subsección 5.5.5 para entrenar un clasificador estadístico. Calcular las funciones de decisión correspondientes y entonces determinar la clase de pertenencia de los siguientes patrones: $\boldsymbol{x} = [9 \quad 7]^T$ y $Y = (98)^T$.

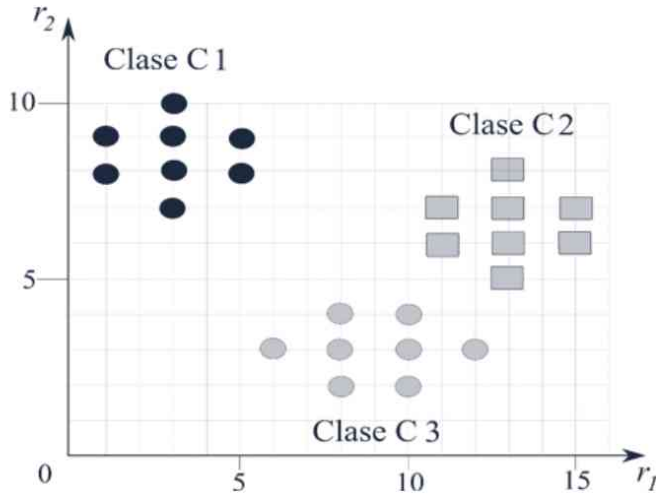



**Figura 5.38**: Distribución de patrones para el problema 5.7.14.

5.7.15 Suponga el siguiente conjunto de patrones mostrado en la figura 5.39. Aplicar el procedimiento estudiado en la sección 5.5.5 para entrenar un clasificador estadístico. Calcular las funciones de decisión correspondientes y entonces determinar la clase de pertenencia de los siguientes patrones: $\boldsymbol{x} = [8 \quad 8]^T$ y $Y = [9 \quad 8]^T$.

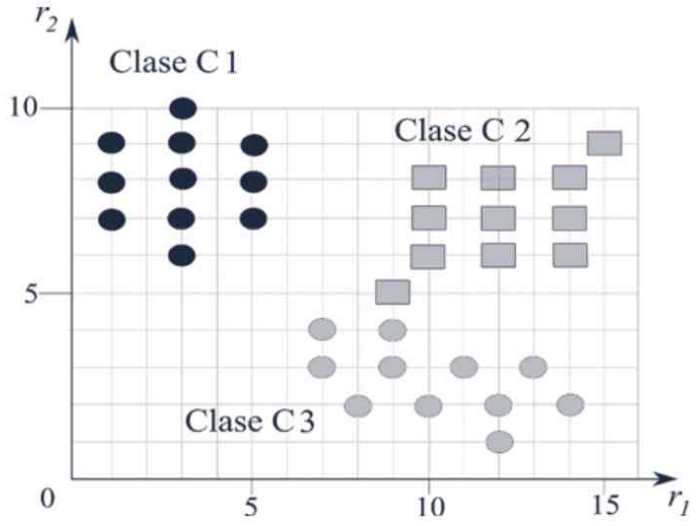



**Figura 5.39**: Distribución de patrones para el problema 5.7.15.

5.7.16 Considere cinco objetos planos similares a los usados en el ejercicio 5.7.9, con forma distinta y eventualmente diferente número de huecos.

Hacer lo siguiente:

Entrenamiento de clasificadores:

1) Tome 10 imágenes de cada uno de estos cinco objetos en diferentes posiciones, orientaciones y tamaños.

2) Umbrale las 50 imágenes a través del método desarrollado en el problema 5.7.2.

3) Para cada muestra de cada uno de los cinco objetos, calcule su primera invariante de Hu $\phi_1$ y su número de Euler, con todos estos valores integre una tabla como se muestra a continuación.

4) Con los 50 valores, entrene un clasificador de distancia euclidiana, de Mahalanobis y bayesiano.

Prueba de clasificadores:

5) Tome otras 10 imágenes de cada uno de los cinco objetos en diferentes posiciones, orientaciones y tamaños.

6) Umbrale las 50 imágenes a través del método desarrollado en el problema 5.7.14.

7) Para las muestras de cada uno de los cinco objetos, calcule su primera invariante de Hu $\phi_1$ y su número de Euler, con todos estos valores integre una tabla como ya se hizo en el paso 3.

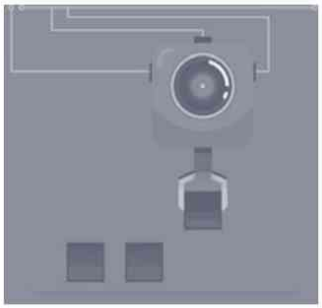
8) Con los 50 valores, pruebe el desempeño de los clasificadores de distancia euclidiana, de Mahalanobis y bayesiano entrenados en el paso 4.

9) ¿Qué clasificador ofrece el mejor desempeño?

Comente al respecto.

Capítulo 6

# El perceptrón y su entrenamiento



# Descripción del capítulo

Este capítulo estudia uno de los paradigmas más utilizados actualmente en la solución de problemas de clasificación y reconocimiento de patrones, la predicción de series de tiempo, el modelado de sistemas y la optimización. Se describen las redes neuronales artificiales, que intentan simular la operación de neuronas biológicas. Después se presentan los pormenores de arquitecturas básicas de redes neuronales más utilizadas en la literatura actual. Se presenta la primera neurona utilizada por la comunidad científica: la unidad de umbralado lógico (UUL) y la forma en que puede ser entrenada a través de la regla del perceptrón para resolver problemas de diversa índole. Enseguida, se dan los detalles de la operación y funcionalidad del conocido perceptrón como extensión de la UUL. Se habla de la neurona lineal ADALINE y cómo esta puede ser entrenada mediante la regla Delta, basada en el principio del descenso del gradiente. Más adelante, se habla de la neurona sigmoidal que, a diferencia del perceptrón clásico puede emitir valores en rango continuo acotado. Se ofrece también una serie de ejemplos ilustrativos. Al final, se presenta un conjunto de ejercicios para que el lector afiance el conocimiento adquirido y ponga en práctica sus capacidades y habilidades.

Los siguientes temas son abordados:


Arquitecturas básicas de redes neuronales.


Modelos de neuronas artificiales.


El perceptrón.


Neurona ADALINE y la regla Delta.


El perceptrón sigmoidal.

# 6.1 Introducción

En este capítulo, primeramente, se presenta un conjunto de conceptos fundamentales relacionados con la neurona biológica, sus partes y su operación básica. Después, se dan detalles sobre los modelos básicos que se pueden encontrar en la literatura. Enseguida, se describen los aspectos sobre una neurona artificial con una entrada y con varias entradas. Más adelante, se explican diferentes tipos de funciones de transferencia que a su salida pueden tener dichos modelos de neuronas.

Después se ofrece un conjunto de detalles sobre arquitecturas de redes neuronales, en particular una red con una capa de neuronas y otra con varias capas de neuronas. Enseguida, se presentan todos los detalles con los primeros modelos de neuronas artificiales para construir redes neuronales artificiales. Primeramente, se habla de la conocida Unidad de Umbralado Lógica (UUL). Se exponen algunas de sus propiedades, como la separabilidad lineal y cómo esta máquina puede ser entrenada mediante la regla del perceptrón. Se habla de la robustez de estas máquinas ante ruido en las entradas y hardware. En segundo lugar, se aborda el perceptrón del Rosenblatt (PR) y sus realizaciones populares estándar. Se describen las diferencias de estas últimas máquinas con respecto al original PR.

En tercer lugar, se aborda la neurona ADALINE, la cual dotada de una función de activación lineal permite resolver ciertos problemas que el perceptrón, por sí solo no puede resolver. Se dan los detalles sobre la regla Delta, con base en el principio del descenso del gradiente para el entrenamiento de ADALINES solas o ubicadas en una capa. En cuarto lugar, se muestra cómo al dotar al perceptrón de una función de transferencia diferenciable en todos sus puntos, acotada y monótona creciente, puede ser utilizada para resolver otros problemas que el perceptrón original no podría resolver. Se dan los detalles de la derivación de la regla Delta adaptada a su entrenamiento.

Finalmente se da un resumen de este capítulo, así como un conjunto de ejemplos ilustrativos para que el lector asimile de manera clara los conceptos presentados. De igual forma, al final del capítulo se ofrece un listado de ejercicios para ejercitar habilidades y al mismo tiempo, reforzar el conocimiento adquirido durante la lectura de este capítulo.

## 6.2 Conceptos básicos

En esta sección se exponen los principios de operación del paradigma neuronal, desarrollando los principios matemáticos de su operación. En cada caso, se presentan varios mecanismos para su entrenamiento en la solución de problemas de complejidad diversa. Cuando sea necesario, se dan ejemplos para asimilar y reforzar los conceptos aprendidos. Primeramente se dan unas palabras relativas al cómputo llevado a cabo por las creaturas biológicas: el cómputo neuronal biológico.

### 6.2.1 Cómputo neuronal

El cómputo neuronal es uno de los paradigmas más poderosos existentes en la naturaleza para la solución de problemas complejos. Dejando de lado a las plantas y otros seres vivos como las bacterias, el resto de los seres vivos, desde los insectos, los peces, las aves, los anfibios, los reptiles, los mamíferos y, finalmente, el hombre, todos poseen una máquina muy poderosa llamada cerebro que les permite desenvolverse de manera muy efectiva dentro su medio ambiente. Esta máquina biológica, regida por las leyes de la naturaleza, como ya vimos en el capítulo 2, es un órgano compuesto por un gran número de unidades de cómputo "sencillas" altamente interconectadas, llamadas neuronas. En todos los seres vivos que las poseen, desde el más sencillo de ellos, el gusano C-Elegans (con unas 300 neuronas) como se describe en [39], hasta el ser humano (con unos 86,000 millones de ellas). La neurona biológica es una célula especializada en el procesamiento de señales de entrada provenientes de los diferentes sentidos; en el ser humano: la vista, el oído, el olfato, el tacto y el gusto.

La neurona natural (ver figura 6.1) como la unidad de procesamiento básico de información, lleva a cabo, de manera muy general, cuatro niveles de tratamiento sobre la información recibida a su entrada, a saber:

1) Procesamiento sináptico.

2) Procesamiento dendrítico.

3) Procesamiento somático.

4) Procesamiento axonal.

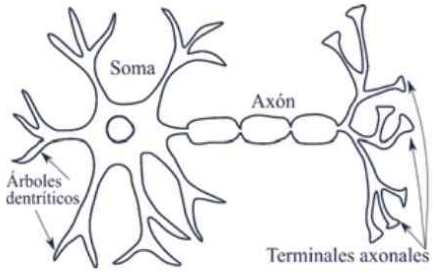



**Figura 6.1**: Neurona y sus diferentes partes: árboles dendríticos, cuerpo o soma, axón y terminales o botones axonales.

Grosso modo, a **nivel sináptico** como se muestra en la figura 6.2, en una sinapsis de tipo químico un potencial de acción provoca que la neurona pre-sináptica libere biomoléculas llamadas neurotransmisores. Estas moléculas (portadoras de información) se unen a receptores en la neurona post-sináptica y modifican la probabilidad de que esta neurona dispare un potencial de acción.

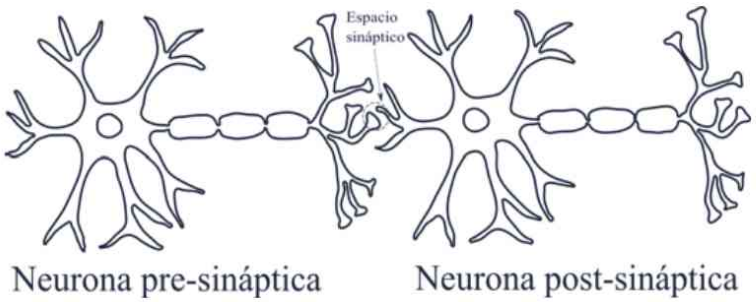



**Figura 6.2**: La sinapsis es el espacio que permite "conectar" neuronas: pre-sinápticas con neuronas post-sinápticas.

A **nivel dendrítico** cada rama dendrítica sirve como receptora de impulsos nerviosos provenientes de axones pertenecientes a otras neuronas conectadas en la neurona de dichas dendritas. La función principal de las dendritas es recibir los impulsos de otras neuronas, procesarlos y enviarlos hasta el soma de la neurona.

A **nivel somático**, la información captada y pre-procesada por las dendritas de la neurona abren y cierran canales iónicos que de acuerdo con su carga contribuyen a la generación de impulsos nerviosos en forma de potenciales de acción. A **nivel axón** estos potenciales

de acción son impulsados desde el cabezal de dicho axón, post-procesando la información "montada" en el potencial de acción a lo largo del axón hasta los botones terminales, y de ahí, de nuevo a los espacios sinápticos que conectan a la neurona procesadora actual a las siguientes neuronas a lo largo de la corteza cerebral.

De manera muy sencilla, esta es la forma en que las neuronas en un cerebro reciben información de los sentidos a los cuales están conectadas o de otras neuronas conectadas a ellas, a través de la sinapsis procesan dicha información y producen los impulsos nerviosos que eventualmente han de disparar otras neuronas, mover músculos, detectar movimiento o la presencia de objetos, reconocer objetos o bien en etapas de abstracción más complejas, permitir que la criatura portadora de un cerebro compuesto de cientos, miles, cientos de miles, millones o cientos de millones de ellas pueda caminar, comer, razonar, incluso pensar, hablar, crear; más aún, experimentar uno de los fenómenos más misteriosos de la vida: la conciencia.

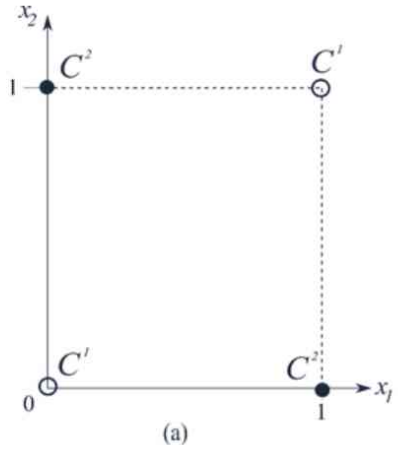
Cualquier modelo de neurona artificial debería incorporar los cuatro niveles de procesamiento mencionados para operar conforme a los principios biológicos en que lo hace una neurona biológica. Esto no impide que se diseñen modelos aproximados de neuronas que al ser combinados formen redes neuronales artificiales, que a su vez permitan resolver eficientemente problemas de detección, clasificación y reconocimiento de patrones o de predicción de series de tiempo o de control de procesos, según convenga.

En las siguientes secciones se estudiarán modelos matemáticos de neuronas artificiales que han demostrado su eficiencia y eficacia en la solución de problemas de clasificación de patrones, la predicción de series de tiempo o de control. En todos los casos, se presentarán sus componentes, las matemáticas detrás de su funcionamiento, así como un conjunto de métodos útiles para su entrenamiento. También se darán ejemplos ilustrativos para profundizar en su operación y su aplicabilidad.

### 6.2.2 Modelos básicos de una neurona

Los modelos de clasificación estudiados en los capítulos 2, 3 y 5 son útiles únicamente para el caso de clases de patrones linealmente separables. En otras palabras, cuando entre cada par de clases es posible encontrar una hipersuperficie que permite separar los elementos de una clase de la otra. En la práctica, la mayoría de los problemas no presentan separación lineal. Consideremos, por ejemplo, el conjunto de patrones mostrado en la figura 6.3. Se

trata del conocido problema del cálculo de la función O-exclusivo (XOR) con dos variables, el cual puede ser visto como un problema de clasificación en dos clases, como se muestra en la figura 6.3b. Cuando $x_1 = x_2$, el patrón es clasificado en la clase $C^1$, mientras que cuando $x_1 \neq x_2$, el patrón es clasificado en la clase $C^2$. Como se puede ver en la figura 6.3a, no es posible encontrar una línea en el plano $x_1 - x_2$ que permita separar a los elementos de las dos clases.




| $x_1$ | $x_2$ | Clase de pertenencia |
|---|---|---|
| 0 | 0 | $C^1$ |
| 0 | 1 | $C^2$ |
| 1 | 0 | $C^2$ |
| 1 | 1 | $C^1$ |

(b)

**Figura 6.3**: Problema O-exclusivo (XOR) para las variables lógicas $x_1$ y $x_2$. En el plano $x_1 - x_2$, figura (a) y en términos de los rasgos $x_1, x_2$ con su tabla de verdad, figura (b).

Otra situación que puede presentarse es que las propiedades estadísticas de los clasificadores lineales no se conozcan o no puedan ser estimadas. En ambos casos, se deben de adoptar esquemas de clasificación más apropiados que puedan ser entrenados mediante un conjunto de ejemplos (esto es, con muestras de cada una de las clases). Uno de los esquemas que permite llevar a cabo esta tarea de manera natural son las Redes Neuronales Artificiales (RNA). De manera muy general, una RNA es una colección de elementos sencillos de cálculo llamados neuronas que presentan una gran interconectividad y que, en su conjunto, integran dicha RNA.

En la literatura, estos dispositivos de cálculo son también conocidos como neuro-computadores; modelos de procesamiento masivamente paralelo y distribuido; además de redes con capas autoadaptables. Como quiera llamárse a estos elementos de cálculo permiten obtener, de manera adaptable e iterada, las funciones de decisión a ser usadas para clasificar patrones linealmente o no linealmente separables. Matemáticamente, una RNA puede ser definida por un grafo dirigido con las siguientes cuatro propiedades:

1) Una variable de estado $n_i$ asociada a cada nodo $i$.

2) Un peso en los reales $p_{ij}$ asociado con la conexión entre los nodos $i$ y $j$.

3) Un desplazamiento $b_i$ asociado a cada nodo.

4) Una función de transición o de activación $f_i\,[n_j,\ p_{ij},\ b_i,\ (j \neq i)]$ para cada nodo $i$ que determina el estado del nodo como en función de los pesos de las conexiones entrantes y los estados de los nodos conectados a dicho nodo por estas conexiones.

Los primeros modelos de redes neuronales datan de principios de la década de 1940, con el trabajo de McCulloch y Pitts [38], los autores describen un modelo útil en el caso binario, conocido como UUL, del inglés Thershod Logic Unit (TLU). En el trabajo de Hebb [33], el autor propone un modelo que trata de simular el concepto de aprendizaje a través de lo que se conoce como refuerzo o asociación. A finales de 1950 y principios de 1960, Rosenblatt [45] y [46], introduce el famoso y polémico perceptrón. En estos trabajos, Rosenblatt demuestra que cuando un perceptrón es entrenado con patrones pertenecientes a clases linealmente separables, los coeficientes del perceptrón son obtenidos en un número finito de pasos. En 1969, Minsky y Papert [40] muestran las limitaciones de dicha máquina de procesamiento.

Durante años, y hasta mediados de la década de 1980, parecía que no había más que hacer respecto del famoso perceptrón que mucho prometía. Pero fue que en 1986, Rumelhart, Hinton y Williams desarrollaron un nuevo mecanismo para el entrenamiento de perceptrones acomodados en varias capas. Este nuevo mecanismo bautizado con el nombre de regla Delta generalizada, como veremos en el capítulo 7, será usado para el entrenamiento de redes de neuronas en capas. Aunque formalmente no se puede probar que la regla Delta generalizada converge a la solución ideal, como sucede con el perceptrón clásico al ser entrenado con patrones linealmente separables, sí ha sido usada, sin embargo, durante bastantes años en la solución de una gran cantidad de problemas prácticos.

Esto ha situado al perceptrón multicapa como uno de los principales modelos en uso actual. En esta sección, primeramente, se expone un modelo general de neurona. Enseguida, este modelo es extendido al caso de varias neuronas en una capa y varias capas. Después se habla acerca el perceptrón simple y como este, al ser acomodado en varias capas, puede ser aplicado para resolver algunos problemas de clasificación. Para una mejor comprensión del material presentado en esta sección se sugiere consultar [34], [28], [36] y [37].

### 6.2.3 Neurona con una entrada

En esta sección se estudia el modelo más sencillo de red neuronal que uno se puede encontrar en la literatura. Se trata de una neurona con una sola entrada. En la figura 6.4 se muestra este modelo de neurona sin factor de desplazamiento o bias (figura 6.4a y con factor de desplazamiento, figura 6.4b).

Como se puede ver en el primer caso, el escalar de entrada $x$ es multiplicado por el peso $p$, dando como resultado el producto $r = px$, antes de ser inyectado a la función $f$, la cual es conocida como función de transferencia o de activación; recibe como entrada el producto $r$ y entrega como salida el escalar $y = f(r) = f(px)$.

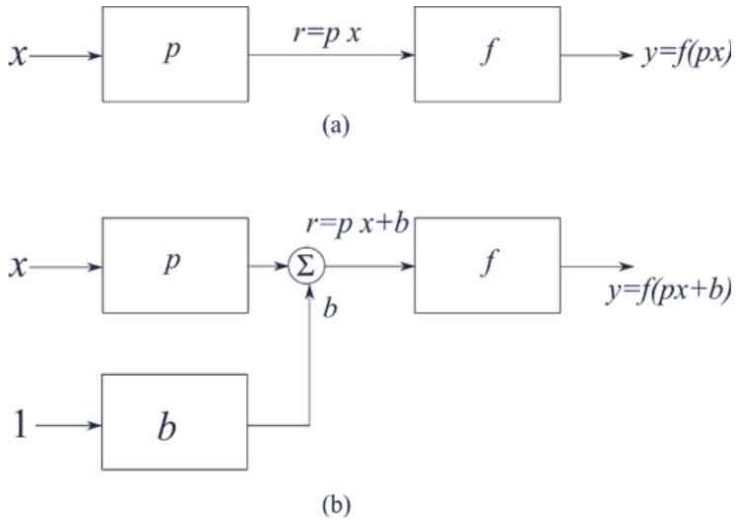



**Figura 6.4**: Modelo de una sola neurona con entrada escalar. (a) Sin factor de desplazamiento. (b) Con factor de desplazamiento.

En el caso del modelo de la figura 6.4b, al producto $px$ le es sumado el factor $b$ llamado factor de desplazamiento, antes de ser inyectado a la función $f$. Este parámetro, como su nombre lo indica, permite desplazar la función de transferencia $f$ a un lado o al otro, en el eje de referencia (dependiendo del signo de $b$), precisamente en $b$. Actúa como un peso, pero con valor multiplicativo 1. La idea de un modelo como el mostrado en la figura 6.4 es que los parámetros $p$ y $b$ pueden ser ajustados de forma que la neurona presente un comportamiento deseado. Estos parámetros pueden ser ajustados manual o automáticamente de manera que la neurona realice una tarea determinada.

### 6.2.4 Neurona con varias entradas

En lugar del escalar $x$ (ver figura 6.5), la entrada a la neurona es ahora el vector $\boldsymbol{x} = [\, x_1 \; x_2 \; \cdots x_n \,]^T$. Antes de ser presentado a la entrada de la neurona, este vector es multiplicado por el vector de pesos $\boldsymbol{p} = [\, p_1 \; p_2 \; \cdots \; p_n \,]^T$, para dar como resultado el escalar $r = \boldsymbol{p}^T \boldsymbol{x}$. Este resultado es equivalente al potencial de membrana en una neurona biológica que se obtiene por la acumulación de los potenciales post-sinápticos, como resultado del procesamiento de cada una de las sinapsis a la entrada de la neurona.

En la figura 6.5a se muestra el modelo neuronal correspondiente sin desplazamiento, mientras que en la figura 6.5b se ilustra el modelo con desplazamiento incluido. El factor de desplazamiento tiene exactamente la misma función que en el caso de la neurona con entrada escalar. La salida $y = f(\boldsymbol{p}^T \boldsymbol{x})$ o $f(\boldsymbol{p}^T \boldsymbol{x} + b)$ es de nuevo un escalar, cuyo valor vendrá dado por el tipo de función de transferencia adoptada.

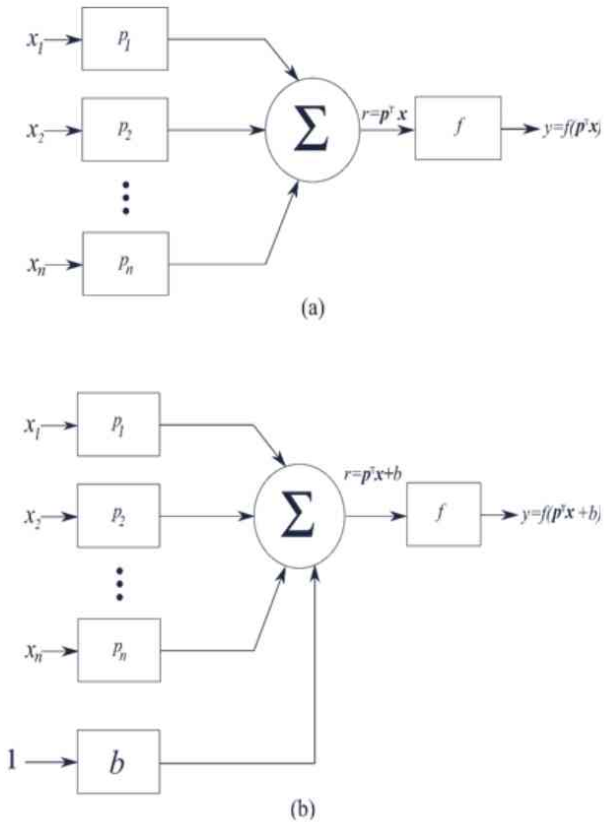



**Figura 6.5**: Modelo de una sola neurona con entrada vectorial. (a) Sin factor de desplazamiento. (b) Con factor de desplazamiento.

### 6.2.5 Ejemplos de funciones de transferencia

Hay una gran variedad de funciones de transferencia $f$ que pueden ser usadas para determinar la salida de la neurona. Tres de las más usadas y algunas de sus variantes se muestran en la figura 6.6. La función de transferencia limitadora (figura 6.6a), por ejemplo, satura la salida de la neurona al valor 0 si el argumento de la función es menor que 0 y 1 si dicho argumento es igual o mayor que 0. La función de transferencia sigmoidea (figura 6.6e), por otro lado, puede recibir como entrada valores entre menos infinito y más infinito, y entrega como salida valores entre 0 y 1. Cuando las entradas a la neurona son de tipo booleano (esto es 0 y 1), y cuando la función de activación es la función limitadora se obtiene, como se verá más adelante, la neurona tipo UUL.

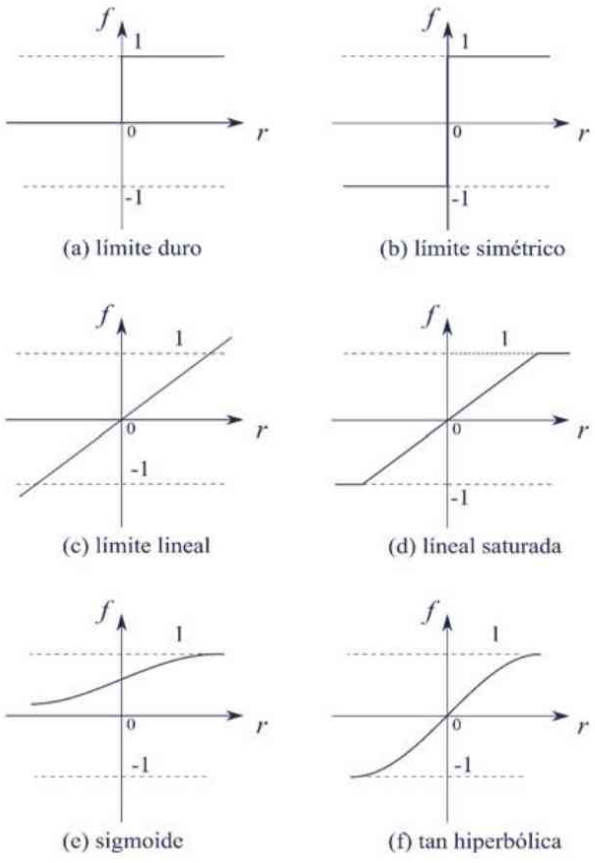



**Figura 6.6**: Algunas funciones de transferencia que pueden ser usadas para entrenar una neurona artificial.

## 6.3 Arquitecturas básicas de RNA

Dos o más neuronas pueden ser acomodadas para formar una capa. Asimismo, varias capas de neuronas pueden conformar una red neuronal artificial multicapa. En esta sección, se estudiarán algunas de las arquitecturas básicas que se utilizan en redes neuronales artificiales (RNA).

### 6.3.1 Red de una capa de neuronas

En la figura 6.7 se muestra un arreglo de $q$ neuronas como la de la figura 6.7b. En este tipo particular de red neuronal, cada neurona realiza el producto interno entre el vector de entrada $\boldsymbol{x} = [\, x_1 \quad x_2 \quad \cdots \quad x_n \,]^T$ y el vector de pesos $\boldsymbol{p}_j = [\, p_{j,1} \quad p_{j,2} \quad \cdots \quad p_{j,n} \,]^T$, con $j = 1, 2, \cdots, q$; produciendo como salida el escalar $r_j = \boldsymbol{p}_j^T \boldsymbol{x}$.

Por lo que, la red entrega como salida el vector $\boldsymbol{y} = [\, y_1 \quad y_2 \quad \cdots \quad y_q \,]^T$, donde el $j$-ésimo elemento del vector $\boldsymbol{y}$ se encuentra dado de la siguiente forma:

$$y_j = f_j(r_j) = f_j(\boldsymbol{p}_j^T \boldsymbol{x}) = f_j(p_{j,1}x_1 + p_{j,2}x_2 + \cdots + p_{j,n}x_n).$$

Los $q$ vectores de pesos pueden ser acomodados en una forma matricial $P \in \mathbb{R}^{n \times q}$ como a continuación se indica:

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{q,1} & p_{q,2} & \cdots & p_{q,n} \end{bmatrix} \tag{6.1}$$

donde $p_{ji} \in \mathbb{R}$, con $j = 1, 2, \cdots q$ e $i = 1, 2, \cdots n$.

El entrenamiento de esta red neuronal consiste en el ajuste de los elementos de la matriz de pesos $P$. Si dicho ajuste se realiza en forma iterada, como veremos más adelante, cada vector $\boldsymbol{p}_j = (\, p_{j,1} \quad p_{j,2} \quad \cdots \quad p_{j,n} \,)^T$, con $j = 1, 2, \cdots, q$ es ajustado en forma independiente, pero al mismo tiempo, durante la misma iteración.

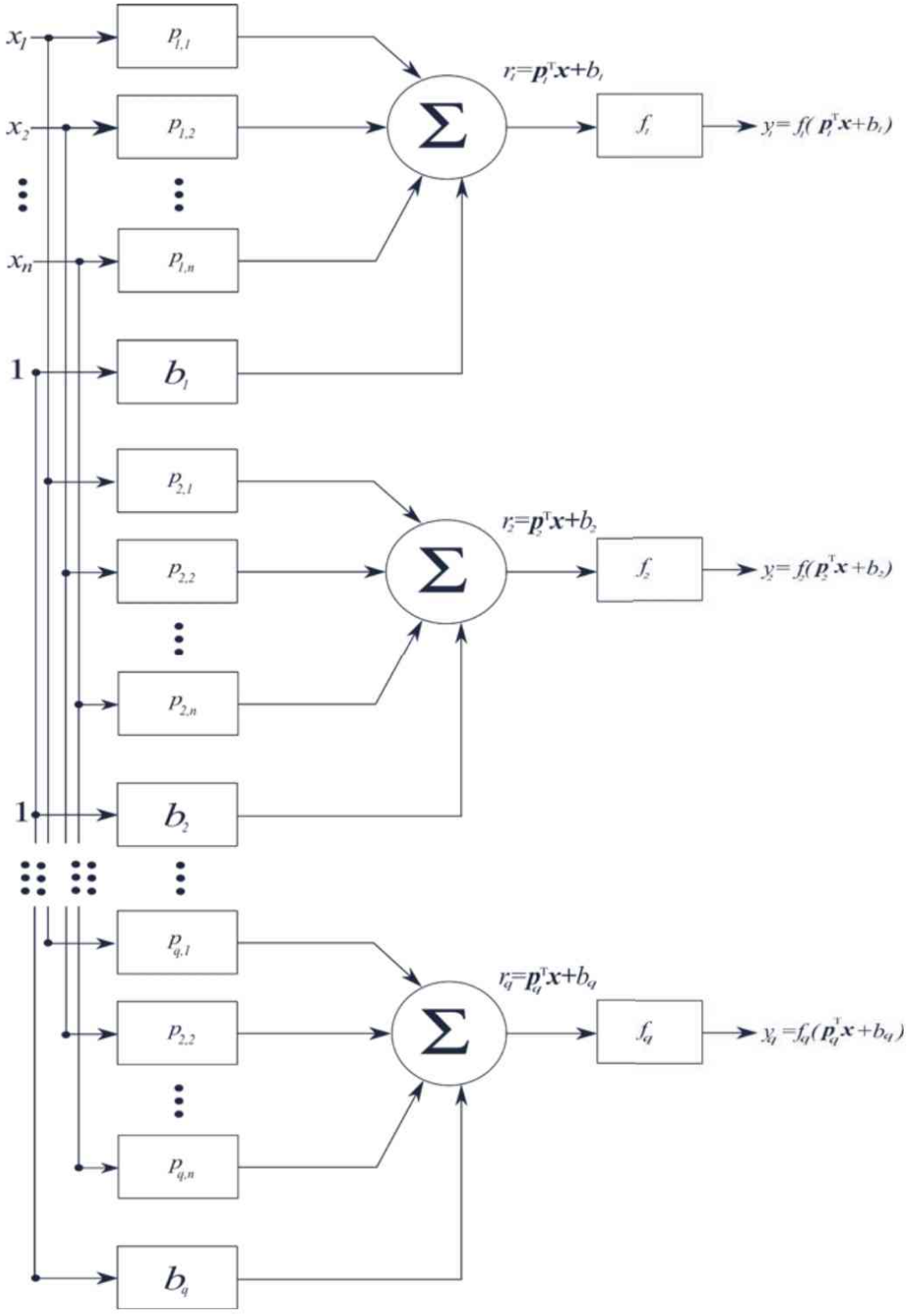



**Figura 6.7**: Arquitectura de una red neuronal de una capa con $q$ neuronas.

### 6.3.2 Red con varias capas de neuronas

Varias capas de neuronas pueden ser acomodadas para formar una RNA. En la figura 6.8 se muestra una de las arquitecturas más sencillas. En esta arquitectura, cada capa se compone de una matriz de pesos $P$, un vector de bias $\boldsymbol{b}$ y un vector de salida $\boldsymbol{y}$.

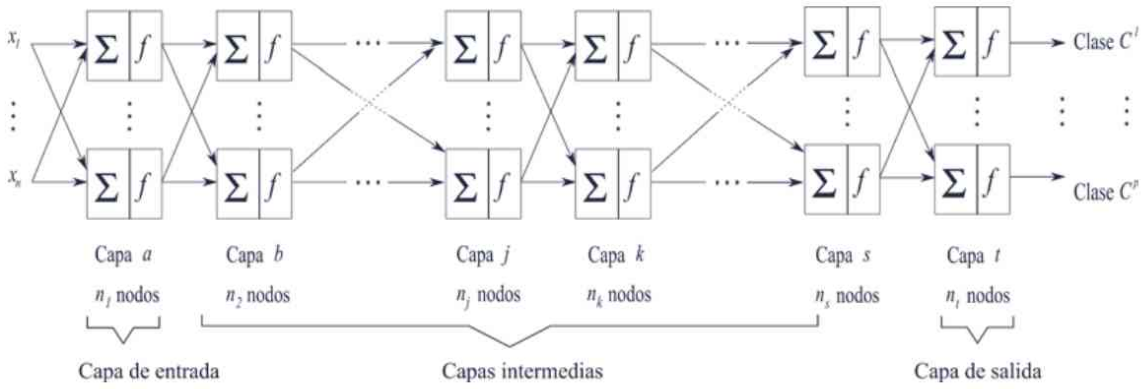



**Figura 6.8**: Arquitectura de una red neuronal con varias capas.

En un arreglo de neuronas como el mostrado en la figura 6.8, cada capa puede contener un número diferente de neuronas. En particular, la primera capa contiene $n_1$ neuronas, la segunda $n_2$ neuronas, y así hasta la última capa con $n_t$ neuronas. Note también cómo las salidas de cada capa, desde la primera hasta la penúltima, constituyen la entrada de la siguiente capa. La matriz de pesos entre la capa $j$ y la capa $k$ viene dada como:

$$P_{kj} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n_j} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n_j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n_k,1} & p_{n_k,2} & \cdots & p_{n_k,n_j} \end{bmatrix}. \tag{6.2}$$

Cada una de las capas de esta arquitectura desempeña un papel diferente. La que está más a la derecha se llama capa de salida, ya que entrega como salida el resultado total de procesamiento de la red completa. La primera capa, por recibir como entrada el vector $\boldsymbol{x} = [\, x_1 \quad x_2 \quad \cdots \quad x_n \,]^T$, se llama capa de entrada. Las capas entre la de entrada y de salida procesan resultados intermedios y se llaman capas intermedias o escondidas.

Las redes neuronales multicapa son muy útiles para resolver muchos tipos de problemas. Enseguida se estudiarán algunos de los modelos más usados, así como los mecanismos que permiten su entrenamiento para la solución de problemas. Se empezará por la sencilla unidad de umbralado lógico (UUL o TLU) y el conocido perceptrón, la ADALINE y la neurona sigmoidal.

## 6.4 Modelos de neuronas artificiales

En esta sección se estudiará el primer modelo de neurona artificial, la Unidad de Umbralado Lógico. En las siguientes secciones se estudiará el perceptrón, la ADALINE y la neurona sigmoidal. Se habla de sus componentes, así como de las respectivas reglas para el ajuste de sus parámetros libres. Para todos los modelos se presentan ejemplos ilustrativos para una mejor comprensión y asimilación del material descrito.

### 6.4.1 Unidad de umbralado lógico (UUL)

Este modelo de neurona fue introducido al mundo en 1943 por los científicos McCulloch y Pitts en 1943 [38]. Fue el primer modelo de neurona propuesto por la comunidad científica. La UUL sin desplazamiento o bias, es una neurona con la estructura mostrada en la figura 6.9. Como se puede ver, el procesamiento de la información de entrada se lleva a cabo en dos pasos. El primer paso involucra el cálculo de un producto punto entre el vector de pesos $\boldsymbol{p}$ y el vector de entrada $\boldsymbol{x}$. El resultado es el escalar:

$$r = \boldsymbol{p}^T\boldsymbol{x} = \sum_{i=1}^{n} p_i x_i. \tag{6.3}$$

El segundo paso involucra la comparación del escalar $r$ con un umbral, digamos $\theta$. La neurona entonces emite un resultado dado por la ecuación de la función limitadora, como sigue:

$$s = \begin{cases} 1 & \text{si} \quad r \geq \theta \\ 0 & \text{si} \quad r < \theta \end{cases} \tag{6.4}$$

De acuerdo con la figura 6.9, la UUL permite separar conjuntos patrones de entrada en dos clases o categorías, 1 para un conjunto de patrones y 0 para otro conjunto. Desde el punto de vista de la clasificación, estas categorías pueden pensarse como dos regiones en un

espacio multidimensional separadas por un híperplano. Al observar la figura 6.9, el lector puede notar que la neurona recibe a su entrada $n$ valores. Cada uno de estos $n$ valores es multiplicado por un peso $p_i,\ i = 1, 2, \cdots, n$.

El resultado a la salida de neurona (el círculo con una sumatoria encerrada) es precisamente: $r = \boldsymbol{p}^T \boldsymbol{x} = \sum_{i=1}^{n} p_i x_i$, como lo estipula la ecuación (6.3). Para mejor apreciar la operación de una UUL consideremos los siguientes dos ejemplos:

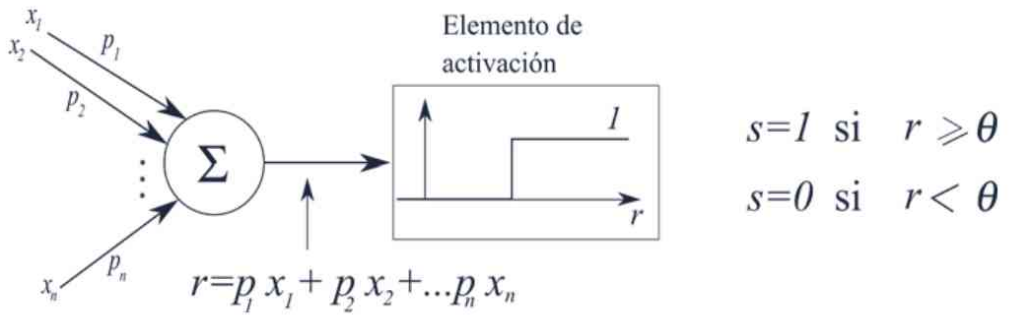



**Figura 6.9**: Arquitectura de una UUL.

## Ejemplo 6.1

Suponga que $n = 3$ y si $x_1 = 0.5$, $x_2 = 0.3$, $x_3 = 0.2$, $p_1 = -0.2$, $p_2 = 0.3$ y $p_3 = -0.4$, entonces: $r = -(0.2)0.5 + (0.3)0.3 - 0.4(0.2) = -0.09$.

Al aplicar la ecuación (6.4) tomando el resultado anterior como argumento, entonces la salida total de la neurona viene dada como $s = f(-0.09) = 0$.

## Ejemplo 6.2

Del ejemplo inmediato anterior, suponga de nuevo que $n = 3$, $x_1 = 0.5$, $x_2 = 0.3$, $x_3 = 0.2$, $p_1 = -0.2$, $p_2 = 0.3$ y $p_3 = 0.4$, entonces: $r = -0.2(0.5) + 0.3(0.3) + 0.4(0.2) = 0.7$.

Ahora, aplicando la ecuación (6.4), entonces la salida total de la neurona viene dada como $s = f(0.7) = 1$.

### 6.4.2 Separabilidad lineal entre clases

Al estudiar las ecuaciones (6.3) y (6.4), uno se puede percatar que la condición de disparo de la UUL (que la neurona pase del estado 0 al estado 1) se da cuando $r = \theta$. Para el caso de dos entradas:

$$p_1 x_1 + p_2 x_2 \quad = \quad \theta. \tag{6.5}$$

Al restar $p_1 x_1$ a ambos lados de esta ecuación:

$$p_2 x_2 = -p_1 x_1 + \theta.$$

Ahora, al dividir ambos lados por $p_2$, se tiene que:

$$x_2 = -\frac{p_1}{p_2} x_1 + \frac{\theta}{p_2}. \tag{6.6}$$

Note lo difícil que es darse cuenta de que esta ecuación es de la forma general de una recta:

$$x_2 = a x_1 + b \tag{6.7}$$

donde $a,\ b \in \mathbb{R}$ son constantes.

La ecuación (6.7) describe una recta con pendiente $a$ e intercepto $b$ con el eje $x_2$. Esta recta, en la literatura, es conocida como línea de decisión o de separación. Note que un sinnúmero de líneas puede ser dibujado sobre el plano. Algunas de estas puede que logren la separabilidad entre los patrones de entrada, otras puede que no lo hagan. Por supuesto para el caso de tres rasgos la separación se da mediante un plano de separación. En el caso general de $n$ rasgos, la separación se da mediante un híperplano de separación. Para ilustrar el concepto de separabilidad lineal entre clases enseguida se estudiará un ejemplo.

De esta discusión se puede observar, por un lado, que los pesos $p_1$ y $p_2$ favorecen el que la recta correspondiente experimente una pendiente. Por otro lado, se puede ver que el parámetro $\theta$ permite que la recta se desplace por plano. Sin este parámetro (lo cual sería equivalente a decir que $\theta = 0$) la recta sí tendría un inclinación pero siempre pasando por el origen del sistema coordenado. Lo dicho, por supuesto, es válido en $n$ dimensiones.

Como se verá enseguida, el uso de los $n$ pesos $p_1, p_2, \cdots, p_n$ y el parámetro $\theta$, en combinación con una función de activación como la dada por la ecuación (6.4), permitirán resolver problemas de clasificación, entre otros.

**• Ejemplo 6.3**

Considérese el siguiente conjunto de patrones y sus respectivas clases de pertenencia:

$$\left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right\} \in C^1 \quad \text{y} \quad \left\{\begin{bmatrix}1\\1\end{bmatrix}\right\} \in C^2$$

Proponer un vector de pesos $\boldsymbol{p} = [p_1 \quad p_2]^T$ y un umbral $\theta$, tal que los patrones queden clasificados en sus respectivas clases: $C^1$ o $C^2$.

**Solución**

Al representar los patrones anteriores en el plano, se obtiene la figura 6.10a. Se puede ver que estos patrones y sus clases de pertenencia tienen una relación directa con el problema de la operación lógica "Y". En la figura 6.10b se muestra la tabla de verdad de esta función.

La figura 6.10c muestra una de tantas líneas que permiten separar los patrones de las dos clases. En general, si para un problema dado con cualquier número de rasgos es posible encontrar al menos un híperplano de separación, entonces se dice que el problema es linealmente separable.

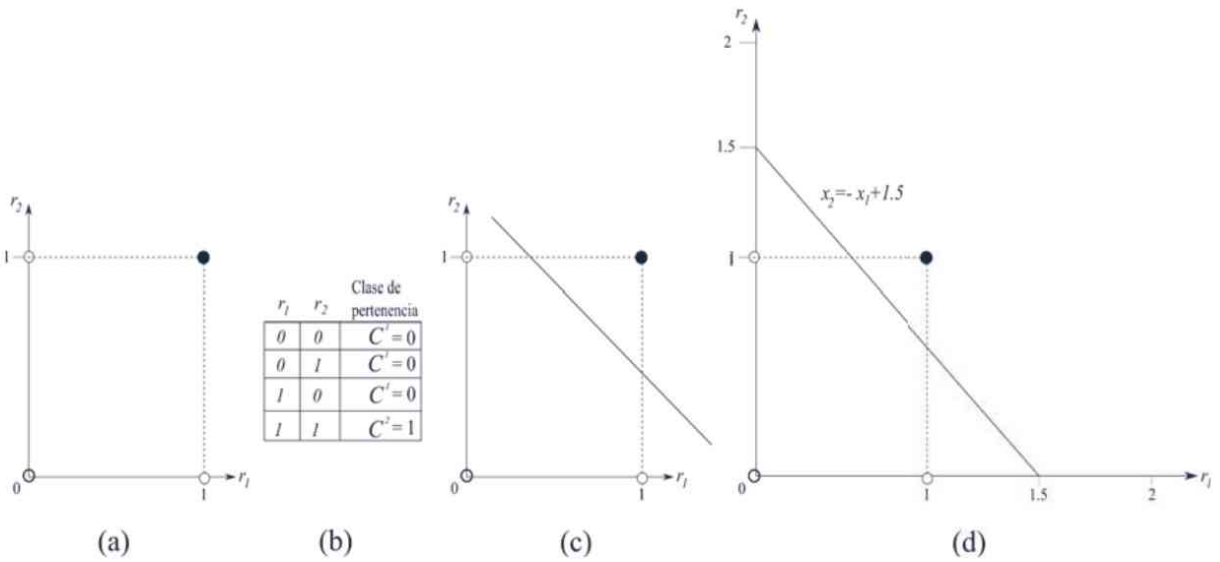



**Figura 6.10**: (a) Conjunto de patrones y (b) su tabla de clasificación. (c) Una línea que permite la separación lineal entre las clases. (d) La línea de separación calculada mediante la ecuación (6.5) con $p_1 = p_2 = 1$ y $\theta = 1.5$.

Pasemos ahora al uso de la ecuación (6.6) para calcular una recta de separación para el caso de dos clases. Para esto tómese el conjunto de patrones del ejemplo 6.1. Como primera aproximación supongamos que $p_1 = p_2 = 1$ y $\theta = 1.5$. Luego entonces: $x_2 = -x_1 + 1.5$.

En la figura 6.10d se muestra esta línea. Se puede ver que aquella logra la separación de los patrones en sus dos clases: $C^1 = 0$ y $C^2 = 1$. Debido a que esta línea permite resolver el problema se opta por mantenerla como solución. No es difícil darse cuenta de que el problema, de encontrar una línea de separación consiste en encontrar el vector de pesos y el umbral asociados. Para esto se puede proceder de varias formas. Una de ellas puede ser por prueba y error al aplicar los siguientes pasos, provocando un ajuste de estos parámetros:

1) Proponer un conjunto de valores: $p_1$, $p_2$, $\theta$ y calcular la línea de decisión correspondiente.
2) Para esta línea, probar si todo el conjunto de patrones es clasificado correctamente.
3) En caso afirmativo, seleccionar esta solución como válida y terminar.
4) En caso fallido, regresar al paso 1).

Es claro que en el caso general este proceso de búsqueda puede tomar mucho tiempo; más aún, sin un conocimiento del problema que se intenta resolver. Lo mejor es usar algún método que, de manera automática, permita ajustar mediante un proceso iterado los valores de los pesos y del umbral.

Antes de explicar la operación de un método para la estimación de estos parámetros, se demostrará que una UUL es un separador lineal entre clases de patrones.

### 6.4.3 La UUL como separador lineal entre clases

Ya se vio que la UUL se comporta como un separador lineal entre clases. Esta operación resulta de combinada de las dos operaciones que la componen:

1) El producto punto entre dos vectores, el de pesos de la red y el vector de entrada.
1) La operación de comparación entre este producto punto con un umbral.

Para dos vectores, digamos, $\boldsymbol{v}$ y $\boldsymbol{p}$ separados por un ángulo $\phi$, el producto interno (producto escalar) entre los vectores $\boldsymbol{p}$ y $\boldsymbol{v}$, $\boldsymbol{p} \cdot \boldsymbol{v}$, se define como:

$$\boldsymbol{p} \cdot \boldsymbol{v} = \|\boldsymbol{p}\|\|\boldsymbol{v}\| \cos(\phi). \tag{6.8}$$

Nótese que en este caso particular se cumple la propiedad conmutativa: $\boldsymbol{p} \cdot \boldsymbol{v} = \boldsymbol{v} \cdot \boldsymbol{p}$.

Es bien sabido que este producto, en términos geométricos, indica qué tan alineados están dos vectores. Si las longitudes de los vectores se consideran fijas entonces $\boldsymbol{p} \cdot \boldsymbol{v}$ depende solo del $\cos(\phi)$. De acuerdo con esto, se tienen tres casos:

1) Si los dos vectores están más o menos alineados el producto interno es positivo.
2) Cuando $\boldsymbol{p}$ y $\boldsymbol{v}$ forman un ángulo de 90 grados: $\boldsymbol{p} \cdot \boldsymbol{v} = 0$.
3) Cuando los dos vectores apuntan más o menos en direcciones opuestas, el producto interno es negativo.

### 6.4.4 Producto interno (forma algebraica)

Es bien conocido que al aplicar las propiedades de descomposición de un vector en sus componentes, así como la suma de vectores, se pude demostrar que para cualesquiera dos vectores $\boldsymbol{p},\ \boldsymbol{v} \in \mathbb{R}^2$:

$$\boldsymbol{p} \cdot \boldsymbol{v} = p_1 v_1 + p_2 v_2. \tag{6.9}$$

En $n$ dimensiones es claro que:

$$\boldsymbol{p} \cdot \boldsymbol{v} = \sum_{i=1}^{n} p_i v_i. \tag{6.10}$$

Este valor se debe interpretar igual que en el caso 2-D:

1) Si $\boldsymbol{p} \cdot \boldsymbol{v}$ es positivo, entonces los dos vectores apuntan más o menos en la misma dirección.
2) Si $\boldsymbol{p} \cdot \boldsymbol{v}$ es negativo, entonces los dos vectores apuntan en direcciones opuestas.
3) Si $\boldsymbol{p} \cdot \boldsymbol{v} = 0$, entonces los vectores forman un ángulo recto.

Es prácticamente imposible visualizar esto en $n$ dimensiones, sin embargo, se puede hacer la analogía con el caso 2-D.

**Comentario**. Es claro, si $\boldsymbol{p} = \boldsymbol{v}$, entonces:

$$\boldsymbol{v} \cdot \boldsymbol{v} \quad = \quad \boldsymbol{v}^T \boldsymbol{v} = \sum_{i=1}^{n} v_i v_i = \sum_{i=1}^{n} v_i^2 = \|\boldsymbol{v}\|^2. \tag{6.11}$$

En otras palabras, el cuadrado de la longitud de un vector es igual al producto interno del vector consigo mismo.

### 6.4.5 Proyección de un vector

De nuevo, sean dos vectores $\boldsymbol{v}$ y $\boldsymbol{p}$. Uno puede preguntarse: ¿Qué magnitud de $\boldsymbol{v}$ se proyecta sobre $\boldsymbol{p}$? Esta porción del vector $\boldsymbol{v}$ sobre $\boldsymbol{p}$ denotada como $\boldsymbol{v_p}$, se conoce como proyección del vector $\boldsymbol{v}$ sobre el vector $\boldsymbol{p}$. Gráficamente este concepto se muestra en la figura 6.11.

Al usar la definición del coseno:

$$\boldsymbol{v_p} \quad = \quad \|\boldsymbol{v}\| \cos(\phi). \tag{6.12}$$

Al multiplicar y dividir la expresión (6.12) por la magnitud de $\|bfp\|$:

$$\boldsymbol{v_p} \quad = \quad \frac{\|\boldsymbol{v}\|\|\boldsymbol{p}\| \cos(\phi)}{\|\boldsymbol{p}\|} = \frac{\boldsymbol{v} \cdot \boldsymbol{p}}{\|\boldsymbol{p}\|}. \tag{6.13}$$

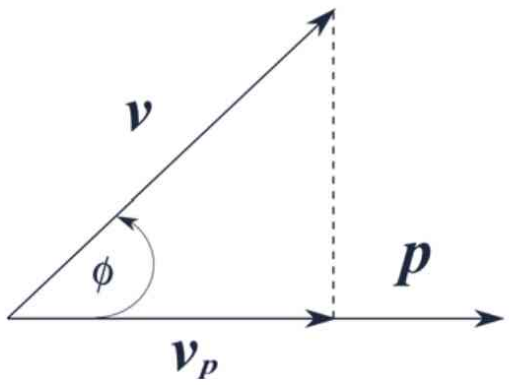



**Figura 6.11**: Proyección de un vector $\boldsymbol{v}$ sobre un vector $\boldsymbol{p}$.

Se puede ver que la proyección del vector $\boldsymbol{v}$ sobre el vector $\boldsymbol{p}$ viene dada por el producto punto entre ambos vectores, normalizada por el tamaño del vector $\boldsymbol{p}$.

### 6.4.6 UUL y separabilidad lineal

Tomando como base lo dicho anteriormente, enseguida se probará que una UUL es una máquina de separación lineal, universal e independiente del número de rasgos descriptores a su entrada. En otras palabras, esta máquina no depende de la dimensión del espacio de los patrones.

Veamos: ya se explicó que al usar el concepto de producto interno, la activación de una UUL de $n$ entradas se puede expresar como: $r = \boldsymbol{p} \cdot \boldsymbol{v}$. De igual forma, de acuerdo con lo visto $\boldsymbol{p} \cdot \boldsymbol{v} = \theta$. Por otro lado, para un vector de entrada arbitrario $\boldsymbol{v}$, ya se vio que su proyección sobre $\boldsymbol{p}$ viene dada como: $\boldsymbol{v_p} = \frac{\boldsymbol{v} \cdot \boldsymbol{p}}{\|\boldsymbol{p}\|}$. Si sobre esta ecuación se impone la restricción dada por $\boldsymbol{p} \cdot \boldsymbol{v_p} = \theta$, entonces: $\boldsymbol{v_p} = \frac{\theta}{\|\boldsymbol{p}\|}$.

Geométricamente (ver figura 6.12), la relación $\boldsymbol{p} \cdot \boldsymbol{v} = \theta$ define una línea perpendicular al vector de pesos $\boldsymbol{p}$. Esta recta define claramente dos regiones, una de un lado de la recta y otra al otro lado. Sean estas dos regiones $A$ y $B$. Si se asume que $\boldsymbol{p}$ y $\theta$ son constantes en dos dimensiones, eventualmente la proyección $\boldsymbol{v_p}$ de $\boldsymbol{v}$ puede llegar a tocar la línea perpendicular al vector $\boldsymbol{p}$, incluso puede sobrepasarla. En otras palabras, la punta de la proyección $\boldsymbol{v_p}$ de $\boldsymbol{v}$ sobre $\boldsymbol{p}$ puede quedar a un lado de la recta, sobre la recta o al otro lado de la misma (ver figuras 6.12a, 6.12b y 6.12c).

En detalle:

a) Si $\boldsymbol{v_p} < \frac{\theta}{\|\boldsymbol{p}\|}$ (figura 6.12a), la proyección de $\boldsymbol{v}$ es corta y su punta no alcanza a tocar a la recta perpendicular al vector $\boldsymbol{p}$. Por tanto, $\boldsymbol{v}$ yace en la región $A$. En este caso $\boldsymbol{p} \cdot \boldsymbol{v} < \theta$ y por tanto, para nuestra UUL, de acuerdo con la ecuación (6.4), $s = 0$. En otras palabras, la UUL no dispara.

b) Si ahora, $\boldsymbol{v_p} \geq \frac{\theta}{\|\boldsymbol{p}\|}$ (figuras 6.12b y 6.12c). En este caso la proyección de $\boldsymbol{v}$ es lo suficientemente larga para que su punta toque o incluso sobrepase la recta perpendicular al vector $\boldsymbol{p}$. Por tanto $\boldsymbol{v}$ yace en la región $B$. En este caso, $\boldsymbol{p} \cdot \boldsymbol{v} \geq \theta$ y por tanto para nuestra UUL de nuevo, de acuerdo con la ecuación (6.4), $s = 1$. En otras palabras, la UUL dispara.

Con esto se puede ver que la UUL funciona exactamente como un separador lineal entre clases, definidas por sus regiones correspondientes. Otra cosa más: Los resultados presentados son generales y son independientes del número de entradas de la UUL.

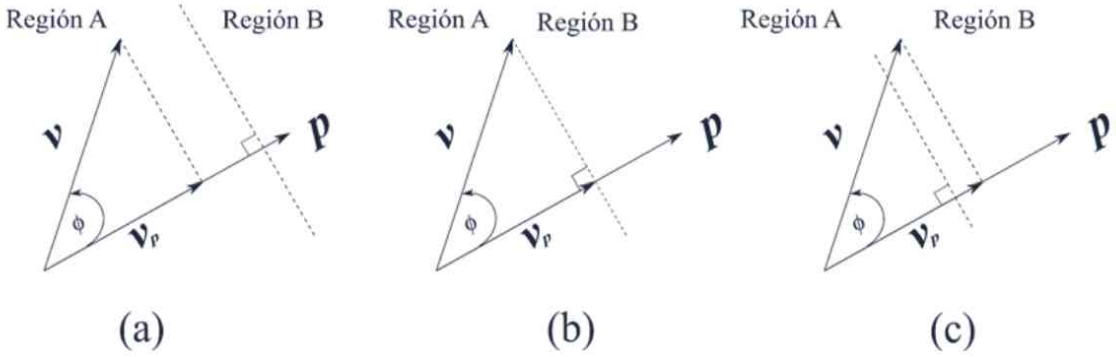



**Figura 6.12**: Proyección de un vector $\boldsymbol{v}$ sobre un vector $\boldsymbol{p}$.(a) La punta de la proyección queda antes de la línea de separación. (b) La punta toca la línea de separación. (c) La punta sobrepasa la línea de decisión.

En resumen:

1) La activación de una UUL viene dada como el producto interno entre el vector de entrada y el vector de pesos: $r = \boldsymbol{p} \cdot \boldsymbol{v}$.

2) La relación $\boldsymbol{p} \cdot \boldsymbol{v} = \theta$ define, en lo general, un hiper-plano en el espacio de patrones que es perpendicular al vector de pesos.

3) A un lado de este hiperplano yacen todos los patrones que son clasificados por la UUL como "0", mientras que todos los patrones que yacen al otro lado del hiperplano son clasificados como "1".

4) El hiperplano es la superficie de decisión de la UUL. Ya que esta superficie es una versión $n$-dimensional de una línea recta, la UUL resulta un clasificador lineal.

Si el conjunto de patrones no puede ser separado por un hiperplano, entonces la UUL no podrá clasificarlos.

### 6.4.7 Aprendizaje en una RNA

Una propiedad muy importante de una RNA es su capacidad de aprender del medio ambiente donde se encuentra inmersa. La RNA aprende de su entorno a través de un proceso iterativo de ajuste de sus pesos y desplazamientos. Este proceso de aprendizaje en el contexto de las RNA puede ser descrito como sigue:

**Aprendizaje** es el proceso mediante el cual los parámetros libres de una RNA son adaptados a través de una estimulación del entorno donde está la RNA.

Este tipo de aprendizaje queda determinado por la manera en que los parámetros son ajustados. Esta definición involucra los siguientes tres eventos:

1) La RNA es estimulada por un medio ambiente.

2) Los parámetros libres (pesos) de la RNA experimentan cambios por esta estimulación.

3) La RNA responde de una nueva manera al ambiente por los cambios en su estructura interna.

Sin pérdida de generalidad, considere el caso de una sola neurona alimentada con un vector $\boldsymbol{x}(n)$ en el tiempo discreto $n$, la salida es $y(n)$, la cual es comparada con la respuesta deseada de salida $d(n)$, como se muestra en la figura 6.13. El error $e(n) = d(n) - y(n)$ es la señal de control usada para el ajuste iterativo de los parámetros libres de la RNA.

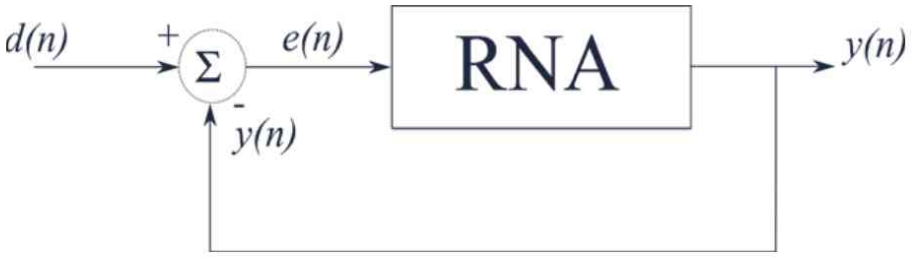



**Figura 6.13**: Esquema para el ajuste de los pesos de la red mediante el cálculo del error entre la salida deseada y la salida de la red.

### 6.4.8 Entrenamiento de una UUL

Se ha visto que toda UUL induce una superficie de separación. Ahora bien, como esta superficie depende del vector de pesos y del umbral, es necesario su ajuste (entrenamiento) para encontrar dicha superficie.

Este ajuste se lleva a cabo mediante un proceso iterativo a partir de un conjunto de patrones, conocido como entrenamiento de la red. Al conjunto de patrones se le llama conjunto de entrenamiento.

El ajuste de los pesos y del umbral se lleva a cabo mediante una regla de aprendizaje bien establecida.

El proceso de diseño de la UUL se realiza desde el punto de vista del entrenamiento supervisado (capítulo 2), esto es que para cada patrón del conjunto de entrenamiento se conoce de antemano la clase de pertenencia.

Sea el siguiente conjunto de entrenamiento:

$$\left(\left\{\boldsymbol{x}_i = [x_{i1},\ x_{i2}, \cdots, x_{in}]^T\right\},\ \{t_i\}\right), \qquad i = 1, 2, \cdots, n;\ \{t_i\} = \{0, 1\}. \tag{6.14}$$

A fin de poder entrenar el umbral como un peso, como se propone en la literatura, se hace un pequeño truco matemático. Ya se vio que para que la UUL emita el valor de 1, entonces utilizando el producto punto:

$$\boldsymbol{p} \cdot \boldsymbol{x} \quad \geq \quad \theta \Longrightarrow \boldsymbol{p} \cdot \boldsymbol{x} + (-1)\theta \geq 0.$$

- Nótese que en este caso se puede pensar al umbral como un peso extra colgado a un valor de -1.

- Particularmente a este umbral negativo (-1), en la literatura se le conoce como desplazamiento, puesto que técnicamente proviene del inglés *bias*.

Tomando en cuenta esto, entonces resulta que el vector de pesos aumentado queda como: $(\,p_1 \quad p_2 \quad \cdots \quad p_n \quad p_{n+1} = \theta\,)^T$.

Con base en este nuevo vector aumentado, la función de la UUL quedaría como:

$$\text{Si } \boldsymbol{p} \cdot \boldsymbol{x} \geq 0 \quad \longrightarrow \quad y = 1$$

$$\text{Si } \boldsymbol{p} \cdot \boldsymbol{x} < 0 \quad \longrightarrow \quad y = 0.$$

Ahora: $\boldsymbol{p} \cdot \boldsymbol{x} = 0$, define el hiperplano de separación que es perpendicular al vector de pesos que pasa por el origen.

Para visualizar esto, considérese el siguiente ejemplo.

**● Ejemplo 6.4**

De acuerdo con las figuras 6.10a y 6.10b, supongamos una UUL que emite el valor de "1" para la entrada (1,1) y "0" para las otras tres entradas. Si $p_1 = 0.5,\ p_2 = 0.5$ y $\theta = 0.75$, la línea de separación, de acuerdo con lo que ya vimos, pasa por los puntos: $x_1 = (0.5, 1)$ y $x_1 = (1, 0.5)$.

**Solución**

Para el patrón de pesos aumentado todos los patrones tienen la forma: $(x_1, x_2, -1)$, ya que como vimos, la tercera componente está anclada al valor constante -1. Note que el vector aumentado tiene una tercera componente igualada al umbral y además es perpendicular al plano que pasa por el origen.

Enseguida se estudiará una primera regla que permite llevar a cabo un ajuste automático de los pesos de una neurona, esto incluye el umbral de disparo.

### 6.4.9 La regla de entrenamiento del perceptrón

Supóngase ahora que se quiere entrenar una UUL para que opere como separador lineal entre dos clases, y esto a partir de un conjunto de entrenamiento dado por la ecuación (6.14). Sin pérdida de generalidad, supóngase que $n = 3$.

Para desarrollar el método de entrenamiento consideraremos que pueden ocurrir dos casos:

**Caso 1.** Se presenta a la entrada de la UUL un vector $\boldsymbol{x}$ con respuesta deseada $t = 1$, pero que el vector actual de pesos $y = 0$. La UUL ha clasificado mal al vector $\boldsymbol{x}$. En consecuencia, el vector de pesos $\boldsymbol{p}$ debe ser ajustado.

Para producir 0, la activación debió haber sido negativa, por tanto el producto $\boldsymbol{p} \cdot \boldsymbol{x}$ fue negativo. Esto es $\boldsymbol{p}$ y $\boldsymbol{x}$ apuntan más o menos en sentidos opuestos.

A fin de corregir esto es necesario rotar un poco al vector $\boldsymbol{p}$ hacia el vector $\boldsymbol{x}$. Lo cual se puede lograr al sumar una fracción de $\boldsymbol{x}$ a $\boldsymbol{p}$ como sigue:

$$\boldsymbol{p}' = \boldsymbol{p} + \alpha \boldsymbol{x}. \tag{6.15}$$

con $0 < \alpha < 1$.

**Caso 2.** Se presenta a la entrada de la UUL un vector $\boldsymbol{x}$ con respuesta deseada $t = 0$, pero que el vector actual de pesos da $y = 1$. La UUL, de nuevo, ha erróneamente clasificado al vector $\boldsymbol{x}$.

Luego entonces el vector de pesos debe ser ajustado. La activación fue positiva; por tanto, el producto $\boldsymbol{p} \cdot \boldsymbol{x}$ fue positivo.

Ahora, $\boldsymbol{p}$ y $\boldsymbol{x}$ apuntan más o menos en el mismo sentido. Se requiere ahora rotar un poco $\boldsymbol{p}$ de manera que se aleje de $\boldsymbol{x}$.

Esto se puede lograr al restar una fracción de $\boldsymbol{x}$ a $\boldsymbol{p}$ como sigue:

$$\boldsymbol{p}' = \boldsymbol{p} - \alpha \boldsymbol{x}. \tag{6.16}$$

Debido a que las diferencias se dan entre $t$ e $y$, como se sugiere en la literatura, las ecuaciones (6.15) y (6.16) se pueden combinar en una sola para obtener el correspondiente ajuste del vector de pesos:

$$\boldsymbol{p}' = \boldsymbol{p} + \alpha\,(t - y)\,\boldsymbol{x}. \tag{6.17}$$

En términos del cambio en los pesos $\triangle \boldsymbol{p} = \boldsymbol{p}^{'} - \boldsymbol{p}$, la ecuación anterior queda como:

$$\triangle \boldsymbol{p} \quad = \quad \boldsymbol{p}^{'} - \boldsymbol{p} = \alpha\,(t-y)\,\boldsymbol{x}. \tag{6.18}$$

De manera individual, se puede demostrar que, para cada componente, este cambio se ve manifestado como:

$$\begin{aligned} \triangle p_i &= \alpha\,(t-y)\,x_i,\ i = 1, 2, \cdots, n+1 \qquad (6.19)\\ p_{n+1} &= \theta \\ x_{n+1} &= -1. \end{aligned}$$

El parámetro $\alpha \in (0, 1)$ es conocido en la literatura como tasa de crecimiento, entrenamiento o ajuste.

Con base en toda esta discusión se puede establecer la siguiente regla para el ajuste de los pesos de una UUL, la cual es conocida en la literatura como regla de entrenamiento del perceptrón, ya que fue usada por primera vez para ajustar los pesos de este tipo particular de neurona.

**Comentario**. Si el problema es linealmente separable, el lector puede demostrar que este algoritmo generará un vector de pesos válido para el problema en cuestión, si dicho vector existe.

Esto ya fue demostrado hace varias décadas y se conoce como el teorema de convergencia del perceptrón, el cual establece que si dos clases de vectores $C^1$ y $C^2$ son linealmente separables, la aplicación del Algoritmo de Entrenamiento del Perceptrón (AEP) eventualmente resultará en un vector de pesos $\boldsymbol{p}$, tal que dicho vector $\boldsymbol{p}$ define una UUL cuya superficie de decisión separa a $C^1$ y $C^2$.

Una propiedad de esta proposición es que una vez que el vector $\boldsymbol{p}$ ha sido encontrado, este permanece estable, lo que significa que no se dan más cambios en el vector de pesos.

Diversas pruebas de este importante resultado pueden ser encontradas en las siguientes referencias [31], [35], [40] y [46].

**Pseudocódigo 6.1**

Regla de entrenamiento para la TLU (regla del perceptrón)

Poner los valores del vector de pesos $\boldsymbol{p}$ en valores aleatorios, por ejemplo, en el rango {-1,+1}

Inicializar $\alpha$ en algún valor de $0 < \alpha < 1$;
Repetir:
**for** (Para cada par del conjunto de entrenamiento $(\boldsymbol{x}, t)$, hasta $y = t$)
  $y = f(r = \boldsymbol{p}^T\boldsymbol{x})$; % calcular el valor de $y$.
  **if** $(y \neq t)$
    Actualizar el vector de pesos mediante la ecuación (6.17) como sigue:

$$\boldsymbol{p}' = \boldsymbol{p} + \alpha\,(t - y)\,\boldsymbol{x}$$

    **else**
      No hacer nada;
    **end**
  **end**
**end**

Para ilustrar la operación del algoritmo de entrenamiento del perceptrón aplicado a una UUL, considérense los siguientes tres ejemplos:

**•• Ejemplo 6.5**

Supongamos que se quiere entrenar una UUL con dos entradas para que realice la función lógica denominada "Y". Como pesos iniciales se seleccionan los siguientes: $p_1 = 0.25,\ p_2 = 0.25$ y $\theta = 1.0$. Además, sea $\alpha = 0.25$.

**Solución**

Para encontrar los pesos de la recta de separación entre las dos posibles clases, se recorrerán los pares de entrenamiento como sigue: $\boldsymbol{x}_1 = (0,0)^T,\ t_1 = 0$, luego para $\boldsymbol{x}_2 = (0,1)^T,\ t_2 = 0$ y $\boldsymbol{x}_3 = (1,0)^T, t_3 = 0$; finalmente, $\boldsymbol{x}_4 = (1,1)^T, t_4 = 1$.

Para cada uno de estos pares se aplicará el algoritmo en forma iterada hasta obtener la solución deseada. De acuerdo con lo visto, los cuatro vectores aumentados son los siguientes: $\boldsymbol{x}_1 = (0,0,-1)^T$, $\boldsymbol{x}_2 = (0,1,-1)^T$, $\boldsymbol{x}_3 = (1,0,-1)^T$, y $\boldsymbol{x}_4 = (1,1,-1)^T$.

Debido a que el problema es linealmente separable, el teorema de convergencia del perceptrón garantiza que se podrá encontrar la UUL buscada. Para facilitar la explicación se acomodarán los resultados iteración a iteración.

**Primera época**: $(p_1 = 0.25,\ p_2 = 0.25,\ p_3 = \theta = 1.00,\ r = p_1\ x_1 + p_2\ x_2 - \theta,\ \triangle p_i = \alpha(t-y)x_i)$. La secuencia de datos se indica en la tabla 6.1:

**Tabla 6.1**: Secuencia de datos para la primera época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.25 | 0.25 | 1.00 | -1.00 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0 | 1 | 0.25 | 0.25 | 1.00 | -0.75 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 0 | 0.25 | 0.25 | 1.00 | -0.75 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 1 | 0.25 | 0.25 | 1.00 | -0.50 | 1 | 0 | 0.25 | 0.25 | 0.25 | -0.25 |

**Segunda época**: $(p_1 = 0.25 + 0.25 = 0.50,\ p_2 = 0.25 + 0.25 = 0.50,\ p_3 = 1.00 - 0.25 = 0.75)$; sus correspondientes datos se muestran en la tabla 6.2.

**Tabla 6.2**: Secuencia de datos para la segunda época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.50 | 0.50 | 0.75 | -0.75 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0 | 1 | 0.50 | 0.50 | 0.75 | -0.25 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 0 | 0.50 | 0.50 | 0.75 | -0.25 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 1 | 0.50 | 0.50 | 0.75 | 0.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

De la tabla 6.2 se puede ver que se ha logrado la convergencia. Luego entonces, los parámetros de la UUL que resuelven el problema quedan como sigue: $p_1 = 0.50$, $p_2 = 0.50$ y $\theta = 0.75$. De acuerdo con lo visto (ecuación 6.6), la ecuación de la frontera de separación entre clases viene dada como: $x_2 = -(p_1/p_2)x_1 + (\theta/p_2) = -(0.5/0.5)x_1 + (0.75/0.5) = -x_1 + 1.5$.

En la figura 6.14 se muestra esta línea de decisión, la cual, como se puede ver, logra separar las dos clases de puntos y, por tanto, implementar la función "Y" lógica.

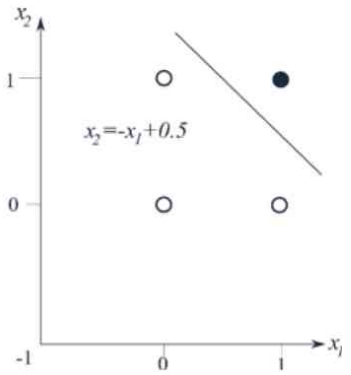



**Figura 6.14**: Conjunto de datos para el ejemplo 6.5.

**•• Ejemplo 6.6**

Supongamos que se desea entrenar una UUL de dos entradas para realizar la función lógica del "O". Como pesos iniciales se seleccionan, ahora, los siguientes: $p_1 = 0.5,\ p_2 = 0.5$ y $\theta = -0.5$. Además sea $\alpha = 0.25$.

**Solución**

De nuevo, para encontrar los pesos de la recta de separación entre las dos posibles clases, se recorrerán los pares de entrenamiento como sigue: $\boldsymbol{x}_1 = (0,\ 0)^T$, $t_1 = 0$, luego $\boldsymbol{x}_2 = (0,\ 1)^T$, $t_2 = 1$, luego $\boldsymbol{x}_3 = (1,\ 0)^T$, $t_3 = 1$, y luego $\boldsymbol{x}_4 = (1,\ 1)^T$, $t_4 = 1$.

Para cada uno de estos pares se aplicará el algoritmo en forma iterada hasta obtener la solución deseada. Los vectores aumentados son los siguientes: $\boldsymbol{x}_1 = (0,\ 0,\ -1)^T$, $\boldsymbol{x}_2 = (0,\ 1,\ -1)^T$, $\boldsymbol{x}_3 = (1,\ 0,\ -1)^T$, y $\boldsymbol{x}_4 = (1,\ 1,\ -1)^T$.

Se puede ver que el problema es linealmente separable. De acuerdo con lo visto, el teorema de convergencia del perceptrón garantiza que se podrá encontrar la UUL buscada. Para facilitar la explicación, una vez más, se acomodarán los resultados iteración a iteración en tablas numéricas como sigue:

Las primeras iteraciones de la primera época se encuentran contenidas en la tabla 6.3.

**Primera época** $(p_1 = 0.50,\ p_2 = 0.50,\ p_3 = \theta = -0.50,\ r = p_1x_1 + p_2x_2 - \theta,\ \triangle p_i = \alpha(t - y)x_i)$.

**Tabla 6.3**: Secuencia numérica de datos para la primera época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.50 | 0.50 | -0.50 | 0.50 | 0 | 1 | -0.25 | 0.00 | 0.00 | 0.25 |
| 0 | 1 | 0.50 | 0.50 | -0.25 | 0.75 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 0 | 0.50 | 0.50 | -0.25 | 0.75 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 1 | 0.50 | 0.50 | -0.25 | 1.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

**Tabla 6.4**: Secuencia numérica de datos para la segunda época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.50 | 0.50 | -0.25 | 0.25 | 0 | 1 | -0.25 | 0.00 | 0.00 | 0.25 |
| 0 | 1 | 0.50 | 0.50 | 0.00 | 0.50 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 0 | 0.50 | 0.50 | 0.00 | 0.50 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 1 | 0.50 | 0.50 | 0.00 | 1.00 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

Las 4 iteraciones siguientes corresponden a la **segunda época** (ver tabla 6.4) con los parámetros ($p_1 = 0.50,\ p_2 = 0.50,\ p_3 = \theta = -0.25$).

Los resultados de la **tercera época** ($p_1 = 0.50,\ p_2 = 0.50,\ p_3 = \theta = 0.00$) se encuentran en la tabla 6.5.

**Tabla 6.5**: Secuencia numérica de datos para la tercera época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.50 | 0.50 | 0.00 | 0.00 | 0 | 1 | -0.25 | 0.00 | 0.00 | 0.25 |
| 0 | 1 | 0.50 | 0.50 | 0.25 | 0.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 0 | 0.50 | 0.50 | 0.25 | 0.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 1 | 0.50 | 0.50 | 0.25 | 0.75 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

La **cuarta época** compuesta por ($p_1 = 0.50,\ p_2 = 0.50,\ p_3 = \theta = 0.25$), cuyos resultados se presentan en la tabla 6.6.

**Tabla 6.6**: Secuencia numérica de datos para la cuarta época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.50 | 0.50 | 0.25 | -0.25 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0 | 1 | 0.50 | 0.50 | 0.25 | 0.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 0 | 0.50 | 0.50 | 0.25 | 0.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1 | 1 | 0.50 | 0.50 | 0.25 | 0.75 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

De esta última tabla 6.6, se puede ver que se ha logrado la convergencia. Luego entonces, los parámetros de la UUL que resuelven el problema quedan como sigue: $p_1 = 0.50,\ p_2 = 0.50$ y $\theta = 0.25$.

De acuerdo con lo visto, (ecuación 6.6), la ecuación de la frontera de separación entre clases viene dada como:

$$x_2 = -\frac{p_1}{p_2}x_1 + \frac{\theta}{p_2} = -\frac{0.5}{0.5}x_1 + \frac{0.25}{0.5} = -x_1 + 0.5.$$

En la figura 6.15 se muestra esta línea de decisión, la cual, como se puede ver, logra separar las dos clases de puntos y, por tanto, implementar la función "O" lógica.

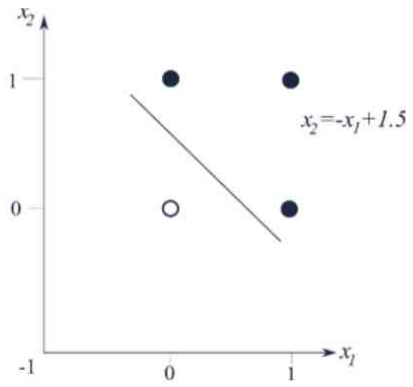



**Figura 6.15**: Conjunto de datos para el ejemplo 6.6.

**• • • Ejemplo 6.7**

Supongamos que se quiere entrenar una UUL de dos entradas para encontrar la línea de separación en las dos clases de puntos mostrados en la figura 6.16a. Como pesos iniciales se seleccionan los siguientes: $p_1 = 0.5,\ p_2 = 0.5$ y $\theta = 0.5$. Además, $\alpha = 0.25$.

**Solución**

Para encontrar los pesos de la recta de separación entre las dos clases, se recorrerán los pares de entrenamiento como sigue: $\boldsymbol{x}_1 = (4,\ 2)^T$, $t_1 = 0$, luego $\boldsymbol{x}_2 = (5,\ 4)^T$, $t_2 = 0$, luego $\boldsymbol{x}_3 = (7,\ 5)^T$, $t_3 = 0$, y luego $\boldsymbol{x}_4 = (9,\ 4)^T$, $t_4 = 0$, $\boldsymbol{x}_5 = (3,\ 6)^T$, $t_5 = 1$, luego $\boldsymbol{x}_6 = (4,\ 8)^T$, $t_6 = 1$, luego $\boldsymbol{x}_7 = (6,\ 7)^T$, $t_7 = 1$, y luego $\boldsymbol{x}_8 = (8,\ 8)^T$, $t_8 = 1$. Para cada uno de estos pares se aplicará el algoritmo en forma iterada hasta obtener la solución deseada.

Los vectores aumentados son los siguientes:
$\boldsymbol{x}_1 = (4,\ 2,\ -1)^T$, $\boldsymbol{x}_2 = (5,\ 4,1)^T$, $\boldsymbol{x}_3 = (7,\ 5,\ -1)^T$, $\boldsymbol{x}_4 = (9,\ 4,\ -1)^T$, $\boldsymbol{x}_5 = (3,\ 6,\ -1)^T$, $\boldsymbol{x}_6 = (4,\ 8,1)^T$, $\boldsymbol{x}_7 = (6,\ 7,\ -1)^T$, y $\boldsymbol{x}_8 = (8,\ 8,\ -1)^T$.

Al observar la figura 6.16a, se puede ver que el problema es, de nuevo, linealmente separable. El teorema de convergencia del perceptrón garantiza que se podrá encontrar la UUL buscada. Para facilitar la explicación, una vez más, se acomodarán los resultados iteración a iteración en una tabla como sigue:

La secuencia de datos para la **primera época** ($p_1 = 0.50,\ p_2 = 0.50,\ p_3 = \theta = 0.50,\ r = p_1x_1 + p_2x_2 - \theta$, $\triangle p_i = \alpha(t - y)x_i$), se encuentra en la tabla 6.7.

Los resultados correspondientes para la **segunda época** ($p_1 = 0.25,\ p_2 = 1.50,\ p_3 = \theta = 0.50$) se encuentran indicados en la tabla 6.8.

La tabla 6.9 presenta los resultados obtenidos para la **tercera época** ($p_1 = -0.75$, $p_2 = 1.00,\ p_3 = \theta = 0.75$).

De la tabla 6.9 se puede ver que una vez más se ha logrado la convergencia. Luego entonces, los parámetros de la UUL que resuelven el problema quedan como a continuación se indica: $p_1 = -0.75, p_2 = 1.00$ y $\theta = 0.75$.

De acuerdo con lo visto, la ecuación de la frontera de separación entre clases viene dada

**Tabla 6.7**: Iteración de datos para la primera época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 0.50 | 0.50 | 0.50 | 2.50 | 0 | 1 | -0.25 | -1.00 | -0.50 | 0.25 |
| 5 | 4 | -0.50 | 0.00 | 0.75 | -3.25 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 7 | 5 | -0.50 | 0.00 | 0.75 | -4.25 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 9 | 4 | -0.50 | 0.00 | 0.75 | -5.25 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 6 | -0.50 | 0.00 | 0.75 | -2.25 | 1 | 0 | 0.25 | 0.75 | 1.50 | 0.25 |
| 4 | 8 | 0.25 | 1.50 | 0.50 | 12.5 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 6 | 7 | 0.25 | 1.50 | 0.50 | 11.5 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 8 | 8 | 0.25 | 1.50 | 0.50 | 13.5 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

**Tabla 6.8**: Resultados para la segunda época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 0.25 | 1.50 | 0.50 | 3.50 | 0 | 1 | -0.25 | -1.00 | -0.50 | 0.25 |
| 5 | 4 | -0.75 | 1.00 | 0.75 | -0.50 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 7 | 5 | -0.75 | 1.00 | 0.75 | -1.00 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 9 | 4 | -0.75 | 1.00 | 0.75 | -3.50 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 6 | -0.75 | 1.00 | 0.75 | 3.00 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 8 | -0.75 | 1.00 | 0.75 | 4.50 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 6 | 7 | -0.75 | 1.00 | 0.75 | 1.75 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 8 | 8 | -0.75 | 1.00 | 0.75 | 1.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

como:

$$x_2 = -\frac{p_1}{p_2}x_1 + \frac{\theta}{p_2} = -\frac{-0.75}{1.0}x_1 + \frac{0.75}{1.0} = 0.75x_1 + 0.75.$$

En la figura 6.16b se muestra esta línea de decisión, la cual, como se puede ver, logra separar las dos clases de puntos.

**Tabla 6.9**: Resultados para la tercera época.

| $x_1$ | $x_2$ | $p_1$ | $p_2$ | $\theta$ | $r$ | $t$ | $y$ | $\alpha(t-y)$ | $\triangle p_1$ | $\triangle p_2$ | $\triangle p_3 = \triangle\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | -0.75 | 1.00 | 0.75 | -1.75 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 4 | -0.75 | 1.00 | 0.75 | -0.50 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 7 | 5 | -0.75 | 1.00 | 0.75 | -1.00 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 9 | 4 | -0.75 | 1.00 | 0.75 | -3.50 | 0 | 0 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 6 | -0.75 | 1.00 | 0.75 | 3.00 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 8 | -0.75 | 1.00 | 0.75 | 4.50 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 6 | 7 | -0.75 | 1.00 | 0.75 | 1.75 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |
| 8 | 8 | -0.75 | 1.00 | 0.75 | 1.25 | 1 | 1 | 0.00 | 0.00 | 0.00 | 0.00 |

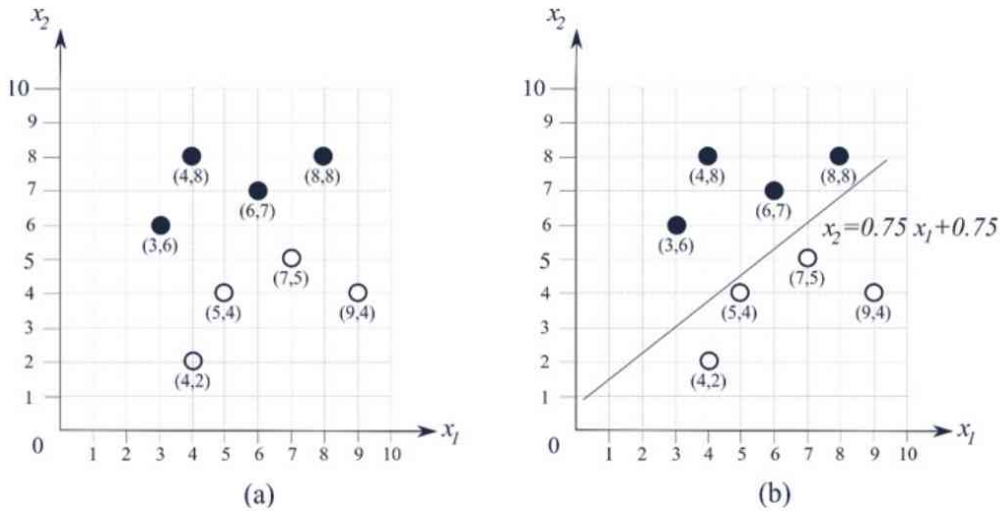



**Figura 6.16**: Conjunto de datos para el ejemplo 6.7.

De estos tres ejemplos se puede ver que una UUL puede utilizarse para implementar funciones booleanas y también resolver problemas de clasificación linealmente separables.

### 6.4.10 Robustez de la UUL ante ruido en las entradas

Enseguida se mostrará cómo una UUL puede emplearse para clasificar versiones ruidosas de los patrones usados para su diseño. Para esto considérese la red diseñada en el ejemplo 6.3.

**Ejemplo 6.8**

Considere las siguientes dos versiones con ruido de los patrones: $\boldsymbol{x} = (0.5,\ 0.6)^T$ y $\boldsymbol{x} = (0.9,\ 0.9)^T$. La primera pertenece a la clase uno, mientras que la segunda corresponde a la clase dos.

Para el patrón: $\boldsymbol{x} = (0.5,\ 0.6)^T$ : $r = (0.5)0.5 + (0.5)0.6 - 1\ (0.75) = 0.55 - 0.75 = -0.25$, $y = 0$. Por tanto el patrón $\boldsymbol{x} = (0.5,\ 0.6)^T$ es asignado a la clase número uno. Para el patrón: $\boldsymbol{x} = (0.9,\ 0.9)^T$ : $r = 0.5(0.9) + 0.5(0.9) - 1(0.75) = 0.9 - 0.75 = 0.15$, $y = 1$. Por tanto el patrón $\boldsymbol{x} = (0.9,\ 0.9)^T$ es asignado a la clase número dos.

## 6.5 El perceptrón

En esta sección revisaremos, primeramente, los principios básicos del perceptrón como lo concibió originalmente Rosenblatt en 1962 [46]. Enseguida, se estudiará la versión más usada por la comunidad.

### 6.5.1 El perceptrón de Rosenblatt

El perceptrón original de Rosenblatt consta de tres capas [46]: Una de unidades de sensado (unidades tipo S), una de asociación (unidades tipo A) y una de respuesta (unidades tipo R). La retina del perceptrón es un arreglo de elementos de sensado (fotoceldas). Una unidad tipo S emite un "1" si es excitada, un "0" si no.

En el modelo original, un número de unidades de retina (retinales) seleccionado al azar era conectado a la capa de asociación (unidades tipo $A$). También existían muchas conexiones entre las unidades tipo $A$ y las tipo $R$, incluso de retroalimentación entre las del tipo R y las del tipo $A$. Esto, por supuesto, pudiera introducir dinámicas no deseadas en la operación del perceptrón. Una versión simplificada sin conexiones laterales y conexiones de retroalimentación se muestra en la figura 6.17.

De acuerdo con esta figura, cada unidad tipo $A$ funciona como sigue. La unidad recibe a su entrada $n$ valores del tipo $x_k$, generalmente al azar, en el ejemplo mostrado en la figura $n = 3$.

La unidad $A$ calcula entonces una suma ponderada sobre los valores $x_k$:

$$\alpha_i = \sum_{k=1}^{n} \beta_k x_k \tag{6.20}$$

Los pesos $\beta_k$ pueden tomar valores +1 o -1 , y son asignados en forma aleatoria.

La suma $\sum_{k=1}^{n} \beta_k x_k$ es comparada con un umbral $u$. La salida $a_i$ de la $i$-ésima unidad $A$ viene dada por la siguiente ecuación:

$$\alpha_i = \begin{cases} 1 & \text{si} \quad \sum_{k=1}^{n} \beta_k x_k \geq u \\ 0 & \text{si} \quad \sum_{k=1}^{n} \beta_k x_k < u \end{cases} \tag{6.21}$$

**Comentario**. El umbral para todas las unidades tipo $A$ es el mismo.

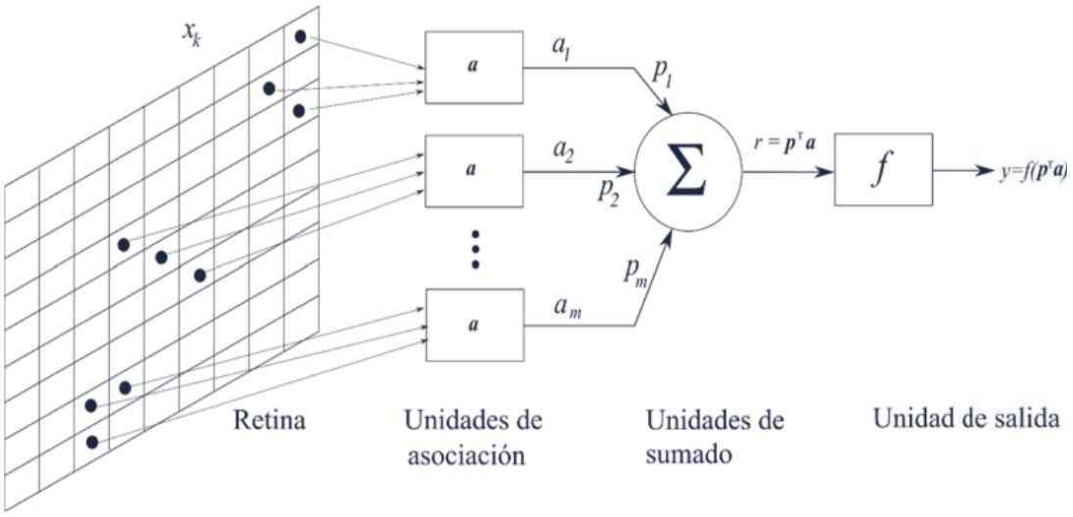



**Figura 6.17**: Versión simplificada del perceptrón original.

La salida binaria de la $i$-ésima unidad $A$, con $i = 1, 2, \cdots, m$ es multiplicada por un peso $p_i$, y una suma ponderada de todas las $m$ salidas ponderadas es obtenida por la unidad de suma. Cada peso $p_i$ puede tomar ser negativo, positivo o cero. La salida de la neurona es binaria: {0, 1} o {-1, 1}, dependiendo de un umbral $\theta$, que normalmente es puesto en cero:

$$y = \sum_{i=1}^{m} p_i s_i. \tag{6.22}$$

Al igual que la simple UUL, el diseño del perceptrón involucra el ajuste de los pesos $p_i$ y la selección del umbral $u$.

De acuerdo con lo visto, se puede pensar de la UUL como un caso especial del perceptrón.

### 6.5.2 El perceptrón estándar

La estructura del perceptrón estándar, normalmente usado en la literatura actual, se muestra en la figura 6.18; para el cual la salida toma la forma de límite simétrico (figura 6.18a, sin bias y 6.18b, con bias). Al igual que la UUL, el perceptrón permite separar conjuntos de patrones linealmente separables en dos clases: $C^1$ y $C^2$.

De acuerdo con esta figura, la respuesta del perceptrón puede ser expresada como sigue:

$$y = f(r) \tag{6.23}$$

donde, como ya vimos:

$$r = \sum_{i=1}^{n} p_i x_i = \boldsymbol{p}^T \boldsymbol{x}, \tag{6.24}$$

con $\boldsymbol{p} = [p_1,\ p_2,\ \cdots,\ p_n]^T$ y $\boldsymbol{x} = [x_1,\ x_2,\ \cdots,\ x_n]^T$:

$$r = \sum_{i=1}^{n+1} p_i x_i = \boldsymbol{p}^{'T} \boldsymbol{x}^{'}, \tag{6.25}$$

con $\boldsymbol{p}^{'} = [p_1,\ p_2,\ \cdots,\ p_n,\ p_{n+1} = \theta]^T$ y $\boldsymbol{x}^{'} = [x_1,\ x_2,\ \cdots,\ x_n,\ x_{n+1} = -1]^T$.

De la misma figura se puede ver que cuando:

1) $r \geq \theta$ para el caso de la ecuación (6.24) o $r \geq 0$ para el caso de la ecuación (6.25), la función de activación del perceptrón ocasiona que la salida $y = +1$, indicando que el patrón de entrada $\boldsymbol{x}$ debe ser clasificado en la clase $C^2$.

2) $r < \theta$ para el caso de la ecuación (6.24) o $r < 0$ para el caso de la ecuación (6.25), la función de activación del perceptrón ocasiona que la salida $y = -1$, indicando que el patrón de entrada $\boldsymbol{x}$ debe ser clasificado en la clase $C^1$.

De acuerdo con lo visto, se puede pensar de la UUL como un caso especial del perceptrón.

### 6.5.2 El perceptrón estándar

La estructura del perceptrón estándar, normalmente usado en la literatura actual, se muestra en la figura 6.18; para el cual la salida toma la forma de límite simétrico (figura 6.18a, sin bias y 6.18b, con bias). Al igual que la UUL, el perceptrón permite separar conjuntos de patrones linealmente separables en dos clases: $C^1$ y $C^2$.

De acuerdo con esta figura, la respuesta del perceptrón puede ser expresada como sigue:

$$y = f(r) \tag{6.23}$$

donde, como ya vimos:

$$r = \sum_{i=1}^{n} p_i x_i = \boldsymbol{p}^T \boldsymbol{x}, \tag{6.24}$$

con $\boldsymbol{p} = [p_1,\ p_2,\ \cdots,\ p_n]^T$ y $\boldsymbol{x} = [x_1,\ x_2,\ \cdots,\ x_n]^T$:

$$r = \sum_{i=1}^{n+1} p_i x_i = \boldsymbol{p}^{'T} \boldsymbol{x}^{'}, \tag{6.25}$$

con $\boldsymbol{p}^{'} = [p_1,\ p_2,\ \cdots,\ p_n,\ p_{n+1} = \theta]^T$ y $\boldsymbol{x}^{'} = [x_1,\ x_2,\ \cdots,\ x_n,\ x_{n+1} = -1]^T$.

De la misma figura se puede ver que cuando:

1) $r \geq \theta$ para el caso de la ecuación (6.24) o $r \geq 0$ para el caso de la ecuación (6.25), la función de activación del perceptrón ocasiona que la salida $y = +1$, indicando que el patrón de entrada $\boldsymbol{x}$ debe ser clasificado en la clase $C^2$.

2) $r < \theta$ para el caso de la ecuación (6.24) o $r < 0$ para el caso de la ecuación (6.25), la función de activación del perceptrón ocasiona que la salida $y = -1$, indicando que el patrón de entrada $\boldsymbol{x}$ debe ser clasificado en la clase $C^1$.

De acuerdo con lo visto, se puede pensar de la UUL como un caso especial del perceptrón.

### 6.5.2 El perceptrón estándar

La estructura del perceptrón estándar, normalmente usado en la literatura actual, se muestra en la figura 6.18; para el cual la salida toma la forma de límite simétrico (figura 6.18a, sin bias y 6.18b, con bias). Al igual que la UUL, el perceptrón permite separar conjuntos de patrones linealmente separables en dos clases: $C^1$ y $C^2$.

De acuerdo con esta figura, la respuesta del perceptrón puede ser expresada como sigue:

$$y = f(r) \tag{6.23}$$

donde, como ya vimos:

$$r = \sum_{i=1}^{n} p_i x_i = \boldsymbol{p}^T \boldsymbol{x}, \tag{6.24}$$

con $\boldsymbol{p} = [p_1,\ p_2,\ \cdots,\ p_n]^T$ y $\boldsymbol{x} = [x_1,\ x_2,\ \cdots,\ x_n]^T$:

$$r = \sum_{i=1}^{n+1} p_i x_i = \boldsymbol{p}^{'T} \boldsymbol{x}^{'}, \tag{6.25}$$

con $\boldsymbol{p}^{'} = [p_1,\ p_2,\ \cdots,\ p_n,\ p_{n+1} = \theta]^T$ y $\boldsymbol{x}^{'} = [x_1,\ x_2,\ \cdots,\ x_n,\ x_{n+1} = -1]^T$.

De la misma figura se puede ver que cuando:

1) $r \geq \theta$ para el caso de la ecuación (6.24) o $r \geq 0$ para el caso de la ecuación (6.25), la función de activación del perceptrón ocasiona que la salida $y = +1$, indicando que el patrón de entrada $\boldsymbol{x}$ debe ser clasificado en la clase $C^2$.

2) $r < \theta$ para el caso de la ecuación (6.24) o $r < 0$ para el caso de la ecuación (6.25), la función de activación del perceptrón ocasiona que la salida $y = -1$, indicando que el patrón de entrada $\boldsymbol{x}$ debe ser clasificado en la clase $C^1$.

Igual a la UUL, un perceptrón como el que se muestra en la figura 6.18, se puede usar en la clasificación de patrones, sus pesos deben ser ajustados. Se puede emplear la misma regla de entrenamiento utilizada para el ajuste de parámetros libres de la UUL (subsección 6.4.8).

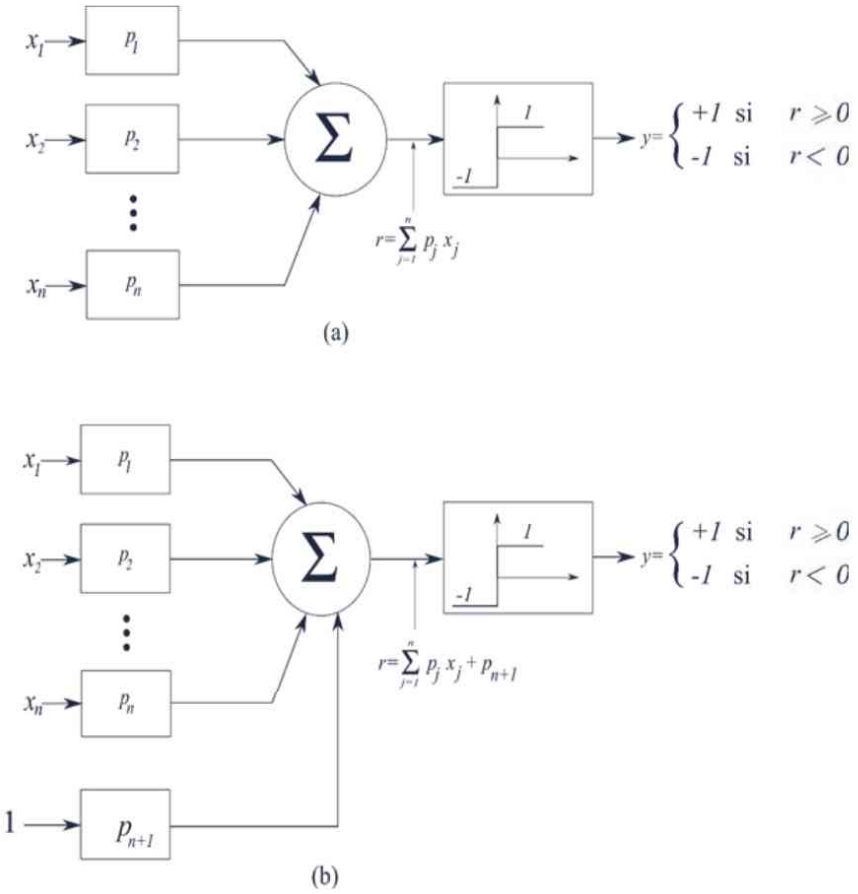



**Figura 6.18**: Representación esquemática del perceptrón para dos clases. (a) sin bias o desplazamiento, y (b) con bias.

De todo esto, se puede ver por medio de una UUL o resolver a través de un perceptrón, en principio, cualquier problema que involucre la separación de un conjunto de patrones en dos clases. Sin embargo, como ya se dijo al principio de la subsección 6.4.1, si el conjunto de patrones no presenta separación lineal, entonces el perceptrón, al igual que la UUL, no podrá separarlos.

El ejemplo más sencillo que se puede encontrar es el mostrado en la figura 6.3, donde como ya se mencionó, se trata del conocido problema del "o" exclusivo. Más adelante se estudiará cómo resolver este problema mediante la combinación de perceptrones.

# 6.6 La ADALINE y la regla Delta

En esta sección revisaremos los pormenores de la neurona lineal tipo ADALINE, así como de la regla Delta, especialmente diseñada para el entrenamiento de este tipo de neuronas. Veremos cómo estas máquinas pueden usarse en la solución de problemas de clasificación lineal y no lineal de patrones.

## 6.6.1 La neurona tipo ADALINE

La ADALINE (del inglés ADAptive LINEar Element) es un tipo de red neuronal basado en el modelo de neurona de McCulloch y Pitts. Fue desarrollada por los científicos Bernard Widrow y Ted Hoff en 1960. Usualmente se compone de una sola capa de neuronas, con $n$ entradas y $p$ salidas. Las $n$ entradas son los componentes de un vector $\boldsymbol{x}$ con $n$ componentes $\boldsymbol{x} = [\,(x_1 \quad x_2 \quad \cdots \quad x_n\,]^T$.

Cada una de las $p$ neuronas, tal y como se muestra en la figura 6.19, recibe el vector $\boldsymbol{x}$ como entrada y realiza un producto punto desplazado por el parámetro $\theta$, para dar como salida el valor:

$$y_j \quad = \quad r_j = \sum_{i=1}^{n} (\, p_{ji} x_i + \theta_j \,) \,. \tag{6.26}$$

Note que a diferencia de la UUL o el perceptrón, la ADALINE usa como función de activación una función lineal pura. Esto, como veremos enseguida, permitirá el uso de derivadas para su entrenamiento.

A diferencia de la UUL o el perceptrón, el entrenamiento de la ADALINE toma en cuenta el error entre la salida estimada respecto a la deseada. Esto se logra mediante la aplicación de la regla Delta, la cual se explicará.

**Comentario**. Cuando las redes neuronales tipo ADALINE son utilizadas como clasificadores, en lugar de etiquetas 0 y 1, como se hace con las UUL, se usan ahora etiquetas -1 y +1.

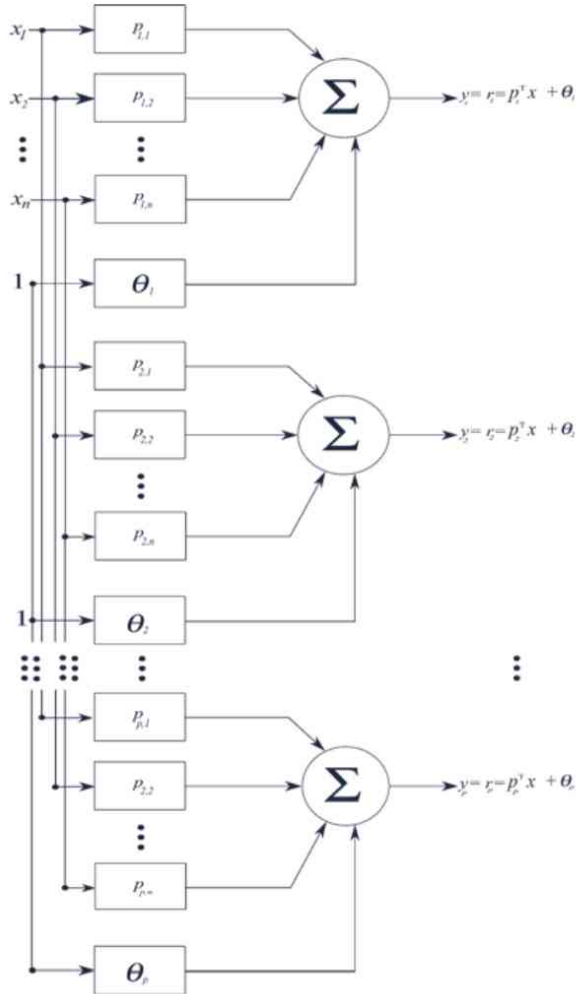


**Figura 6.19**: Red neuronal tipo ADALINE de $n$ entradas y $p$ salidas.

### 6.6.2 La regla Delta

Enseguida se verán los pormenores para la derivación de una segunda regla para el ajuste de los pesos de una RNA tipo ADALINE. Se trata de la conocida regla Delta. Primero se estudiará cómo encontrar el mínimo de una función mediante la técnica del descenso del gradiente.

Sin pérdida de generalidad, considérese el caso de una variable y que depende de una variable única $x$, esto es $y = f(x)$. Supóngase que se desea encontrar el valor $x_0$ para el cual: $y(x_0) \leq y(x)$, y esto para toda $x$. Sea $x^*$ el mejor estimado actualizado de $x_0$.

Algo que se puede hacer para obtener un buen estimado consiste en modificar el valor de $x$ poco a poco tratando de seguir la cuesta hacia abajo de la función. Como referencia a la figura 6.20a se tienen dos casos:

**Caso 1.** Al aumentar $x$ empezando en $x^*$, provoca un decremento en $y$. Esto causa un cambio positivo $\triangle x > 0$ al estimado de $x^*$.

**Caso 2.** Al disminuir $x$ resulta también en un decremento en $y$. Esto causa un cambio negativo $\triangle x < 0$.

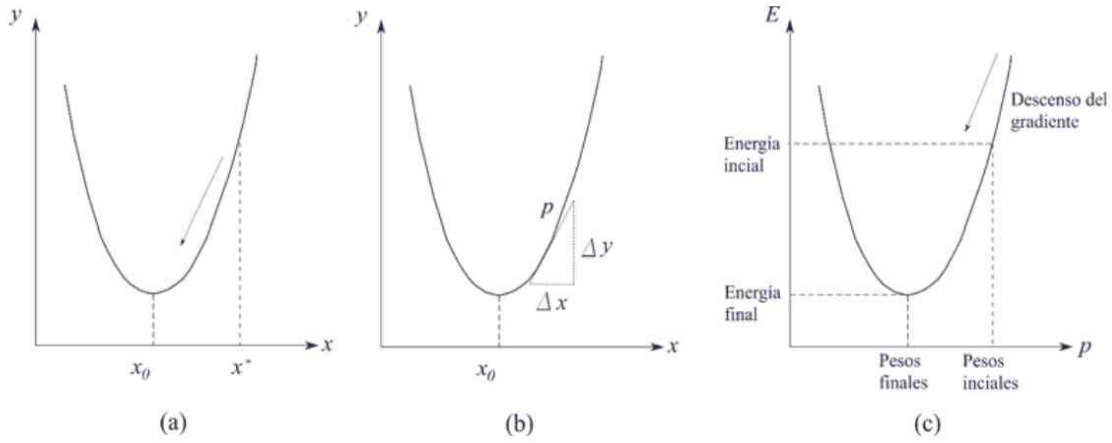



**Figura 6.20**: (a) Ejemplificación de la obtención del mínimo de una función. (b) Ejemplo del cálculo aproximado de la pendiente de una función en un punto $p$. (c) Ejemplificación del ajuste de los pesos mediante el descenso del gradiente.

**Comentario**. Es bien sabido que este conocimiento queda almacenado en la pendiente de la función en el punto $x^*$. Esto es, si $x$ crece y crece $y$, por tanto, la pendiente es positiva, si no es negativa. Por otro lado, la pendiente en cualquier punto $p$ viene dada por la tangente a dicho punto sobre la curva. Una manera de visualizar el cálculo de la pendiente es la siguiente: De acuerdo con la figura 6.20b se dibuja la tangente sobre la función que pasa por el punto $p$. Enseguida, se completa el triángulo y se miden los lados $\triangle x$ y $\triangle y$ (figura 6.20b). Si se parte del hecho de que los cambios en $\triangle x$ y $\triangle y$ son muy pequeños, entonces la derivada $dy$ es aproximadamente igual a $\triangle y$, esto es $dy \approx \triangle y$.

Esto es lo mismo que:

$$\triangle y = \frac{\triangle y}{\triangle x} \triangle x. \tag{6.27}$$

Ya que $\frac{\triangle y}{\triangle x}$ es precisamente la pendiente:

$$\delta y = \text{pendiente } \triangle x. \tag{6.28}$$

Como es bien sabido:

$$\delta y \approx \frac{dy}{dx} \triangle x \tag{6.29}$$

con $\frac{dy}{dx}$ la diferencial o derivada de $y$ con respecto a $x$.

Si se supone que $\triangle x$ puede obtenerse mediante un pequeño cambio en la pendiente a través de una pequeña constante $\alpha > 0$, con el objetivo de asegurar que $dy \approx \triangle y$, luego entonces:

$$\delta x = -\alpha \frac{dy}{dx}. \tag{6.30}$$

Al sustituir esta ecuación en la ecuación (6.28), se tiene que:

$$\delta y \approx -\alpha \left(\frac{dy}{dx}\right)^2. \tag{6.31}$$

**Comentario**. El cuadrado en la ecuación (6.32) siempre dará un valor positivo, el valor negativo de $\alpha$ asegura que $dy < 0$. Ambos resultados garantizan un viaje hacia el mínimo de la función como se desea.

Por otro lado, es fácil ver que la repetición iterada de la ecuación (6.32) coadyuvará al acercamiento a $x_0$, esto es, a la solución buscada. A esta técnica se le conoce en la literatura como técnica de descenso del gradiente [24] y [27].

Todo lo dicho se puede extender al caso multivariable; ahora la variable $y$ depende de $n$ variables, esto es:

$$\triangle x_i = -\alpha \frac{\partial y}{\partial x_i}. \tag{6.32}$$

Vamos ahora a analizar cómo lograr que el gradiente descienda en función del error. Sin pérdida de generalidad, considérese una sola neurona tipo ADALINE (figura 6.19). De nuevo, para cada patrón $\boldsymbol{x}$ se tiene una clase deseada $t$.

Al hacer la lectura especializada, se sabe que el comportamiento de la neurona (red) queda caracterizado por completo con el vector aumentado de pesos $\boldsymbol{p} = [\, p_1 \quad p_2 \quad \cdots \quad p_{n+1} \,]^T$, de esta forma cualquier función $E$ que exprese la discrepancia entre la salida deseada $y$ actual puede ser considerada dependiente del vector de pesos, esto es:

$$E \;=\; E(p_1,\, p_2,\, \cdots,\, p_{n+1}). \tag{6.33}$$

De acuerdo con lo visto, como se ilustra en la figura 6.20(c), el vector óptimo puede ser encontrado al minimizar la función $E$ mediante la técnica del descenso del gradiente. Esto implica una modificación sobre cada peso, como sigue:

$$\triangle p_i \;=\; -\alpha \frac{\partial E}{\partial p_i} \tag{6.34}$$

en forma vectorial como:

$$\triangle \boldsymbol{p} \;=\; -\alpha \frac{\partial E}{\partial \boldsymbol{p}}. \tag{6.35}$$

Falta definir un error $e$. Si $n$ es número de patrones del conjunto de entrenamiento, y si $\boldsymbol{x}^k,\ , k = 1, 2, \cdots, n$, es cada uno de estos patrones y si se da una importancia igual al error para cada patrón $\boldsymbol{x}^k$, entonces el error total puede ser expresado como sigue:

$$e \;=\; \frac{1}{n} \sum_{k=1}^{n} e_k. \tag{6.36}$$

Una definición usualmente utilizada para medir el error $e_k$ entre el valor deseado y el calculado es la siguiente:

$$e_k \;=\; \frac{1}{2} \left( t_k - y_k \right)^2 = \frac{1}{2} \left( t_k - r_k \right)^2. \tag{6.37}$$

En el caso de una neurona tipo ADALINE ecuación (6.26): $y = r = \sum_{i=1}^{n} \left( p_i x_i + \theta \right)$. El error total $e$ viene dado al sustituir la ecuación (6.37) en la ecuación (6.36):

$$e \;=\; \frac{1}{n} \sum_{k=1}^{n} e_k = \frac{1}{n} \sum_{k=1}^{n} \left( \frac{1}{2} \left( t_k - r_k \right)^2 \right). \tag{6.38}$$

Con todo esto, se puede derivar una segunda regla para el entrenamiento de una RNA, en este caso una red tipo ADALINE como la mostrada en la figura 6.19. Se trata de la conocida y útil regla Delta. Para el ajuste de cada peso $p_i$, se presenta a la red el patrón $\boldsymbol{x}^k,\ k = 1, 2, \cdots, n$, para obtener $\frac{\partial e_k}{\partial p_i}$. Esto permite una estimación de $\frac{\partial e}{\partial p_i}$. Al usar la ecuación (6.37) y al expresar $r_k$ en términos de los pesos $p_i$ y las entradas $x_{k,i}$, el lector puede demostrar que:

$$\frac{\partial e_k}{\partial p_i} \;=\; -\left( t_k - r_k \right) x_{k,i} = \left( r_k - t_k \right) x_{k,i}. \tag{6.39}$$

Como ya se vio, $x_{k,i}$ representa el $i$-ésimo componente del vector de entrenamiento $\boldsymbol{x}^k = [\, x_{k,1},\ x_{k,2},\ \cdots,\ x_{k,i},\ \cdots,\ x_{k,i+1}\,]^T$. Al usar esta estimación del gradiente, se tiene al sustituir en (6.34):

$$\triangle p_i \quad = \quad -\alpha \sum_k (\, r_k - t_k \,)\, x_{k,i} = -\alpha \sum_k \delta_k x_{k,i}. \tag{6.40}$$

A esta regla se le conoce como regla de Widrow-Hoff o regla Delta, por la diferencia entre la salida deseada $t_k$ y la salida calculada $r_k$. De acuerdo con la literatura, para valores pequeños del parámetro $\alpha$, la regla Delta converge en sus soluciones (ver [48]).

Si comparamos las dos reglas estudiadas hasta el momento, se puede apreciar lo siguiente:

**Caso 1.** Mientras que la regla Delta se vale de la técnica del descenso del gradiente, la regla del perceptrón se basa en la manipulación de superficies de separación.

**Caso 2.** Mientras que la regla Delta usa la activación de la red, la regla del perceptrón compara la salida de la red con la salida deseada. Si el problema es linealmente separable la regla Delta provocará cambios en los pesos (esto es, los valores $\pm 1$ nunca se alcanzarán), mientras que la regla del perceptrón no lo hará.

En forma similar a como se hizo en el caso de la regla del perceptrón, la actualización del vector de pesos viene dada como:

$$p^{\text{nuevo}} \quad = \quad p^{\text{viejo}} + \bigtriangledown \boldsymbol{p}. \tag{6.41}$$

El entrenamiento de una neurona tipo ADALINE se lleva a cabo por la aplicación iterada de la ecuación (6.41). Cerca del valor óptimo de pesos $\boldsymbol{p}^*$, la magnitud del gradiente $|\bigtriangledown e| \approx 0$.

**Comentario**. A diferencia de la regla del perceptrón, la regla Delta siempre convergerá a una solución. Cabe aclarar que si el problema es no linealmente separable, algunos patrones quedarán mal clasificados.

**Pseudocódigo 6.2**

Regla de entrenamiento Delta

Poner los valores del vector de pesos $\boldsymbol{p}$ en valores aleatorios, por ejemplo, en el rango $\{-1,\ +1\}$.
Inicializar $\alpha$ en algún valor entre $0 < \alpha < 1$.

**repeat**
  **for** (Para cada par de entrenamiento $(\boldsymbol{x}^k,\ t_k)$)
    Calcular la salida de la neurona: $y_k = r_k = \boldsymbol{p}^T\boldsymbol{x}^k$;
    Calcular el término Delta: $\delta_k = r_k - t_k$;
    Calcular el gradiente del vector error respecto al vector de pesos $\frac{\partial e_k}{\partial \boldsymbol{p}}$;
    Calcular el cambio en los pesos $\triangle \boldsymbol{p}$ (6.40);
    Obtener el nuevo vector de pesos como: $p^{\text{nuevo}} \longleftarrow p^{\text{viejo}} + \bigtriangledown \boldsymbol{p}$ (6.41);
  **end**
**until** $|\bigtriangledown e| \approx 0$ ;

Para ilustrar la operación de la regla de entrenamiento Delta aplicado a una neurona ADALINE, considérese los siguientes ejemplos.

**•• Ejemplo 6.9**

Entrenar una neurona ADALINE con una entrada par que la neurona emita el valor de salida $y = 0.5$ para una entrada de $x = 0.2$.

**Solución**

En este caso tan sencillo, el lector puede ver que se tiene un solo dato de entrenamiento $(x = 0.2,\ t = 0.5)$. Suponga, además, que se elige como valor inicial para el peso de interconexión $p = 2.0$. También suponga $\alpha = 0.5$.

En la explicación, procedamos iteración a iteración:

**Iteración 1:**

$$y = r = px = 2.0(0.2) = 0.4,$$
$$\delta = r - t = 0.4 - 0.5 = -0.1$$
$$\bigtriangledown e = \delta x = -0.1(0.2) = -0.02$$
$$\triangle p = -\alpha\ \bigtriangledown e = -0.5(-0.02) = 0.01,$$
$$p \longleftarrow 2.0 + 0.01 = 2.01$$
$$|\nabla e| = 0.02.$$

**Iteración 2:**

$$y = r = px = 2.01(0.2) = 0.402$$
$$\delta = r - t = 0.402 - 0.5 = -0.098$$
$$\nabla e = \delta x = -0.098(0.2) = -0.0196$$
$$\triangle p = -\alpha \nabla e = -0.5(-0.0196) = 0.0098$$
$$p \longleftarrow 2.01 + 0.0098 = 2.0198$$
$$|\nabla e| = 0.0196.$$

**Iteración 3:**

$$y = r = px = 2.0198(0.2) = 0.40396$$
$$\delta = r - t = 0.40396 - 0.5 = -0.09604$$
$$\nabla e = \delta x = -0.09604(0.2) = -0.019208$$
$$\triangle p = -\alpha \nabla e = -0.5(-0.019208) = 0.009604$$
$$p \longleftarrow 2.0198 + 0.009604 = 2.029404$$
$$|\nabla e| = 0.019208.$$

Si el lector continúa con este proceso iterado, podrá demostrar que el valor encontrado por la regla Delta es el siguiente: $p^* = 2.5$. La neurona ADALINE buscada queda como:

$$y = r = 2.5x.$$

Al sustituir el valor de $x = 0.2$ en la ecuación anterior, se puede ver lo siguiente: observe que $y = r = 2.5 \times 0.2 = 0.5$, valor que coincide con el deseado en $t = 0.5$.

### • • • Ejemplo 6.10

Suponga que se quiere entrenar una neurona ADALINE de dos entradas para que realice la función lógica del "O". Como pesos iniciales se seleccionan los siguientes: $p_1 = 1.0,\ p_2 = -1.0$ y $\theta = -1.0$. Además, considere $\alpha = 0.25$.

### Solución

Para encontrar los pesos de la recta de separación entre las dos posibles clases, los pares de entrenamiento se recorren en el siguiente orden: $\boldsymbol{x}_1 = (0,\ 0)^T$, $t_1 = -1$, luego $\boldsymbol{x}_2 = (0,\ 1)^T$, $t_2 = 1$, luego $\boldsymbol{x}_3 = (1,\ 0)^T$, $t_3 = 1$, y luego $\boldsymbol{x}_4 = (1,\ 1)^T$, $t_4 = 1$. Para cada uno de estos pares se aplica el algoritmo en forma iterada hasta obtener la solución deseada.

Los vectores aumentados son los siguientes: $\boldsymbol{x}_1 = (0,\ 0,\ -1)^T$, $\boldsymbol{x}_2 = (0,\ 1, -1)^T$, $\boldsymbol{x}_3 = (1,\ 0,\ -1)^T$, y $\boldsymbol{x}_4 = (1,\ 1,\ -1)^T$.

De acuerdo con lo visto, el error cuadrático medio dado por las ecuaciones (6.36) y (6.37) deberá disminuir provocando el posicionamiento, poco a poco, de la línea de separación entre clases. Para facilitar la comprensión de la operación de la regla Delta, se muestran los cálculos para la primera época de recorrido. Estos se acomodan en tablas como sigue:

**Época 1**: $p_1 = 1.0,\ p_2 = -1.0,\ \theta = -1.0$ y $\alpha = 0.25$, $y = p_1x_1 + p_2x_2 + \theta$. Los valores para la salida y de la neurona, su error $\delta$, así como los incrementos en los tres pesos $\triangle p_1$, $\triangle p_2$ y $\triangle\theta$ se muestran en la tabla 6.10:

**Tabla 6.10**: Resultados para la primera época.

| $x_1$ | $x_2$ | $x_3$ | $r$ | $t$ | $\delta = r - t$ | $\triangle p_1 = -0.25\delta x_1$ | $\triangle p_2 = -0.25\delta x_2$ | $\triangle p_3 = -0.25\delta x_3$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | -1 | 1.0 | -1 | 2.0 | 0.0 | 0.0 | 0.5 |
| 0 | 1 | -1 | -0.5 | 1 | -1.5 | 0.0 | 0.375 | -0.375 |
| 1 | 0 | -1 | 1.875 | 1 | 0.875 | -0.21875 | 0.0 | 0.21875 |
| 1 | 1 | -1 | 0.8125 | 1 | -0.1875 | 0.046875 | 0.046875 | -0.046875 |

Ahora, los valores actualizados de los pesos $p_1$, $p_2$ y $\theta$, con respecto a los pesos viejos, se muestran en la tabla 6.11:

**Tabla 6.11**: Valores actualizados de $p_1$, $p_2$ y $\theta$.

| $p_1$ viejo | $p_2$ viejo | $\theta$ viejo | $p_1$ nuevo | $p_2$ nuevo | $\theta$ nuevo |
|---|---|---|---|---|---|
| 1.0 | -1.0 | -1.0 | 1.0 | -1.0 | -1.0+0.5=-0.5 |
| 1.0 | -1.0 | -0.5 | 1.0+0.0=1.0 | -1.0+0.375=-0.625 | -0.5-0.375=-0.875 |
| 1.0 | -0.625 | -0.875 | 1.0-0.21875=0.78125 | -0.625+0.0=-0.625 | -0.875+0.21875=-0.65625 |
| 0.78125 | -0.625 | -0.65625 | 0.78125+0.046875=0.828125 | -0.625+0.046875=-0.578125 | -0.65625-0.046875=-0.703125 |

El valor del error individual debido a cada patrón se muestra en la tabla 6.12:

**Tabla 6.12**: Errores individuales de cada patrón.

| $\delta = r - t$ | $e^k = \frac{1}{2}(r_k - t_k)^2$ |
|---|---|
| 2.0 | 2.0 |
| -1.5 | 1.125 |
| 0.875 | 0.3828125 |
| -0.1875 | 0.017578125 |

Finalmente, el error $e$ acumulado al final de la primera época se calcula como sigue:

$$e_1 \quad = \frac{1}{n}\sum_{k=1}^{n} e^k = \frac{1}{4}(2.0 + 1.125 + 0.3828125 + 0.017578125) = 0.881347657.$$

El lector puede demostrar que los valores de este error a lo largo de las 10 primeras iteraciones son las que se indican en la tabla 6.13:

**Tabla 6.13**: Valores de las primeras 10 iteraciones del error $e_1$.

| Época $i$ | $e_i$ |
|---|---|
| 1 | 0.88134766 |
| 2 | 0.59744397 |
| 3 | 0.44203641 |
| 4 | 0.35351476 |
| 5 | 0.30221814 |
| 6 | 0.27200475 |
| 7 | 0.25390052 |
| 8 | 0.24285272 |
| 9 | 0.23598038 |
| 10 | 0.23161877 |

De igual manera, el lector puede mostrar que los valores finales de los pesos $p_1$, $p_2$ y $\theta$, después de la ejecución de la décima época son los resultados que se presentan en la tabla 6.14:

**Tabla 6.14**: Valores finales de los pesos $p_1$, $p_2$ y $\theta$.

| $p_1$ | $p_2$ | $\theta$ |
|---|---|---|
| 0.616 | 0.691 | 0.171 |

La figura 6.21a muestra la neurona correspondiente que como se puede ver tiene tres entradas, tres pesos, incluyendo en bias y una salida correspondiente a la activación de la neurona.

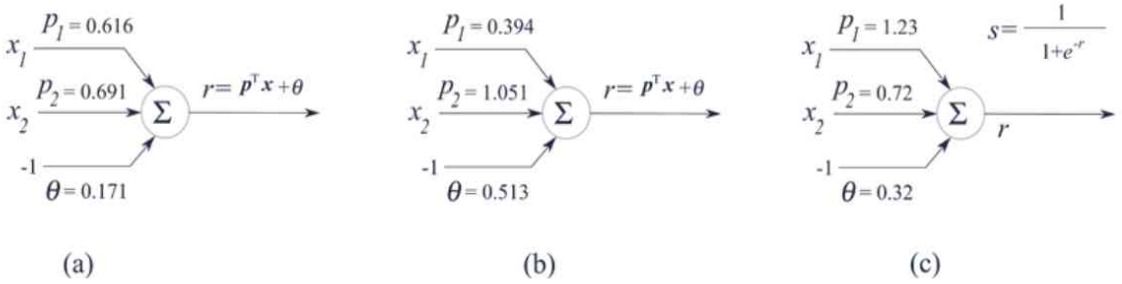



**Figura 6.21**: (a) Neurona resultante del problema 6.7. (b) Neurona resultante del problema 6.8. (c) Neurona del problema 6.9.

En la figura 6.22a se muestra la gráfica del error $e$, mientras que en la figura 6.22b se muestra la línea de separación entre las dos clases de puntos que realizan la función "O" lógica. De acuerdo con la ecuación 6.19, la línea de separación entre ambas clases viene dada como $x_2 = -0.8909x_1 + 0.2473$.

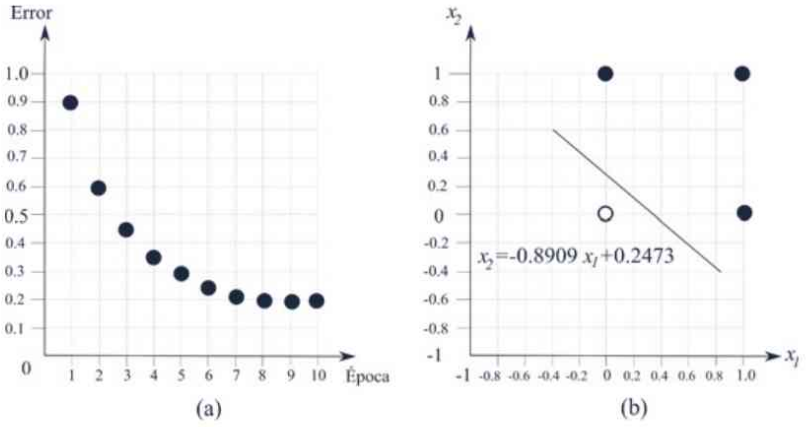



**Figura 6.22**: (a) Gráfica del error $e$. (b) Línea de separación dada por $x_2 = -0.8909x_1 + 0.2473$ entre las dos clases.

Para verificar el correcto funcionamiento de la neurona resultante con pesos $p_1 = 0.616$, $p_2 = 0.691$ y $\theta = 0.171$, clasifiquemos los 4 puntos: $\boldsymbol{x}_1 = (00)^T$, $\boldsymbol{x}_2 = (01)^T$, $\boldsymbol{x}_3 = (1,\ 0)^T$, y $\boldsymbol{x}_4 = (1,\ 1)^T$. Para estos cuatro puntos:

$$\begin{aligned}
\boldsymbol{x}_1 = (0,\ 0)^T: &\quad y = p_1x_1 + p_2x_2 + \theta = 0.616(0) + 0.691(0) + 0.171(-1) = -0.171 \\
\boldsymbol{x}_2 = (1,\ 0)^T: &\quad y = p_1x_1 + p_2x_2 + \theta = 0.616(1) + 0.691(0) + 0.171(-1) = 0.445 \\
\boldsymbol{x}_3 = (0,\ 1)^T: &\quad y = p_1x_1 + p_2x_2 + \theta = 0.616(0) + 0.691(1) + 0.171(-1) = 0.520 \\
\boldsymbol{x}_4 = (1,\ 1)^T: &\quad y = p_1x_1 + p_2x_2 + \theta = 0.616(1) + 0.691(1) + 0.171(-1) = 1.136.
\end{aligned}$$

El valor negativo para el primer patrón indica que dicho patrón se encuentra por debajo de la línea de decisión $x_2 = -0.8909x_1 + 0.2473$, por tanto, pertenece a la primera clase. Por otro lado, los valores positivos para el resto de los tres patrones indican que estos se encuentran por encima de la línea de decisión, es decir que pertenecen a la segunda clase.

Una de las ventajas de la regla Delta es que, aunque el problema de clasificación a resolver sea linealmente no separable, siempre dará como resultado una solución. Algunos patrones, por supuesto, quedarán mal clasificados al final del proceso de entrenamiento. Para ilustrar lo dicho, considérese el siguiente ejemplo.

**• • • Ejemplo 6.11**

Suponga que se quiere entrenar una neurona ADALINE de dos entradas para encontrar la línea de separación entre las clases de patrones sin separación lineal, mostradas en la figura 6.23a. Suponga que como pesos iniciales se seleccionan los siguientes: $p_1 = 1.0$, $p_2 = -1.0$ y $\theta = 1.0$, y que además $\alpha = 0.1$.

**Solución**

Para encontrar los pesos de la recta de separación entre las dos posibles clases, los pares de entrenamiento se recorren en el siguiente orden: $\boldsymbol{x}_1 = (0,\ 0)^T$, $t_1 = -1$, luego $\boldsymbol{x}_2 = (1,\ 0)^T$, $t_2 = -1$, luego $\boldsymbol{x}_3 = (0.5,\ 0.6)^T$, $t_3 = -1$, y luego $\boldsymbol{x}_4 = (0,\ 1)^T$, $t_4 = 1$, $\boldsymbol{x}_5 = (1,\ 1)^T$, $t_5 = 1$, $\boldsymbol{x}_6 = (0.5,\ 0.4)^T$, $t_6 = 1$. Para cada uno de estos pares se aplica el algoritmo en forma iterada hasta obtener la solución deseada. Los vectores aumentados son los siguientes:

$$\begin{aligned} \boldsymbol{x}_1 &= (0,\ 0,\ -1)^T,\ \boldsymbol{x}_2 = (1,\ 0,\ -1)^T,\ \boldsymbol{x}_3 = (0.5,\ 0.6,\ -1), \\ \boldsymbol{x}_4 &= (0,\ 1,\ -1)^T,\ \boldsymbol{x}_5 = (1,\ 1,\ -1)^T,\ \boldsymbol{x}_6 = (0.5,\ 0.4,\ -1)^T. \end{aligned}$$

De acuerdo con lo visto, el error cuadrático medio dado por las ecuaciones 6.36 y 6.37 deberá disminuir, provocando el posicionamiento, poco a poco, de la línea de separación entre clases. Para facilitar la comprensión de la operación de la regla Delta, se muestran los cálculos para la primera época de recorrido.

Estos se acomodan de nuevo en tablas como sigue:
**Época 1**: $p_1 = 1.0,\ p_2 = -1.0,\ \theta = 1.0$ y $\alpha = 0.1 :\ y = p_1x_1 + p_2x_2 + \theta$. Los valores para la salida y de la neurona, su error $\delta$, así como los incrementos en los tres pesos $\triangle p_1$, $\triangle p_2$ y $\triangle\theta$ se muestran en la tabla 6.15:

**Tabla 6.15**: Valores de salida de la neurona y error $\delta$.

| $x_1$ | $x_2$ | $x_3$ | $r$ | $t$ | $\delta = r - t$ | $\triangle p_1 = -0.1\delta x_1$ | $\triangle p_2 = -0.1\delta x_2$ | $\triangle p_3 = -0.1\delta x_3$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | -1 | -1.0 | -1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1 | 0 | -1 | 0.0 | -1 | 1.0 | -0.1 | 0.0 | 0.1 |
| 0.5 | 0.6 | -1 | -1.25 | -1 | -0.25 | 0.0125 | 0.015 | -0.025 |
| 0 | 1 | -1 | -2.06 | 1 | -3.06 | 0.0 | 0.306 | -0.306 |
| 1 | 1 | -1 | -0.5355 | 1 | -1.5355 | 0.15355 | 0.15355 | -0.15355 |
| 0.5 | 0.4 | -1 | -0.292605 | 1 | -1.292605 | 0.06463025 | 0.0517042 | -0.1292605 |

**Tabla 6.16**: Valores actualizados de los pesos $p_1$, $p_2$ y $\theta$.

| $p_1$ viejo | $p_2$ viejo | $\theta$ viejo | $p_1$ nuevo | $p_2$ nuevo | $\theta$ nuevo |
|---|---|---|---|---|---|
| 1.0 | -1.0 | 1.0 | 1.0 | -1.0 | 1.0 |
| 1.0 | -1.0 | 1.0 | 0.9 | -1.0 | 1.1 |
| 0.9 | -1.0 | 1.1 | 0.9125 | -0.985 | 1.075 |
| 0.9125 | -0.985 | 1.075 | 0.9125 | -0.679 | 0.769 |
| 0.9125 | -0.679 | 0.769 | 1.06605 | -0.52545 | 0.61545 |
| 1.06605 | -0.52545 | 0.61545 | 1.13068025 | -0.4737458 | 0.4861895 |

Los valores actualizados de los pesos $p_1$, $p_2$ y $\theta$, con respecto a los pesos viejos, se presentan en la tabla 6.16:

El valor del error individual debido a cada patrón se muestra en la tabla 6.17:
Finalmente, el error $e$ acumulado al final de la primera época se calcula como sigue:

$$e_1 = \frac{1}{n}\sum_{k=1}^{n} e^k = \frac{1}{6}(0.0 + 0.5 + 0.03125 + 4.6818 + 1.78888 + 0.83554) = 1.20455733.$$

El lector puede demostrar que los valores de este error a lo largo de las 10 primeras iteraciones son los que se presentan en la tabla 6.18.
El lector puede mostrar que los valores finales de los pesos $p_1$, $p_1$ y $\theta$, después de la ejecución de la décima época son los que corresponden a la tabla 6.19.

**Tabla 6.17**: Valor del error individual debido a cada patrón.

| $\delta = y - t$ | $e^k = \frac{1}{2}(y_k - t_k)^2$ |
|---|---|
| 0.0 | 0.0 |
| 1.0 | 0.5 |
| -0.25 | 0.03125 |
| -3.06 | 4.6818 |
| -1.5355 | 1.178880125 |
| -1.292605 | 0.83541384 |

**Tabla 6.18**: Valores del error a lo largo de las 10 primeras iteraciones.

| Época $i$ | $e_i$ |
|---|---|
| 1 | 1.20455733 |
| 2 | 0.82105018 |
| 3 | 0.71069428 |
| 4 | 0.62906931 |
| 5 | 0.55899079 |
| 6 | 0.5006037 |
| 7 | 0.45314097 |
| 8 | 0.41502852 |
| 9 | 0.38460621 |
| 10 | 0.36040221 |

**Tabla 6.19**: Valores finales de los pesos $p_1$, $p_1$ y $\theta$.

| $p_1$ | $p_2$ | $\theta$ |
|---|---|---|
| 0.394 | 1.051 | 0.513 |

La figura 6.23b muestra la neurona correspondiente con tres entradas, tres pesos, incluyendo en bias y una salida correspondiente a la activación de la neurona. En la figura 6.23a se muestra la gráfica del error $e$, mientras que en la figura 6.23b se muestra la línea de separación entre las dos clases de puntos. De acuerdo con la ecuación 6.19, la línea de separación entre ambas clases viene dada como $x_2 = -0.3751x_1 + 0.4876$.

Para verificar el correcto funcionamiento de la neurona resultante con pesos $p_1 = 0.394$, $p_2 = 1.051$ y $\theta$=0.513, clasifiquemos los seis puntos:

$$\begin{aligned}
\boldsymbol{x}_1 &= (0,\ 0): \quad y = 0.394(0) + 1.051(0) + 0.513(-1) = -0.513 \\
\boldsymbol{x}_2 &= (1,\ 0): \quad y = 0.394(1) + 1.051(0) + 0.513(-1) = -0.119 \\
\boldsymbol{x}_3 &= (0.5,\ 0.6): \quad y = 0.394(0.5) + 1.051(0.6) + 0.513(-1) = 0.425 \\
\boldsymbol{x}_4 &= (0,\ 1): \quad y = 0.394(0) + 1.051(1) + 0.513(-1) = 0.538 \\
\boldsymbol{x}_5 &= (1,\ 1): \quad y = 0.394(1) + 1.051(1) + 0.513(-1) = 0.932 \\
\boldsymbol{x}_6 &= (0.5,\ 0.4): \quad y = 0.394(0.5) + 1.051(0.4) + 0.513(-1) = 0.104.
\end{aligned}$$

El valor negativo para los dos primeros patrones indica que dichos patrones se encuentran por debajo de la línea de decisión $x_2 = -0.3751x_1 + 0.4876$, por tanto, dichos patrones pertenecen a la primera clase. Por otro lado, los valores positivos para los patrones 4, 5 y 6, indican que estos se encuentran por encima de la línea de decisión, por lo que pertenecen a la segunda clase. Finalmente, el valor positivo para el tercer patrón muestra que este se encuentra por encima de la línea de decisión (ver figura 6.23). De esta misma figura se observa que este patrón debería haber emitido un valor negativo, diciéndonos que ha sido mal clasificado. La neurona encontrada por la regla Delta encontró una buena línea de decisión que como ya vimos, clasificará algunos patrones de manera errónea.

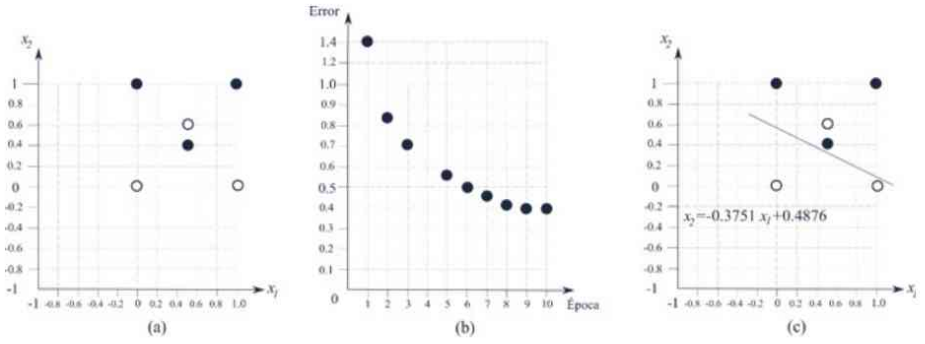



**Figura 6.23**: (a) Clases de patrones para el ejemplo 6.11. (b) Gráfica de error. (c) Línea de separación entre clases.

## 6.7 El perceptrón sigmoidal

En la subsección 6.2.5 se mencionó que una neurona puede tener a su salida diversas funciones de transferencia. Dependiendo de la función de transferencia utilizada, será el resultado obtenido a la salida de la neurona. Ya se vio que si usa una función escalón a la salida se puede obtener uno de dos valores {0, 1} para el caso de la función límite duro o {-1, 1} en el caso de una función limitadora simétrica.

Se ha visto que la regla del perceptrón es útil para el caso de perceptrones con funciones limitadoras duras o duras simétricas. Se sabe también que los perceptrones potencian su poder cuando son acomodados en capas. Si se quisiera usar una regla como Delta es necesario cambiar la función limitadora dura por una que sea diferenciable.

En esta sección se estudiará una modificación de la regla Delta para perceptrones con funciones diferenciables, en este caso para el caso de funciones semilineales o sigmoideas como la mostrada en la figura 6.6e. En otras palabras, se quiere entrenar una neurona tipo perceptrón dada por el siguiente par de ecuaciones:

$$a = \sum_{i=1}^{n+1} p_i x_i \tag{6.42}$$

$$s = \frac{1}{1+e^{-a}} \tag{6.43}$$

El índice $i$ corre de 1 a $n+1$, ya que $\boldsymbol{x} = (x_1\ , x_2, \cdots,\ x_{n+1})^T$ y $\boldsymbol{p} = (p_1,\ p_2,\ \cdots,\ \theta)^T$.

Ambas funciones poseen buenas propiedades de derivación, lo cual, por supuesto, coadyuva en el entrenamiento de la red. De nuevo, se quiere calcular el gradiente de la función de error con respecto a cada peso. Para el peso $p_i$ y para el $k$-ésimo par de entrenamiento $(\boldsymbol{x}^k,\ t_k)$, este gradiente se puede expresar a través de la regla de la cadena como:

$$\frac{\partial e_k}{\partial p_i} = \frac{\partial e_k}{\partial s_k}\frac{\partial s_k}{\partial p_i}, \tag{6.44}$$

El primer término $\frac{\partial e_k}{\partial s_k}$ representa la Delta entre el valor deseado $t_k$ y la salida calculada por la neurona $s_k$. Si se supone que la función costo es de nuevo $e_k = \frac{1}{2}(t_k - s_k)^2$, entonces:

$$\delta = \frac{\partial e_k}{\partial s_k} = -(t_k - s_k) = (s_k - t_k). \tag{6.45}$$

Por otro lado en el segundo término $\frac{\partial s_k}{\partial p_i}$ de la misma ecuación (6.44), el lector puede demostrar que a través de la regla de la cadena:

$$\frac{\partial s}{\partial p_i} = \frac{\partial s_k}{\partial a_k}\frac{\partial a_k}{\partial p_i} = s_k(1-s_k)x_i. \tag{6.46}$$

Por tanto:

$$\frac{\partial e_k}{\partial p_i} = -s_k(1-s_k)(t_k-s_k)x_i. \tag{6.47}$$

El cambio en los pesos entre la $i$-ésima entrada y la salida para para el $k$-ésimo par de entrenamiento $(\boldsymbol{x}^k,\ t_k)$ viene dado como:

$$\triangle p_i = -\alpha\frac{\partial e_k}{\partial p_i} = \alpha s_k(1-s_k)(t_k-s_k)x_i. \tag{6.48}$$

Para todos los $n$ pares de entrenamiento, este cambio viene dado como:

$$\triangle p_i = \alpha\frac{1}{n}\sum_{k=1}^{n} s_k(1-s_k)(t_k-s_k)x_i. \tag{6.49}$$

Finalmente, con respecto al vector de pesos, el vector gradiente del error $e$ viene dado como:

$$\nabla e_p = \left[\frac{\partial e}{\partial p_1},\ \frac{\partial e}{\partial p_2},\ \cdots,\ \frac{\partial e}{\partial \theta}\right]^T. \tag{6.50}$$

La correspondiente regla Delta para una neurona con función de transferencia sigmoidal viene dada como sigue:

**Pseudocódigo 6.3**

Regla de entrenamiento Delta para una neurona sigmoidal

Poner los valores del vector de pesos $\boldsymbol{p}$ en valores aleatorios, en el rango {-1, +1}.
Inicializar $\alpha$ en algún valor entre $0 < \alpha < 1$.

**repeat**
  **for** (Para cada par de entrenamiento $(\boldsymbol{x}^k,\ t_k),\ k = 1, 2, \cdots, n$)
    Calcular activación de la neurona: $a = \boldsymbol{p}^T\boldsymbol{x}^k$;
    Calcular salida de la neurona: $s = f(a) = (1 + e^{(-a)})^{(-1)}$;
    Calcular el gradiente del vector error respecto al vector de pesos $\frac{\partial e_k}{\partial \boldsymbol{p}}$;
    Calcular el cambio en el vector de pesos $\triangle \boldsymbol{p}$;
    Obtener el nuevo vector de pesos como: $\boldsymbol{p}^{\text{nuevo}} \longleftarrow \boldsymbol{p}^{\text{viejo}} + \triangle \boldsymbol{p}$
  **end**
**until** $|\bigtriangledown e| \approx 0$ ;

Para ilustrar la operación de este algoritmo de entrenamiento al caso de neuronas tipo sigmoide, considérese el siguiente ejemplo:

**• • • Ejemplo 6.12**

Entrenar una neurona sigmoide para realizar la función lógica "O" de dos entradas. Como pesos iniciales se seleccionan los siguientes: $p_1 = 0.5$, $p_2 = -0.5$ y $\theta = -1.0$. Además, sea $\alpha = 0.8$.

**Solución**

Para encontrar los pesos de la recta de separación entre las dos posibles clases, los pares de entrenamiento se recorren en el siguiente orden: $\boldsymbol{x}_1 = (0,\ 0)^T$, $t_1 = -1$, luego $\boldsymbol{x}_2 = (0,\ 1)^T$, $t_2 = 1$, luego $\boldsymbol{x}_3 = (1,\ 0)^T$, $t_3 = 1$, y luego $\boldsymbol{x}_4 = (1,\ 1)^T$, $t_4 = 1$. Para cada uno de estos pares se aplica el algoritmo descrito en forma iterada hasta obtener la solución deseada. Los vectores aumentados son los siguientes:

$$\boldsymbol{x}_1 = (0,\ 0,\ -1)^T,\ \boldsymbol{x}_2 = (0,\ 1,\ -1)^T,\ \boldsymbol{x}_3 = (1,\ 0,\ -1)^T,\ \boldsymbol{x}_4 = (1,\ 1,\ -1)^T.$$

De acuerdo con lo visto, el error cuadrático medio $e_k$ deberá disminuir provocando el posicionamiento, poco a poco, de la línea de separación entre clases. Para facilitar la comprensión de la operación de la regla Delta, se muestran los cálculos para la primera época de recorrido.

Estos se acomodan en tablas como sigue:

**Época 1**: $p_1 = 0.5$, $p_2 = -0.5$, $\theta = -1.0$ y $\alpha = 0.8$: $a = p_1x_1 + p_2x_2 + \theta$ y $s = f(a) = (1 + e^{-a})^{-1}$. Los valores para la salida y de la neurona, su error $\delta$, así como los incrementos en los tres pesos $\triangle p_1$, $\triangle p_2$ y $\triangle\theta$ se muestran en la tabla 6.20:

**Tabla 6.20**: Valores de salida para la neurona, error $\delta$ e incrementos de los tres pesos $\triangle p_1$, $\triangle p_2$ y $\triangle\theta$.

| $x_1$ | $x_2$ | $x_3$ | $s$ | $t$ | $\boldsymbol{e_p} = \frac{\partial e_k}{\partial \boldsymbol{p}}$ | $\triangle p_1 = -0.8\delta x_1$ | $\triangle p_2 = -0.8\delta x_2$ | $\triangle p_3 = -0.8\delta x_3$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | -1 | 0.731 | -1 | $[0,\ 0,\ -0.34]^T$ | 0.0 | 0.0 | 0.272 |
| 0 | 1 | -1 | 0.5566 | 1 | $[0,\ -0.11,\ 0.11]^T$ | 0.0 | 0.087 | -0.087 |
| 1 | 0 | -1 | 0.7883 | 1 | $[-0.035,\ 0,\ -0.84]^T$ | 0.028 | 0.0 | -0.028 |
| 1 | 1 | -1 | 0.7229 | 1 | $[-0.055,\ -0.055,\ 0.055]^T$ | 0.044 | 0.044 | 0.044 |

**Tabla 6.21**: Valores actualizados de los pesos $p_1,\ p_2$ y $\theta$.

| $p_1$ viejo | $p_2$ viejo | $\theta$ viejo | $p_1$ nuevo | $p_2$ nuevo | $\theta$ nuevo |
|---|---|---|---|---|---|
| 0.5 | -0.5 | -1.0 | 0.5 | -0.5 | -0.727 |
| 0.5 | -0.5 | -0.727 | 0.5 | -0.412 | -0.815 |
| 0.5 | -0.412 | -0.815 | 0.528 | -0.412 | -0.843 |
| 0.528 | -0.412 | -0.843 | 0.572 | -0.368 | -0.887 |

Los valores actualizados de los pesos $p_1,\ p_2$ y $\theta$, con respecto a los pesos viejos, se muestran en la tabla 6.21.

El valor del error individual debido a cada patrón se muestra en la tabla 6.22.

**Tabla 6.22**: Valor del error individual debido a cada patrón.

| $\delta_k = -(t_k - s_k)$ | $e^k = \frac{1}{2}(t_k - s_k)^2$ |
|---|---|
| -1.731 | 1.498 |
| 0.443 | 0.098 |
| 0.211 | 0.022 |
| 0.277 | 0.039 |

Finalmente, el error $e$ acumulado al final de la primera época se calcula como sigue:

$$e_1 = \frac{1}{n}\sum_{k=1}^{n} e_k = \frac{1}{4}(1.498 + 0.098 + 0.022 + 0.039) = 0.414.$$

El lector puede demostrar que los valores de este error a lo largo de las 10 primeras iteraciones son las que se indican en la tabla 6.23.

De igual manera, el lector puede mostrar que los valores finales de los pesos $p_1,\ p_2$ y $\theta$, después de la ejecución de la décima época son las que se presentan en la tabla 6.24.
La figura 6.22 ilustra la neurona correspondiente con tres entradas, tres pesos, incluyendo en bias y una salida correspondiente a la activación de la neurona.

En la figura 6.24a se muestra la gráfica del error $e$, mientras que en la figura 6.24b se ve la línea de separación entre las dos clases de puntos que realizan la función "O" lógica.

**Tabla 6.23**: Error de las primeras 10 iteraciones.

| Época $i$ | $e_i$ |
|---|---|
| 1 | 0.414 |
| 2 | 0.403 |
| 3 | 0.392 |
| 4 | 0.380 |
| 5 | 0.368 |
| 6 | 0.355 |
| 7 | 0.342 |
| 8 | 0.330 |
| 9 | 0.318 |
| 10 | 0.307 |

**Tabla 6.24**: Valores finales de los pesos $p_1$, $p_2$ y $\theta$.

| $p_1$ | $p_2$ | $\theta$ |
|---|---|---|
| 1.23 | 0.72 | 0.32 |

De acuerdo con la ecuación (6.19), la línea de separación entre ambas clases viene dada como $x_2 = -1.71x_1 + 0.44$. Para verificar el correcto funcionamiento de la neurona resultante con pesos $p_1 = 1.23$, $p_2 = 0.72$ y $\theta = 0.32$, clasifiquemos los 4 puntos: $\boldsymbol{x}_1 = (0,\ 0)^T$, $\boldsymbol{x}_2 = (0,\ 1)^T$, $\boldsymbol{x}_3 = (1,\ 0)^T$, y $\boldsymbol{x}_4 = (1,\ 1)^T$.

Para estos cuatro puntos:

$$
\begin{aligned}
\boldsymbol{x}_1 &= (0,\ 0)^T : \ a = 1.23(0) + 0.72(0) + 0.32(-1) = -0.32,\ s = \tfrac{1}{1+e^{-a}} = \tfrac{1}{1+e^{-(-0.32)}} = 0.42 \\
\boldsymbol{x}_2 &= (0,\ 1)^T : \ a = 1.23(0) + 0.72(1) + 0.32(-1) = 0.4,\ s = \tfrac{1}{1+e^{-a}} = \tfrac{1}{1+e^{-0.4}} = 0.598 \\
\boldsymbol{x}_3 &= (1,\ 0)^T : \ a = 1.23(1) + 0.72(0) + 0.32(-1) = 0.91,\ s = \tfrac{1}{1+e^{-a}} = \tfrac{1}{1+e^{-0.91}} = 0.713 \\
\boldsymbol{x}_4 &= (1,\ 1)^T \ : a = 1.23(1) + 0.72(1) + 0.32(-1) = 1.63,\ s = \tfrac{1}{1+e^{-a}} = \tfrac{1}{1+e^{-1.63}} = 0.836.
\end{aligned}
$$

El valor de 0.42 abajo del 0.5 (punto de inflexión de la función sigmoide, figura 6.6e) y cerca del 0 para el primer patrón, indica la pertenencia de dicho patrón a la primera clase. Note que en este caso el patrón está debajo de la línea de decisión (figura 6.24b). Por otro lado, los valores de 0.598, 0.713, y 0.836 arriba del 0.5 y cercanos al valor de 1.0 para el resto en los tres patrones, indican una probabilidad alta de su pertenencia a la segunda clase. Note que en estos tres casos los puntos están por encima de la línea de decisión (figura 6.24b).

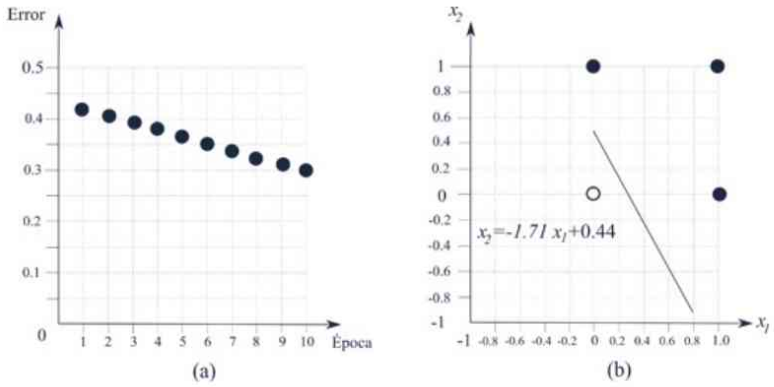



**Figura 6.24**: (a) Gráfica del error $e$. (b) Línea de separación entre clases.

### • • • Ejemplo 6.13

Considere un robot tipo péndulo mostrado en la figura 6.25, desarrolle un programa en MATLAB que implemente dos neuronas con función de transferencia tangente hiperbólica (tanh), una neurona para posicionar al péndulo desde cualquier posición inicial en una referencia deseada $d(n) = \frac{\pi}{2}$ rad; y la segunda neurona que produzca en el péndulo una respuesta transitoria sin sobretiros ni oscilaciones o picos, tal que alcance el estado estacionario libre de vibraciones mecánicas, en forma suave (respuesta sobreamortiguada). Representar el error de posicionamiento $e(n)$.

### Solución

Como una aplicación directa de redes neuronales en control de servomecanismos, considere el caso de dos neuronas con función de transferencia tangente hiperbólica; una neurona para controlar la posición y otra para la inyección de amortiguamiento, tal que la respuesta transitoria del péndulo (ver figura 6.25) no exhiba sobreimpulso o picos ni rizo o vibraciones mecánicas y le permita estabilizarse en la posición deseada $d(n)$ en forma suave y libre de ruido mecánico.

Desde el punto de vista de control automático, la figura 6.13 representa control en lazo abierto puesto que la respuesta de la neurona $y(n)$ aún no está conectada a alguna planta física, por ejemplo al péndulo. Por otro lado, la figura 6.26 se conoce como control en lazo cerrado, debido a que la respuesta del péndulo se retroalimenta a la neurona a través del error $e(n) = d(n) - q(n)$; donde el movimiento del péndulo está representado por $q(n)$. La respuesta de ambas neuronas (la de posición y amortiguamiento) es convertida a energía mecánica y aplicada al péndulo, de esta forma se puede mover con desplazamiento articular $q$ y velocidad rotacional $\dot{q}$. El objetivo de control consiste en lograr que $e(n) \to 0$ como $n \to \infty$, entonces si $e(n)$ es cero, el péndulo habrá alcanzado la posición deseada $d$, es decir: $q(n) = d(n)$ y por lo tanto, su velocidad de movimiento $\dot{q}(n) = 0$. El péndulo se mantendrá es esta posición $q(n)$, hasta que cambie el valor de la referencia deseada $d(n)$.

Además, un aspecto importante en control de robots manipuladores es que la respuesta transitoria sea suave, sin sobreimpulsos o picos y que llegue a la referencia sin oscilaciones. Para lograr esto, se incluye un elemento de amortiguamiento o freno mecánico realizado a través de una neurona tangente hiperbólica que procesa la velocidad de movimiento $\dot{q}$, como se indica en la figura 6.26.

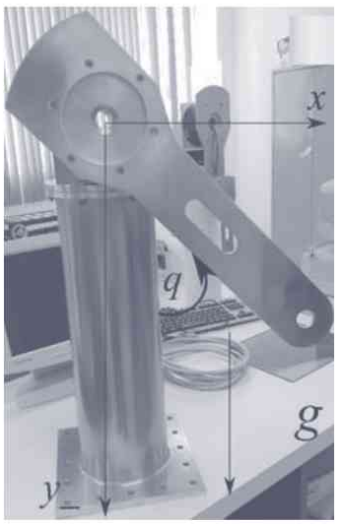



**Figura 6.25**: Péndulo robot.

La figura 6.25 muestra a un servomecanismo real configurado como péndulo robot; el cual consiste en un servomotor de transmisión directa de 15 Nm, acoplado a una barra metálica de aluminio con longitud $l_p$ y sometido a la acción de la gravedad. El péndulo se mueve en el plano vertical $x - y$, por lo que cambios de posición $q$ generan un gradiente de energía potencial, es decir, se produce el par gravitacional como uno de los efectos físicos que se encuentra presente en la dinámica no lineal del péndulo. Generalmente, la posición de casa del péndulo es seleccionada a conveniencia del usuario, sin embargo, es común ubicarla sobre el eje $y_-$, por lo que la posición inicial $q(0) = 0$ rad, así como velocidad de movimiento inicial $\dot{q} = 0$ $\frac{\text{rad}}{\text{s}}$. La posición $q(n)$ del péndulo es controlada a través de una red neuronal simple con función de transferencia tangente hiperbólica y otra neurona (con igual estructura tangente hiperbólica) para inyectar amortiguamiento a través de la velocidad de movimiento $\dot{q}(n)$, tal que el péndulo

tenga una respuesta transitoria sin picos ni oscilaciones, evitando vibraciones mecánicas y alcanzando el estado estacionario de manera suave, sin rizo ni ruido en las señales de posición $q(n)$ y velocidad rotacional de movimiento $\dot{q}(n)$.

La figura 6.26 presenta el diagrama a bloques para controlar la posición $q(n)$ del péndulo con efecto de amortiguamiento $\dot{q}(n)$. Se emplean dos neuronas de la forma tangente hiperbólica, una de ellas procesa el error de posición $e(n)$ y la segunda procesa la velocidad de movimiento $\dot{q}(n)$ para inyectar energía de amortiguamiento.

Observe también de la figura 6.26 que se requiere de compensación del par gravitacional, esto es debido a que el péndulo se mueve en el plano vertical $x - y$; la acción del campo gravitacional es sobre el eje $y_-$, entonces su energía potencial provoca un gradiente de la energía potencial, es decir, el fenómeno físico del par gravitacional se requiere compensar con la misma magnitud pero en sentido contrario. El par gravitacional del péndulo se encuentra dado por $mgl_c \operatorname{sen}(q)$ (ver tabla 6.25 para descripción y nomenclatura de variables y parámetros del péndulo).

El algoritmo de control está formado por dos neuronas simples con estructura tangente hiperbólica de la siguiente forma:

$$\tau \quad = \quad y_1 - y_2 + mgl_c \operatorname{sen}(q) = k_p \tanh(e) - k_v \tanh(\dot{q}) + mgl_c \operatorname{sen}(q). \qquad (6.51)$$

La neurona de posición está compuesta por $y_1 = k_p \tanh(e)$, donde $y_1$ es la respuesta de la neurona de posición, $e = d - q$ representa el error de posición, el peso ponderado de la neurona $y_1$ es $k_p$, también conocido en el ámbito de control como ganancia proporcional. Por otro lado, la neurona encargada de inyectar amortiguamiento es: $y_2 = k_v \tanh(\dot{q})$, donde $k_v$ es el peso para esta neurona, conocido como ganancia derivativa y al término de amortiguamiento $-y_2 = -k_v \tanh(\dot{q})$ se le conoce en robótica y control como acción de control derivativa. Entonces, el torque o energía mecánica aplicada al péndulo resulta: $\tau = y_1 - y_2 + mgl_c \operatorname{sen}(q)$.

Evidentemente, en un sistema real, el péndulo tiene un sistema electrónico que convierte la señal de entrada en Volts a energía aplicada al servomotor. La señal del algoritmo de control $\tau$ implementado en lenguaje C, por ejemplo en un sistema empotrado, $\tau$ es una señal de entrada al amplificador electrónico del péndulo: dicha señal de entrada se encuentra en Volts (se requiere la conversión de digital hacia analógico, usando un DAC al menos con una resolución de 12 bits), entonces el amplificador electrónico transforma ese voltaje en energía mecánica o torque [Nm].

Debido a que el control del péndulo es un proceso dinámico no lineal, se requiere de un modelo que describa el comportamiento dinámico del péndulo; el cual está dado como:

$$\tau \quad = \quad i_p\ddot{q} + b\dot{q} + mgl_c \text{ sen}(q). \tag{6.52}$$

El listado completo de parámetros y variables de estado del algoritmo de control del péndulo (6.52) se encuentra descrito en la tabla 6.25. El modelo dinámico del péndulo (6.52) describe los efectos físicos presentes en el sistema mecánico, tales como efecto inercial: $i_p\ddot{q}$; fricción viscosa $b\dot{q}$; y, el par gravitacional $mgl_c$ sen$(q)$.

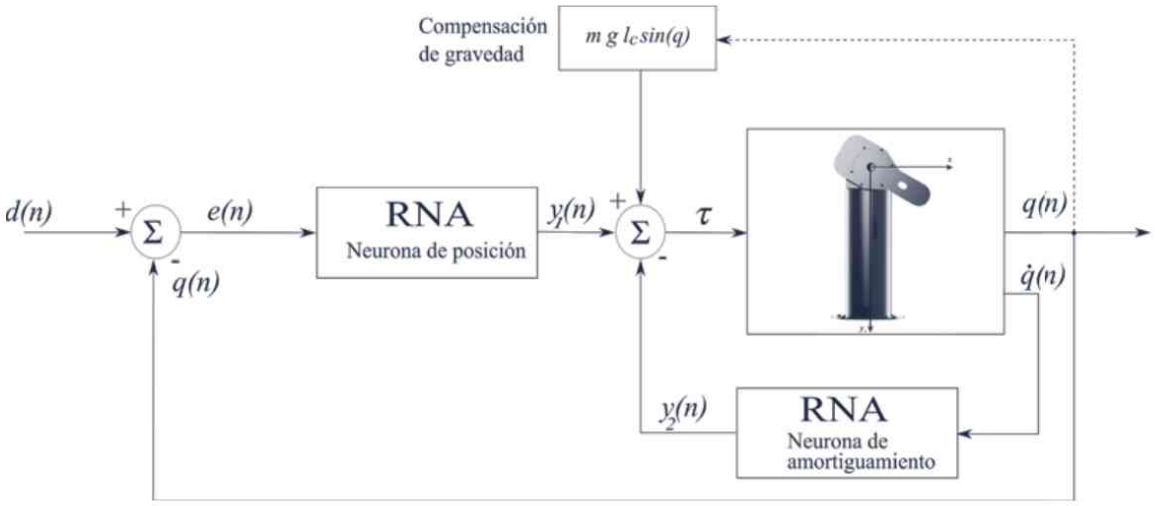



**Figura 6.26**: Diagrama de bloques para controlar la posición de un péndulo a través de neuronas RNA tangente hiperbólica.

A continuación se describe el procedimiento de simulación del algoritmo de control (tangente hiperbólica y acción de control derivativo hiperbólico), así como todos los detalles de programación.

El formato matemático requerido para simular cualquier sistema dinámico (lineal y no lineal) es por medio de una ecuación diferencial ordinaria de primer orden (ODE) expresada en las variables de estado que definen el problema de control de posición $[\, q \quad \dot{q}\,]^T$; la cual se obtiene por combinar el algoritmo de control compuesto por redes neuronales tangente hiperbólicas (6.51) y el modelo dinámico del péndulo (6.52), resultando el siguiente sistema dinámico (conocido también como ecuación en lazo cerrado):

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} \quad = \quad \begin{bmatrix} \dot{q} \\ \frac{1}{i_p}\left[\, \tau - b\dot{q} - mgl_c \text{ sen}(q)\,\right] \end{bmatrix}. \tag{6.53}$$

La ecuación en lazo cerrado (6.53) es una ODE y se encuentra ya lista para ser implementada a través de cierto algoritmo de integración numérica, por ejemplo Runge-Kutta 4/5 adaptable (para lectores que no están familiarizados con diseño de algoritmos de control y su simulación en MATLAB, consultar [43] y [44]).

Para simular el control de posición del péndulo se requieren de tres archivos en MATLAB denominados cap6_simupendulo.m, cap6_pendulo.m y cap6_tanhpendulo.m, los cuales se encuentran descritos en los cuadros de código 6.1, 6.2 y 6.3, respectivamente.

**Tabla 6.25**: Lista de parámetros y variables de estado del péndulo.

| Descripción | Símbolo | Valor/Unidades |
|---|---|---|
| Posición aticular | $q$ | rad |
| Velocidad aticular | $\dot{q}$ | $\frac{\text{rad}}{\text{s}}$ |
| Torque o par | $\tau$ | Nm |
| Masa | $m$ | 5 kg |
| Centro de masa | $l_c$ | 0.01 m |
| Constante de la aceleración gravitacional | $g$ | 9.81 $\frac{\text{m}}{\text{s}^2}$ |
| Coeficiente de fricción viscosa | $b$ | 0.17 Nm-s |
| Momento de inercia | $i_p$ | 0.165 Nm-s |
| Longitud | $l_p$ | 0.45 m |

El programa principal es cap6_simupendulo.m (cuadro de código 6.1). La línea de programación 5 define los parámetros de simulación: el vector tiempo de 0 a 10 segundos, con un paso de integración $h = 2.5$ mseg. La línea 5 configura las cotas superiores ($10^{-06}$) de los errores de integración numéricos por Runge-Kutta; la línea 8 es para el proceso de simulación de control para el péndulo. Las líneas 10 y 11 presentan las gráficas posición $q = x(:, 1)$ y la velocidad de movimiento del péndulo $\dot{q} = x(:, 2)$.

El error de posición $e = d - q$ se encuentra representado en la línea 13 (ver figura 6.27). Observe la convergencia asintótica del error de posición, lo que indica que la posición del péndulo $q$ alcanza la posición deseada $d = \frac{\pi}{2}$rad o 90°. También note el comportamiento sin sobreimpulsos, ni rizo o vibración mecánica de la respuesta transitoria del péndulo, alcanzado el estado estacionario en forma suave, tal y como se ilustra en las figuras 6.27a y 6.27b; esto se debe al desempeño de las redes neuronales compuestas por funciones hiperbólicas.

**Código MATLAB 6.1 cap6_simupendulo.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 6. El perceptrón y su entrenamiento.
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**"  .

Programa: cap6_simupendulo.m MATLAB versión 2020b

```
1  clc; % limpia pantalla.
2  clearvars; %remueve todas las variables del espacio actual de trabajo.
3  close all; % cierra gráficas, archivos y recursos abiertos.
4  format short % formato corto a 4 dígitos después del punto decimal.
5  ti=0; tfinal=10; h=0.0025; ts=ti:h:tfinal; %vector de tiempo.
6  opciones=odeset('RelTol',1e-06,'AbsTol',1e-06,'InitialStep',h,'MaxStep',h);
7  % simulación numérica del sistema ODE (6.53).
8  [t, x]=ode45('pendulo',[ti; tfinal],[0; 0],opciones);
9  figure
10 subplot(2,1,1); plot(t,180*x(:,1)/pi) % posición q en grados.
11 subplot(2,1,2); plot(t,180*x(:,2)/pi) % velocidad q̇ en grados/segundos.
12 figure
13 plot(t, 180*(pi/2-x(:,1))/pi) % error de posición e en grados (ver figura 6.27).
14 figure
15 % cinemática directa del péndulo ubicada en el IV cuadrante del plano x − y.
16 lp=0.45;  xp=lp*sin(x(:,1)); % coordenada x_p: x_p = l_p sen(q).
17 yp=-lp*cos(x(:,1)) % coordenada y_p: y_p = −l_p cos(q).
18 plot(xp, yp) % movimiento del péndulo en su espacio de trabajo, figura 6.27b.
```

El programa principal cap6_simupendulo.m utiliza la función pendulo.m, cuyo código fuente se encuentra descrito en el cuadro de código **MATLAB** 6.2; esta función implementa el modelo dinámico del péndulo, ecuación (6.52). Los valores numéricos de los parámetros del péndulo son los descritos en la tabla 6.25. Observe que en la línea de programación 11 se registra la energía mecánica (torque) del algoritmo de control RNA tangente hiperbólico (6.51), el cual se encuentra implementado en la función cap6_tanhpendulo.m del cuadro de código **MATLAB** 6.3 y es parte fundamental de la implementación de la ODE (6.53), como se ilustra en la línea de programación 14.

**Código MATLAB 6.2 cap6_pendulo.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización**.
Capítulo 6. El perceptrón y su entrenamiento
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap6_pendulo.m **MATLAB** versión 2020b

```
1  function xp=pendulo(t,x)
2      % vector de estados (entradas)
3      q=x(1);% posición articular q
4      qp=x(2);% velocidad articular q̇
5      % parámetros del péndulo.
6      m=5;% masa.
7      lc=0.01;% centro de gravedad.
8      g=9.81;% constante de aceleración gravitacional g.
9      b=0.17;% coeficiente de fricción viscosa b.
10     ip=0.16;% momento de inercia del rotor i_p.
11     tau=tanhpendulo(q,qp);% algoritmo de control RNA hiperbólico.
12     qpp=(tau-b*qp-m*g*lc*sin(q))/ip;% aceleración del péndulo.
13     % ODE ecuación (6.53).
14     xp=[qp;% xp(1)=x(2) velocidad articular.
15         qpp];%xp(2)=qpp aceleración articular.
16 end
```

En el cuadro de código **MATLAB** 6.3 se describe la función cap6_tanhpendulo.m, que implementa el algoritmo de control hiperbólico, ecuación (6.51). En la línea de programación 18 se encuentran programadas ambas neuronas, para el error de posición $e$ y para la inyección de amortiguamiento, usando la velocidad de movimiento rotacional $\dot{q}$ del péndulo.

**Código MATLAB 6.3 cap6_tanhpendulo.m**

**Inteligencia Artificial Aplicada a Robótica y Automatización.**
Capítulo 6. El perceptrón y su entrenamiento
Juan Humberto Sossa Azuela y Fernando Reyes Cortés.
Alfaomega Grupo Editor: "**Te acerca al conocimiento**" .

Programa: cap6_tanhpendulo.m **MATLAB** versión 2020b

```
1  function tau = tanhpendulo(x,xp)
2      % variables de estados (entradas).
3      q=x;% posición articular.
4      qp=xp;% velocidad articular.
5      % parámetros del péndulo (ver tabla 6.25).
6      m=5;% masa m.
7      lc=0.01;% centro de gravedad l_c.
8      g=9.81;% constante de aceleración gravitacional g.
9      % pesos de la red neuronal k_p y k_v.
10     kp=4;% ganancia proporcional.
11     kv=4;% ganancia derivativa.
12     d=pi/2;% referencia (posición deseada d = pi/2 rad).
13     % torque o par gravitacional.
14     par_grav = m*g*lc*sin(q);% gradiente de la energía potencial.
15     % error de posición e = d - q.
16     e=d-q;
17     % algoritmo de control RNA tangente hiperbólico, ecuación (6.51).
18     tau=kp*tanh(e)-kv*tanh(qp)+par_grav;
19 end
```

La figura 6.27a muestra la convergencia hacia 0 del error de posición $e = 0$, es decir, alcanza la posición deseada $d$. Observe que la respuesta transitoria está libre de sobreimpulsos, sin oscilaciones ni vibraciones y entra al estado estacionario de manera suave, sin rizo o ruido mecánico. La figura 6.27b corresponde a la trayectoria en el plano $x - y$.

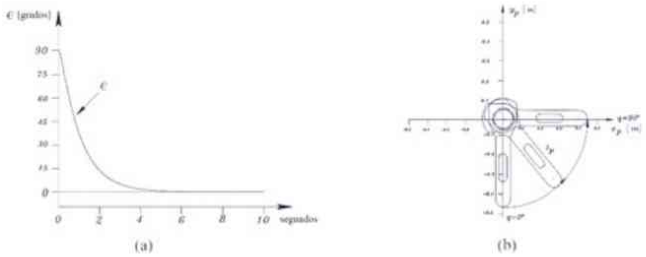



**Figura 6.27**: Error de posición $e(n) = d(n) - q(n)$ $(n = 4000)$ del esquema RNA tangente hiperbólica y trayectoria en el plano $x - y$.

**Procedimiento para ejecutar el programa: cap6_pendulo.m**

Los archivos de los programas: cap6_simupendulo.m, cap6_pendulo.m y cap6_tanhpendulo.m se encuentran disponibles en el sitio Web de esta obra.

Descargar en disco duro todos los archivos a la misma carpeta definida por el usuario. Dentro del Editor de MATLAB, seleccionar el programa principal cap6_simupendulo.m y presionar el comando **Run**.

## 6.8 Resumen

En este capítulo se sentaron las bases del paradigma de cómputo neuronal, muy útil en la clasificación de patrones, la predicción el control de procesos. La neuronal artificial tiende a simular, de manera muy vaga, la operación de la neurona biológica. Primeramente, se habló de aspectos básicos relacionados con la neurona biológica y el cerebro de criaturas vivas. Después se estudió la unidad de umbralado lógico (UUL), sus componentes y operación. Se presentaron algunos ejemplos correspondiente a la regla de entrenamiento con ejemplos numéricos donde esta máquina puede ser usada. Enseguida se estudió el perceptrón como máquina ampliada de la UUL, sus componentes y la correspondiente regla

de entrenamiento, algunos ejemplos explicativos y limitaciones. La neurona ADALINE y la regla Delta fueron presentadas. Finalmente, se dieron los detalles de cómo al dotar a un perceptrón de una función de transferencia sigmoidal, este nuevo modelo de neurona es capaz de resolver problemas lineales y no lineales al mapear un vector de entrada a un valor en un rango entre 0 y 1; el cual es visto como la probabilidad de pertenencia de un patrón de entrada a una clase.

## 6.9 Problemas propuestos

En esta sección el lector encontrará un conjunto de problemas propuestos, cuyo desarrollo y solución le permitirán reafirmar los conceptos vistos en este capítulo.

6.9.1 Aplique las ecuaciones (6.3) y (6.4) a los siguientes pares de vectores:

a) $\boldsymbol{p} = [2 \quad -3 \quad 4]^T$ y $\boldsymbol{x} = [2 \quad 3 \quad -5]^T$.

b) $\boldsymbol{p} = [-5 \quad 7 \quad 2]^T$ y $\boldsymbol{x} = [-3 \quad -2 \quad 1]^T$.

b) $\boldsymbol{p} = [0.3 \quad -0.4 \quad 0.1 \quad -0.25]^T$ y $\boldsymbol{x} = [2 \quad -4 \quad 2 \quad -1]^T$.

6.9.2 Sobre los mismos pares de vectores del problema 6.9.1, aplique ahora las ecuaciones (6.42) y (6.43).

6.9.3 Aplique la regla del perceptrón para ajustar los pesos de un perceptrón con los siguientes conjuntos de entrenamiento:

a) $\boldsymbol{x}_1 = (2,3)^T$, $t_1 = 0$; $\boldsymbol{x}_2 = (5,2)^T$, $t_2 = 0$; $\boldsymbol{x}_3 = (3,3)^T$, $t_1 = 0$; $\boldsymbol{x}_4 = (3,6)^T$, $t_4 = 1$; $\boldsymbol{x}_5 = (6,7)^T$, $t_5 = 1$; $\boldsymbol{x}_6 = (4,8)^T$, $t_6 = 1$.

b) $\boldsymbol{x}_1 = (2,3,1)^T$, $t_1 = 0$; $\boldsymbol{x}_2 = (5,2,2)^T$, $t_2 = 0$; $\boldsymbol{x}_3 = (3,3,3)^T$, $t_1 = 0$; $\boldsymbol{x}_4 = (3,6,4)^T$, $t_4 = 1$; $\boldsymbol{x}_5 = (6,7,5)^T$, $t_5 = 1$; $\boldsymbol{x}_6 = (4,8,5)^T$, $t_6 = 1$.

6.9.4 Aplique la regla del Delta para ajustar los pesos de una neurona tipo ADALINE con los mismos conjuntos de entrenamiento del problema 6.9.3.

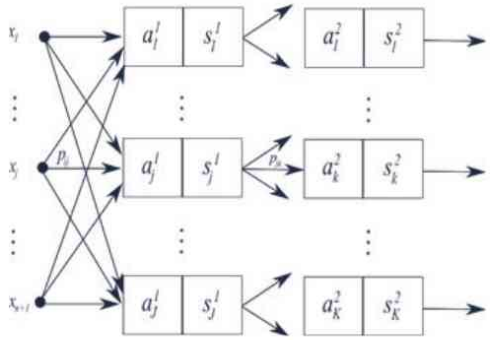
6.9.5 Utilice las neuronas ADALINE entrenadas en el ejemplo 6.5 y clasifique los vectores del problema 6.9.4.

6.9.6 Utilice las neuronas ADALINE entrenadas en el problema 6.5 y clasifique los vectores del problema 6.9.4.

6.9.7 Aplique la regla Delta modificada para ajustar los pesos de una neurona tipo sigmoidal con los mismos conjuntos de entrenamiento del problema 6.9.4.

Capítulo 7

# El perceptrón multicapa y su entrenamiento



# Descripción del capítulo

En este capítulo se continúa con el estudio del perceptrón como máquina de procesamiento de información, pero ahora acomodado en capas. Se presentan los detalles sobre la derivación de la regla Delta generalizada, útil para el entrenamiento de arreglos de perceptrones en capas empleado en la solución de problemas no lineales.

Enseguida se explica la función de la capa intermedia de transformar un problema no lineal de clasificación en uno lineal para que el último o últimos perceptrones, que son máquinas lineales, puedan resolver el problema. Después, se detalla cómo al agregar el término de momento en la ecuación de ajuste de pesos de la red neuronal, se consigue acelerar dicho proceso de ajuste.

Finalmente, se ofrecen las bases de operación de una red neuronal convolucional. Esto se hace a través de un ejemplo, lo cual permite que el lector aprecie de manera sencilla las capacidades de esta poderosa máquina de procesamiento.

Los siguientes temas son abordados:

Perceptrón multicapa.

Red neuronal convolucional.

# 7.1 Introducción

En este capítulo, en primer lugar, se presentan los detalles sobre el perceptrón multicapa. Se estudia cómo al acomodarlo en capas puede utilizarse para resolver problemas de clasificación convexos y no convexos.

En segundo lugar, se ofrecen todos los detalles sobre la derivación de la regla Delta generalizada, útil para el entrenamiento de arreglos de perceptrones en capas para la solución de problemas no lineales.

En tercer lugar, y para el caso particular de una red neuronal artificial de tres capas, se explica mediante un ejemplo la función de la capa intermedia para transformar un problema no lineal de clasificación en uno lineal; esto con el fin de que el último o últimos perceptrones, que son máquinas lineales puedan resolver el problema.

En cuarto lugar, se explica cómo al agregar el término de momento en la ecuación de ajuste de pesos de la red neuronal, se consigue acelerar dicho proceso de ajuste.

En quinto lugar, se ofrecen las bases de operación de una red neuronal convolucional. Esto se hace a través de un ejemplo, lo cual permite que el lector aprecie de manera sencilla las capacidades de esta poderosa máquina de procesamiento.

Al final, como siempre, se da un resumen y un conjunto de ejercicios para que el lector ponga en práctica sus capacidades y habilidades como programador.

# 7.2 El perceptrón multicapa

En esta sección se estudiará el perceptrón multicapa como máquina general para la solución de problemas no lineales, en particular, de clasificación. También revisaremos la regla Delta generalizada o regla Delta con propagación de errores hacia atrás, útil para el entrenamiento de perceptrones acomodados en varias capas.

### 7.2.1 El perceptrón multicapa y la regla Delta

Cuando varios perceptrones son acomodados en varias capas sucesivas se obtiene lo que se conoce como una red neuronal multinivel o red neuronal multicapa de perceptrones. Es así cuando realmente el perceptrón, siendo una máquina de procesamiento lineal, potencia su poder permitiendo resolver problemas no lineales, de alta complejidad.

En las figuras 7.1a y 7.1d se muestran dos configuraciones, una con una capa intermedia y otra con dos capas intermedias. Con dos capas se pueden resolver problemas de separación bi-clase, como los mostrados en las figuras 7.1b y 7.1c. Con tres capas, por otro lado, se pueden resolver problemas como los mostrados en las figuras 7.1e y 7.1f. Estos últimos problemas no pueden ser resueltos con un arreglo en dos capas debido a que con ellas se pueden generar superficies de separación no convexas.

Aunque en principio, como función de activación de cada neurona se puede usar cualquier función de transferencia, se suele utilizar una que sea diferenciable, para poder aplicar reglas que hagan uso de derivadas, como es el caso de la regla Delta generalizada, que a continuación se describe.

### 7.2.2 La regla Delta generalizada

La regla Delta generalizada (RDG) o regla de propagación hacia atrás (RPHA) tiene sus antecedentes en la regla Delta descrita anteriormente. Al igual que la primera, la RDG o RPHA utiliza el principio del descenso del gradiente para el ajuste de los pesos sinápticos de una red neuronal. La regla RDG es uno de los procedimientos más utilizados en la actualidad. En lo general, incorpora dos fases de operación:

1) **Fase de cálculos hacia adelante**. Durante esta fase los valores de un vector de entrada $\boldsymbol{x} = (x_1,\ x_2,\ \cdots,\ x_n,\ x_{n+1})^T$ son propagados de la capa de entrada hacia la capa de salida de la red, pasando por todas las capas intermedias de la red y dando como resultado un conjunto de salidas.

2) **Fase de cálculos hacia atrás**. Durante esta fase los errores a la salida son propagados hacia atrás, desde la capa de salida hacia la capa de entrada, siendo usados para realizar los ajustes correspondientes sobre los pesos de interconexión entre las neuronas de las diferentes capas de la red.

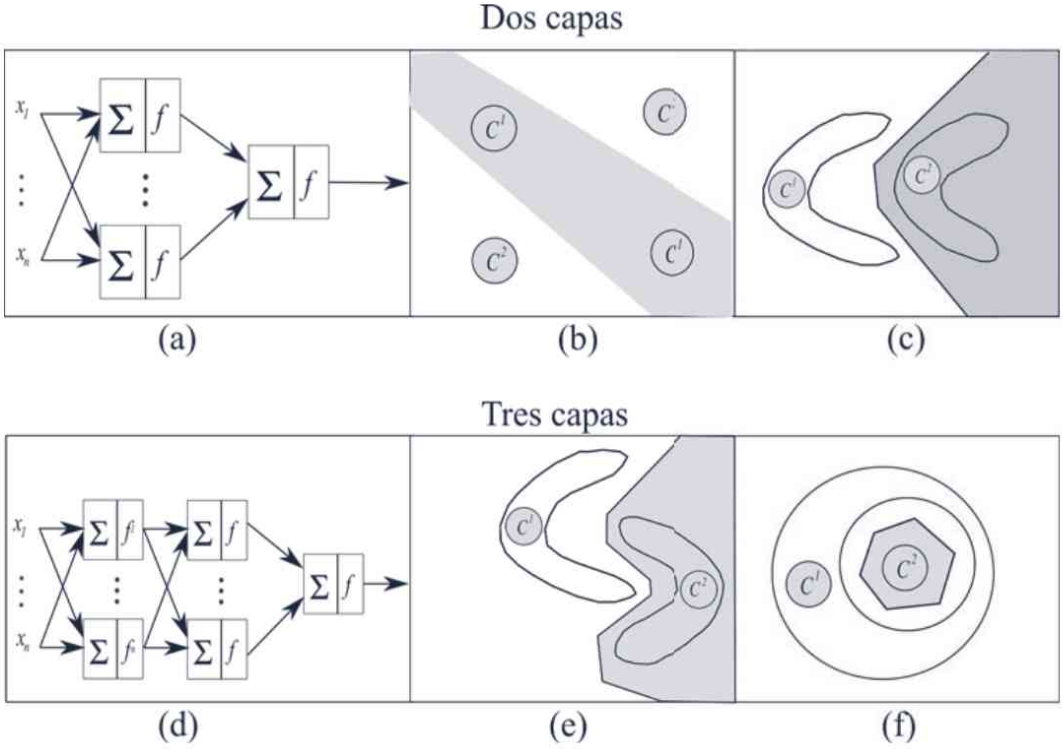



**Figura 7.1**: (a) Red neuronal con dos capas. (b) y (c) Problemas de clasificación que pueden ser resueltos con esta arquitectura. (d) Red neuronal con tres capas. (e) y (f) Problemas de clasificación que pueden ser resueltos con esta arquitectura (adaptada de Lippman [36]).

En esta sección se verán los pormenores para la derivación de la RDG, con un conjunto de ejemplos que ayudarán al lector a entender los principios de operación de este poderoso método. Sin pérdida de generalidad, se supondrá una red de tres capas, una de entrada (0), una intermedia (1) y una de salida (2) como se muestra en la figura 7.2. Se usará la siguiente notación:

1) La capa de entrada contiene $n+1$ neuronas (representadas en la figura 7.2 por puntos negros). Estas neuronas reciben los $n+1$ valores del vector de entrada $\boldsymbol{x}$ (incluye el bias) y los envían a las neuronas de la capa intermedia. El $i$-ésimo elemento de la $i$-ésima neurona se representa por la variable $x_i$ (figura 7.2).

2) La activación de la $j$-ésima neurona de la capa intermedia (capa 1) se denotará como $a_j^1$, mientras que la correspondiente activación de la $k$-ésima neurona de la capa de salida (capa 2) se denotará como $a_k^2$.

3) El estado de salida de la $j$-ésima neurona de la capa intermedia (capa 1) se denotará como $s_j^1$, mientras que el correspondiente estado de la $k$-ésima neurona de la capa de salida (capa 2) se denotará como $s_k^2$.

4) El número de neuronas en cada una de las tres capas viene dado por las variables $I$, $J$ y $K$, respectivamente (figura 7.2).

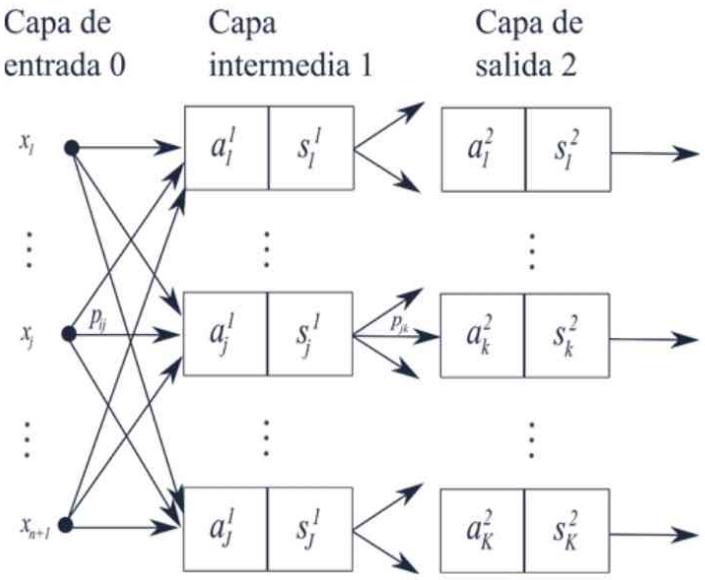



**Figura 7.2**: Red neuronal con $I$ entradas, $J$ neuronas en la capa escondida y $K$ neuronas en la capa de salida.

**Fase de propagación de los valores hacia adelante, de la entrada hacia la salida**. Durante esta fase, el $d$-ésimo vector de entrenamiento de los $n$ componentes: $\boldsymbol{x}_d = (x_{d,1},\ x_{d,2} \cdots x_{d,n},\ x_{d,n+1} = -1)^T$, junto con su etiqueta de pertenencia $t_d$ es propagado desde la capa de entrada hasta la capa de salida para obtener los resultados correspondientes.

Para el $d$-ésimo patrón de entrenamiento, de acuerdo con lo visto anteriormente, se tiene lo siguiente:

1) La entrada a la $j$-ésima neurona de la capa intermedia (como se muestra en la figura 7.2) viene dada como:

$$a^1_{jd} = \sum_{i=1}^{I+1} p_{ij} x_{id} \tag{7.1}$$

donde, como ya vimos, $x_{id}$ es el $i$-ésimo elemento del vector de entrada $\boldsymbol{x}_d$ y $p_{ij}$ es el peso de interconexión en la $i$-ésima neurona de la capa de entrada y la $j$-ésima neurona de la capa intermedia (ver figura 7.2). El índice $i$ corre de 1 a $I+1$ ya que incluye el bias del vector de entrada.

Si se supone que la función de activación de la $j$-ésima neurona de la capa intermedia es sigmoidal (ver capítulo 6, figura 6.6e), entonces su salida viene dada como:

$$s^1_{jd} = f(a^1_{jd}) = \left(1 + e^{-a^1_{jd}}\right)^{-1}. \tag{7.2}$$

2) De manera análoga, para la $k$-ésima neurona de la capa de salida (figura 7.2), su entrada y su correspondiente salida (suponiendo que su función de activación es también sigmoidal) vienen dadas, respectivamente, como:

$$a^2_{kd} = \sum_{j=1}^{J+1} p_{jk} s^1_{jd} \tag{7.3}$$

$$s^2_{kd} = f(a^2_{kd}) = \left(1 + e^{-a^2_{kd}}\right)^{-1}. \tag{7.4}$$

Con $p_{jk}$ el peso de interconexión en la $j$-ésima neurona de la capa escondida y la $k$-ésima neurona de la capa de salida (figura 7.2), $s^1_{jd}$ la salida de la $j$-ésima neurona intermedia, $a^2_{kd}$ la activación de la $k$-ésima neurona de la capa de salida y $s^2_{kd}$ su correspondiente salida.

El índice $j$ corre de 1 a $J+1$ ya que incluye el bias del vector de entrada correspondiente que, en este caso, se compone de las salidas de las neuronas de la capa intermedia.

3) En resumen, la salida de la $k$-ésima neurona de la capa de salida viene dada por la siguiente expresión:

$$s_{kd}^2 = f_k\left(\sum_{j=1}^{J+1} p_{jk} f_j\left(\sum_{i=1}^{I+1} p_{ij} x_{id}\right)\right). \tag{7.5}$$

4) El error para el $d$-ésimo patrón, respecto a su etiqueta de pertenencia $t_d$, viene dado como:

$$e_d = \frac{1}{2}(t_d - s_{kd}^2)^2. \tag{7.6}$$

5) Para los $n$ pares patrones de entrenamiento, el error total (época a época) viene dado como:

$$e = \frac{1}{n}\sum_{d=1}^{n} e_d = \frac{1}{n}\sum_{d=1}^{n}\frac{1}{2}\left(t_d - s_{kd}^2\right)^2. \tag{7.7}$$

**Fase de propagación hacia atrás de los errores**. Durante esta fase, los errores obtenidos en cada una de las neuronas de salida son usados para modificar, primeramente, los pesos de interconexión entre dichas neuronas y las neuronas de la capa intermedia. Posteriormente, los errores en las neuronas intermedias son usados para modificar los pesos de interconexión entre las neuronas intermedias y las de entrada.

Para el $d$-ésimo patrón de entrenamiento:

1) Un cambio en el peso $p_{jk}$ entre la $j$-ésima neurona escondida y la $k$-ésima neurona de salida, viene dado como:

$$\triangle p_{jk} = -\alpha\frac{\partial e_d}{\partial p_{jk}} \tag{7.8}$$

donde, como ya vimos, $\alpha$ es la variable de entrenamiento para la red. Este gradiente sobre el error $e_d$ respecto al peso de interconexión $p_{jk}$, se puede expresar con base en la entrada $a_{kd}^2$ a la $k$-ésima neurona de salida a través de la regla de la cadena viene dado como:

$$\frac{\partial e_d}{\partial p_{jk}} = \frac{\partial e_d}{\partial a_{kd}^2}\frac{\partial a_{kd}^2}{\partial p_{jk}}. \tag{7.9}$$

El primer término de la ecuación anterior, como suele hacerse en la literatura, puede ser definido como el término delta $\delta$ respecto a la muestra $d$, esto es $\frac{\partial e_d}{\partial a^2_{kd}} = \delta_{kd}$.

Por otro lado, tomando en cuenta la ecuación (7.3): $\frac{\partial a^2_{kd}}{\partial a_{p_{jk}}} = s^1_{jd}$. Con base en este sencillo análisis, la ecuación (7.8) se puede expresar como:

$$\triangle p_{jkd} \quad = \quad = -\alpha \delta_{kd} s^1_{jd}. \tag{7.10}$$

Enseguida, se usará esta ecuación para actualizar los pesos para las neuronas de la capa de salida.

Después se verá cómo esta misma formulación se puede adaptar para el cálculo de los pesos de interconexión entre las neuronas de la capa de entrada y la capa intermedia.

2) El término Delta $\delta$ para la $k$-ésima neurona de la capa de salida a partir del error $e_d$ con respecto a la entrada $a^2_{kd}$ a dicha neurona, se puede expresar mediante la regla de la cadena como:

$$\delta_{kd} \quad = \quad \frac{\partial e_d}{\partial s^2_{kd}} \frac{\partial s^2_{kd}}{\partial a^2_{kd}}. \tag{7.11}$$

De la ecuación (7.6) $\frac{\partial e_d}{\partial s^2_{kd}} = -(t_d - s^2_{kd})$, mientras que $\frac{\partial s^2_{kd}}{\partial a^2_{kd}} = s^2_{kd}(1 - s_k d^2)$; en consecuencia:

$$\delta_{kd} \quad = \quad -(t_d - s^2_{kd}) s^2_{kd}(1 - s^2_{kd}). \tag{7.12}$$

De la ecuación (7.10), el cambio en los pesos entre la $j$-ésima neurona de la capa intermedia y la $k$-ésima neurona de la capa de salida con respecto al patrón $d$ vienen pues dados como:

$$\triangle p_{jk} \quad = \quad -\alpha \delta_{kd} s^1_{jd} = \alpha (t_d - s^2_{kd}) s^2_{kd}(1 - s^2_{kd}) s^1_{jd}. \tag{7.13}$$

El nuevo valor del peso $p_{jk}$ entre $j$-ésima neurona de la capa intermedia y la $k$-ésima neurona de la capa de salida viene dado por el valor anterior de dicho peso y el cambio de la ecuación (7.13) como sigue:

$$p^{\text{nuevo}}_{jk} \quad = \quad p^{\text{viejo}}_{jk} + \triangle p_{jk}. \tag{7.14}$$

3) El término Delta para el cálculo del peso de interconexión entre la $i$-ésima neurona de entrada y la $j$-ésima neurona intermedia puede obtenerse como sigue.

El gradiente del error con respecto al peso $p_{ij}$ se puede obtener como:

$$\frac{\partial e_d}{\partial p_{ij}} = \frac{\partial e_d}{\partial a^1_{jd}} \frac{\partial a^1_{jd}}{\partial p_{ij}}. \tag{7.15}$$

No es difícil darse cuenta de que el primer término de esta ecuación es la Delta para la $j$-ésima neurona intermedia:

$$\delta_{jd} = \frac{\partial e_d}{\partial a^1_{jd}}. \tag{7.16}$$

Por otro lado, el segundo término de la ecuación (7.15), al tomar en cuenta la ecuación (7.2), viene dado como:

$$\frac{\partial a^1_{jd}}{\partial p_{ij}} = x_{ij}. \tag{7.17}$$

Por lo tanto:

$$\frac{\partial e_d}{\partial p_{ij}} = \delta_{jd} x_{id}. \tag{7.18}$$

Como suele hacerse en la literatura:

$$\delta_{jd} = \frac{\partial e_d}{\partial s^1_{jd}} \frac{\partial s^1_{jd}}{\partial a^1_{jd}}. \tag{7.19}$$

El primer término de esta ecuación puede expresarse usando la regla de la cadena como $\frac{\partial e_d}{\partial a^3_{kd}} \frac{\partial a^2_{kd}}{\partial s^1_{jd}}$.

Debido a que todas las $K$ neuronas de la capa de salida influencian con su error propagado hacia atrás a la $j$-ésima neurona intermedia, entonces:

$$\frac{\partial e_d}{\partial s^1_{jd}} = \sum_{k=1}^{K} \frac{\partial e_d}{\partial a^2_{kd}} \frac{\partial a^2_{kd}}{\partial s^1_{jd}}. \tag{7.20}$$

De igual forma, debido a que $\frac{\partial e_d}{\partial a^2_{kd}} = \delta_{kd}$ y $\frac{\partial a^2_{kd}}{\partial s^1_{jd}} = p_{jk}$ como se indica en la ecuación (7.3).

En consecuencia:

$$\frac{\partial e_d}{\partial s^1_{jd}} = \sum_{k=1}^{K} \delta_{kd} p_{jk}. \tag{7.21}$$

Al sustituir esta última ecuación en la ecuación (7.19):

$$\delta_{jd} = \sum_{k=1}^{K} \delta_{kd} p_{jk} \frac{\partial s^1_{jd}}{\partial a^1_{jd}}. \tag{7.22}$$

Sabiendo que $\frac{\partial s^1_{jd}}{\partial a^1_{jd}} = s^1_{jd}(1 - s^1_{jd})$ y al reacomodar términos, se tiene que:

$$\delta_{jd} = s^1_{jd}(1 - s^1_{jd}) \sum_{k=1}^{K} \delta_{kd} p_{jk}. \tag{7.23}$$

Con todo esto:

$$\frac{\partial e_d}{\partial p^1_{ij}} = \left[ s^1_{jd}(1 - s^1_{jd}) \sum_{k=1}^{K} \delta_{kd} p_{jk} \right] x_{id}. \tag{7.24}$$

El cambio en los pesos entre las neuronas de la capa de entrada y la capa intermedia con respecto al patrón $d$ vienen dados como:

$$\triangle p_{ij} = -\alpha \delta_{jd} x_{id}. \tag{7.25}$$

Finalmente, el nuevo valor del peso $p_{ij}$ entre $i$-ésima neurona de entrada y la $j$-ésima neurona intermedia viene dado por el valor anterior de dicho peso y el cambio (7.25), viene dado como:

$$p_{ij}^{\text{nuevo}} = p_{ij}^{\text{viejo}} + \triangle p_{ij}. \tag{7.26}$$

La correspondiente RDG para una red neuronal con $I$ entradas, $J$ neuronas intermedias y $K$ neuronas de salida con funciones de transferencia sigmoidal viene dada como se indica en el pseudocódigo 7.1:

**Pseudocódigo 7.1**

## Regla de entrenamiento RDG para una RNA con neuronas sigmoidales

Poner los valores del vector de pesos $\boldsymbol{p}$ en valores aleatorios, por ejemplo, en el rango $\{-1,\ 1\}$.
Inicializar $\alpha$ en algún valor entre $0 < \alpha < 1$.

**repeat**

Poner la variable error en 0: $e = 0$;

**for** (Para cada par de entrenamiento $(\boldsymbol{x}^d,\ t_d),\ d = 1, n$)

Fase de propagación hacia adelante:
**for** (Para cada una de las $J$ neuronas de la capa intermedia)
Calcular sus activaciones mediante la ecuación (7.1).
Calcular sus salidas mediante la ecuación (7.2).
**end**

**for** (Para cada una de las $K$ neuronas de la capa de salida)
Calcular sus activaciones mediante la ecuación (7.3)
Calcular sus salidas mediante la ecuación (7.4)
**end**

Fase de propagación hacia atrás:
**for** (Para cada una de las $K$ neuronas de la capa de salida:)
Calcular sus correspondientes deltas $\delta_{kd}$ (7.12)
Calcular los cambios en los pesos $\triangle p_{jk} = -\alpha \delta_{kd} s^1_{jd}$ (7.13)
Ajustar los pesos como $p^{\text{nuevo}}_{ij} = p^{\text{viejo}}_{ij} + \triangle p_{ij}$ (7.14)
**end**

**for** (Para cada una de las $J$ neuronas de la capa intermedia)
Calcular sus correspondientes deltas $\delta_{jd}$ (7.23)
Calcular los cambios en los pesos $\triangle p_{ij} = -\alpha \delta_{jd} x^1_{id}$ (7.25)
Ajustar los pesos $p^{\text{nuevo}}_{ij} = p^{\text{viejo}}_{ij} + \triangle p_{ij}$ (7.26)
**end**

Obtener el error correspondiente $e_d = \frac{1}{2}(t_d - s^2_{kd})^2$ (7.6)
**end**

Obtener el error acumulado $e$: $e = \frac{1}{n}\sum_{d=1}^{n} e_d$ (7.7)

**until** $|\bigtriangledown e| \approx 0$ ;

Con el fin de facilitar la explicación de la operación del algoritmo anteriormente descrito, considérese el problema O-exclusivo.

**• • • Ejemplo 7.1**

Sea una red neuronal con una capa escondida con dos neuronas $N_1^1$ y $N_2^1$ y la capa de salida con una neurona $N_1^2$ como se muestra en la figura 7.3a. $a_1^1$, $a_2^1$ y $a_1^2$ son las correspondientes activaciones de las tres neuronas, ecuación (7.2) y $s_1^1$, $s_2^1$ y sus correspondientes salidas, ver la ecuación (7.2). Como pesos iniciales suponga los siguientes para $s_1^2$: $p_{11} = 1,\ p_{21} = -1,\ p_{31} = 1,\ p_{12} = -1,\ p_{22} = 1,\ p_{32} = -1,\ p_{13} = 1,\ p_{23} = -1,\ p_{33} = 1$. Además, suponga que $\alpha = 0.5$.

**Solución**

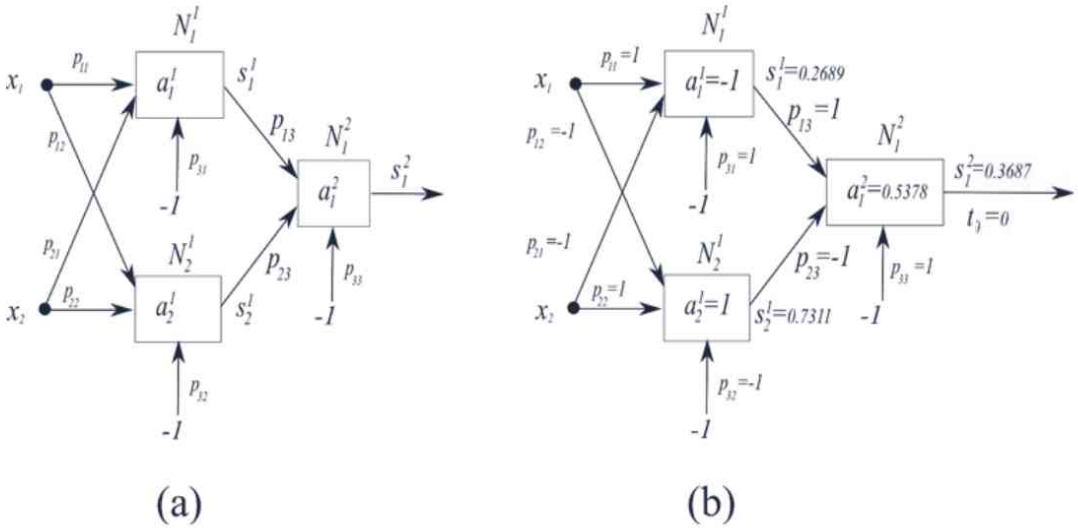



**Figura 7.3**: (a) Red neuronal con 2 neuronas en la capa escondida y una neurona en la capa de salida para el ejemplo 7.1. (b) Red neuronal con valores iniciales de sus pesos, así como las correspondientes activaciones y salidas de las tres neuronas.

De acuerdo con la visto anteriormente, el conjunto de pares de entrenamiento para el problema O-exclusivo como a continuación se indica: $\boldsymbol{x}_1 = (0,\ 0)^T,\ t_1 = 0\ \boldsymbol{x}_2 = (0,\ 1)^T,\ t_2 = 1,\ \boldsymbol{x}_3 = (1,\ 0)^T,\ t_3 = 1$ y $\boldsymbol{x}_4 = (1,\ 1)^T,\ t_4 = 0$.

Para este conjunto de vectores de entrenamiento, suponga que el conjunto de vectores aumentados es el siguiente: $\boldsymbol{x}_1 = (0,\ 0,\ -1)^T, \boldsymbol{x}_2 = (0,\ 1,\ -1)^T,\ \boldsymbol{x}_3 = (1,\ 0, 1)^T$ y $\boldsymbol{x}_4 = (1,\ 1, 1)^T$.

Para fines de comprensión de la operación de los diferentes pasos del algoritmo RDG, considérese únicamente el primer recorrido (primera época) del conjunto extendido anterior. De acuerdo con el algoritmo, tenemos que: $e = 0$. Para el primer par de entrenamiento: $\boldsymbol{x}_1 = (0,\ 0)^T,\ t_1 = 0$:

**Fase de propagación hacia adelante**:
**Neurona** $N_1^1$: Las entradas para esta neurona se muestran en las columnas 1, 2 y 3 de la tabla 7.1:

**Tabla 7.1**: Fase de propagación hacia adelante: **Neurona** $N_1^1$.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $p_{11}$ | $p_{21}$ | $p_{31}$ | $a_1^1$ | $s_1^1$ |
| 0 | 0 | -1 | 1 | -1 | 1 | -1 | 0.2689 |

Los correspondientes pesos, de acuerdo con lo especificado y la figura 7.3a, aparecen en las columnas 4, 5 y 6 de la misma tabla. La séptima columna muestra la activación correspondiente a dicha neurona: $a_1^1 = 1(0) + (-1)(0) + 1(-1) = -1$, de acuerdo con la ecuación (7.1); mientras que, la octava columna muestra el valor de la salida de la neurona: $s_1^1 = (1 + e^{-(-1)})^{-1} = 0.2689$, ecuación (7.2).

**Neurona** $N_2^1$: Las entradas para esta neurona (figura 7.3a) son las mismas que para la neurona $N_1^1$ y se muestran en las primeras tres columnas de la tabla 7.2:

**Tabla 7.2**: Fase de propagación hacia adelante: **Neurona** $N_2^1$.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $p_{12}$ | $p_{22}$ | $p_{32}$ | $a_2^1$ | $s_2^1$ |
| 0 | 0 | -1 | -1 | 1 | -1 | 1 | 0.7311 |

Los correspondientes pesos, de acuerdo con lo especificado y la figura 7.3a, aparecen en la cuarta, quinta y sexta columna de la tabla 7.2. La séptima columna muestra la activación correspondiente a dicha neurona: $a_2^1 = -1(0)+1(0)+(-1)(-1) = 1$ ecuación (7.1), mientras que la octava columna muestra el valor de la salida de la neurona: $s_2^1 = (1+e^{-1})^{-1} = 0.7311$, ecuación (7.2).

**Neurona** $N_1^2$: De acuerdo con la figura 7.3a, las entradas para esta neurona son las salidas de las neuronas $N_1^1$ y $N_2^1$; estas entradas se muestran en las primeras tres columnas de la tabla 7.3 (la tercera se corresponde con el bias a $N_1^2$):

**Tabla 7.3**: Fase de propagación hacia adelante: **Neurona** $N_1^2$.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| $s_{1d}^1$ | $s_{2d}^1$ | $x_3$ | $p_{13}$ | $p_{23}$ | $p_{33}$ | $a_1^2$ | $s_1^2$ |
| 0.2689 | 0.7311 | -1 | 1 | -1 | 1 | -1.4622 | 0.1881 |

Los correspondientes pesos, de acuerdo con la figura 7.3a, aparecen en la cuarta, quinta y sexta columna de la tabla anterior. La séptima columna muestra la activación correspondiente a dicha neurona: $a_1^2 = 1(0.2689) + (-1)(0.7311) + 1(-1) = -1.4622$, ecuación (7.3), mientras que la octava columna muestra el valor de la salida de la neurona: $s_1^2 = (1 + e^{-1})^{-1} = 0.1881$, ecuación (7.5). La figura 7.3b resume los resultados obtenidos hasta el momento.

**Fase de propagación hacia atrás**:
Delta para la neurona de salida $N_1^2$. Tomando en cuenta los resultados obtenidos anteriormente y de acuerdo con la ecuación (7.12):

$$\delta_{k=1,d} = -(t_d - s_{1d}^2)s_{1d}^2(1 - s_{1d}^2) = -(0 - 0.1881) \times (0.1881) \times (1 - 0.1881) = 0.02874.$$

El cambio en cada peso de la neurona $N_1^2$ es calculado de acuerdo con la ecuación (7.13) como sigue:

$$\begin{aligned} \triangle p_{13} &= -\alpha s_{1d}^1 \, \delta_{k=1,d} - 0.5 \times (0.2689) \times (0.02874) = -0.00386 \\ \triangle p_{23} &= -\alpha s_{2d}^1 \, \delta_{k=1,d} = -0.5 \times (0.7311) \times 0.02874 = -0.01050 \\ \triangle p_{33} &= -\alpha x_3 \delta_{k=1,d} = -0.5 \times (-1) \times 0.02874 = 0.01437. \end{aligned}$$

Asimismo el ajuste en cada peso de $N_1^2$ se calcula mediante la ecuación (7.14) como:

$$\begin{aligned} p_{13}^{\text{nuevo}} &= p_{13} + \triangle p_{13} = 1 - 0.00386 = 0.9961 \\ p_{23}^{\text{nuevo}} &= p_{23} + \triangle p_{23} = -1 - 0.01050 = -1.0105 \\ p_{33}^{\text{nuevo}} &= p_{33} + \triangle p_{33} = 1 + 0.01437 = 1.0143. \end{aligned}$$

Delta para la neurona intermedia $N_1^1$. De acuerdo con la ecuación (7.12):

$$\delta_{j=1,d} = s_{1d}^1(1-s_{1d}^1) \quad \delta_{k=1,d}\, p_{13}^{\text{nuevo}} = 0.2689 \times (1-0.2689) \times 0.02874 \times (0.9961) = 0.00565.$$

El cambio en cada peso de la neurona $N_1^1$ es calculado de acuerdo con la ecuación (7.13), como sigue:

$$\begin{aligned}
\triangle p_{11} &= -\alpha\delta_{j=1,d}x_{1d} = -0.5(0.00565)(0) = 0.0 \\
\triangle p_{21} &= -\alpha\delta_{j=1,d}x_{2d} = -0.5(0.00565)(0) = 0.0 \\
\triangle p_{31} &= -\alpha\delta_{j=1,d}x_{3d} = -0.5(0.00565)(-1) = 0.002825.
\end{aligned}$$

Asimismo el ajuste para cada peso de $N_1^1$ se calcula mediante la ecuación (7.14) como:

$$\begin{aligned}
p_{11}^{\text{nuevo}} &= p_1 1 + \triangle p_{11} = 1.0 + 0.0 = 1.0 \\
p_{21}^{\text{nuevo}} &= p_2 1 + \triangle p_{21} = -1.0 + 0.0 = -1.0 \\
p_{31}^{\text{nuevo}} &= p_3 1 + \triangle p_{31} = 1.0 + 0.002825 = 1.002825.
\end{aligned}$$

Delta para la neurona intermedia $N_2^1$. De acuerdo con la ecuación (7.23):

$$\delta_{j=2,d} = s_{2d}^1(1-s_{2d}^1)\delta_{k=1,d}\, p_{23}^{\text{nuevo}} = 0.7311 \times (1-0.7311) \times (0.02874) \times (-1.0105) = -0.00565.$$

El cambio en cada peso de la neurona $N_2^1$ es calculado de acuerdo con la ecuación (7.25), como sigue:

$$\begin{aligned}
\triangle p_{12} &= -\alpha\ \delta_{j=2,d}\ x_{1d} = -0.5 \times (-0.00565) \times (0) = 0.0 \\
\triangle p_{22} &= -\alpha\ \delta_{j=2,d}\ x_{2d} = -0.5 \times (-0.00565) \times (0) = 0.0 \\
\triangle p_{32} &= -\alpha\ \delta_{j=2,d}\ x_{3d} = -0.5 \times (-0.00565) \times (-1) = -0.002825.
\end{aligned}$$

Asimismo el ajuste en cada peso de $N_2^1$ se calcula mediante la ecuación (7.26) como:

$$\begin{aligned}
p_{12}^{\text{nuevo}} &= p_{12} + \triangle p_{12} = -1.0 + 0.0 = -1.0 \\
p_{22}^{\text{nuevo}} &= p_{22} + \triangle p_{22} = 1.0 + 0.0 = 1.0 \\
p_3 2^{\text{nuevo}} &= p_{32} + \triangle p_{32} = -1.0 - 0.0087 = -1.002825.
\end{aligned}$$

El error para este primer patrón vale: $e_{d=1} = 0.5(t_{d=1} - s_{1d=1}^2)^2 = (0 - 0.1881)^2 = 0.01769$. La primera fila de la tabla 7.4 muestra los valores de los nueve pesos de la red entrenada con el primer patrón $\boldsymbol{x}_d = (0,\ 1,\ -1)^T,\ t_d = 0$.

**Tabla 7.4**: Valores de los pesos de la red del ejemplo 7.1 para cada una de las asociaciones.

| $\boldsymbol{x}_d$ | $t_d$ | $p_{11}^{\text{nuevo}}$ | $p_{21}^{\text{nuevo}}$ | $p_{31}^{\text{nuevo}}$ | $p_{12}^{\text{nuevo}}$ | $p_{22}^{\text{nuevo}}$ | $p_{32}^{\text{nuevo}}$ | $p_{13}^{\text{nuevo}}$ | $p_{23}^{\text{nuevo}}$ | $p_{33}^{\text{nuevo}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\begin{bmatrix}0\\0\\-1\end{bmatrix}$ | 0 | 1 | -1 | 1.00282 | -1 | 1 | -1.00282 | 0.9961 | -1.0105 | 1.0143 |
| $\begin{bmatrix}0\\1\\-1\end{bmatrix}$ | 1 | 1 | -0.994 | 0.9973 | -1 | 0.994 | 0.9972 | 1.0024 | -0.9641 | 0.9617 |
| $\begin{bmatrix}1\\0\\-1\end{bmatrix}$ | 1 | 1.018 | -0.994 | 0.979 | -1.017 | 0.994 | -0.979 | 1.038 | -0.927 | 0.889 |
| $\begin{bmatrix}1\\1\\-1\end{bmatrix}$ | 0 | 1.014 | -0.998 | 0.983 | -1.014 | 0.997 | -0.983 | 1.033 | -0.941 | 0.907 |

De continuar así, entonces el resto de los tres patrones tienen los siguientes valores: $\boldsymbol{x}_2 = (0,\ 1,\ -1)^T$, $t_2 = 1$, $\boldsymbol{x}_3 = (1,\ 0,1)^T$, $t_3 = 1$ y $\boldsymbol{x}_4 = (1,\ 1,1)^T$, $t_4 = 0$, las filas segunda, tercera y cuarta de la tabla 7.4 resumen los valores de los pesos actualizados para cada patrón.

Los correspondientes errores para los patrones segundo, tercero y cuarto son los siguientes:

$$\begin{aligned} e_{d=2} &= 0.5(t_{d=2} - s_{1d=2}^2)^2 = (1 - 0.1471)^2 = 0.3667 \\ e_{d=3} &= 0.5(t_{d=3} - s_{1d=3}^2)^2 = (1 - 0.2653)^2 = 0.2587 \\ e_{d=4} &= 0.5(t_{d=4} - s_{1d=4}^2)^2 = (0 - 0.1835)^2 = 0.0240. \end{aligned}$$

El error acumulado para esta primera época de acuerdo con la ecuación 4.64b, vale:

$$e_1 = \frac{1}{n}\sum_{d=1}^{n} e_d = \frac{1}{4}(0.01769 + 0.3667 + 0.2587 + 0.0240) = 0.1668.$$

**•• Ejemplo 7.2**

Aplicar el algoritmo RDG al problema O-exclusivo usando la red neuronal mostrada en la figura 7.3a, con los mismos pesos iniciales y parámetro de aprendizaje que para el ejemplo 7.1. En este caso aplicar el algoritmo RDG durante 1500 épocas.

**Solución**

Del ejemplo 7.1, el conjunto de pares de entrenamiento extendidos con sus etiquetas (con valores 1) es el siguiente:

$$\boldsymbol{x}_1 = (0,\ 0,\ 1)^T,\ t_1 = 0,$$

$$\begin{aligned} \boldsymbol{x}_2 &= (0,\ 1,\ 1)^T,\ t_2 = 1 \\ \boldsymbol{x}_3 &= (1,\ 0,\ 1)^T,\ t_3 = 1 \\ \boldsymbol{x}_4 &= (1,\ 1,\ 1)^T,\ t_4 = 0. \end{aligned}$$

Al iniciar y correr el algoritmo RGD con el conjunto de pesos: $p_{11} = 1$, $p_{21} = -1$, $p_{31} = 1$, $p_{12} = -1$, $p_{22} = 1$, $p_{32} = -1$, $p_{13} = 1$, $p_{23} = -1$, $p_{33} = 1$ y $\alpha = 0.5$.

El lector puede demostrar que después de 1500 iteraciones, el error calculado toma el siguiente valor: $e_{1500} = 0.002277$.

Al aplicar el algoritmo RDG con los pesos iniciales y el $\alpha$, escogidos, el lector puede demostrar que después de 1500 iteraciones dicho algoritmo converge a los siguientes pesos finales:

$$\begin{aligned} \text{Neurona } N_1^1 :\ p_{11} &= 5.9789,\ p_{21} = -5.9956,\ p_{31} = 3.1813 \\ \text{Neurona } N_2^1 :\ p_{12} &= -4.4399,\ p_{22} = 4.7874,\ p_{32} = 2.2021 \\ \text{Neurona } N_1^2 :\ p_{13} &= -5.2988,\ p_{23} = -5.4481,\ p_{33} = 7.8381. \end{aligned}$$

### • • • Ejemplo 7.3

Calcular la salida de la red neuronal evolucionada en el ejemplo 7.2 con los valores de los cuatro patrones de entrenamiento para verificar si el valor de salida correspondiente se acerca al valor deseado.

### Solución

En la tabla 7.5 se muestran las salidas calculadas mediante la red evolucionada en el ejemplo 7.2. Nótese cómo dichas salidas calculadas tienden a los valores deseados. El lector puede demostrar que si crece el número de épocas en el entrenamiento de la red, la nueva red podría producir salidas más cercanas a los valores deseados.

Nótese como se esperaba, los valores de salida de la neurona de salida para los patrones $\boldsymbol{x}_1 = (0\ 0)^T$ y $\boldsymbol{x}_4 = (1\ 1)^T$ se aproximan a cero (segunda y quinta fila de la tabla 7.5 ), mientras que los valores de la misma neurona para los patrones $\boldsymbol{x}_2 = (0\ 1)^T$ y $\boldsymbol{x}_3 = (1\ 0)^T$ se acercan a 1.0 (tercera y cuarta fila de la misma tabla 7.5).

**Tabla 7.5**: Valores calculados a la salida de la red evolucionada en el ejemplo 7.3.

| $\boldsymbol{x}_d$ | $t_d$ | Valores calculados |
|---|---|---|
| $\begin{bmatrix}0\\0\\1\end{bmatrix}$ | 0 | 0.104 |
| $\begin{bmatrix}0\\1\\1\end{bmatrix}$ | 1 | 0.890 |
| $\begin{bmatrix}1\\0\\1\end{bmatrix}$ | 1 | 0.882 |
| $\begin{bmatrix}1\\1\\1\end{bmatrix}$ | 0 | 0.091 |

### 7.2.3 Función de la capa intermedia

En esta subsección vamos a estudiar varias propiedades interesantes ligadas al uso de una capa intermedia en una RNA. La primera es que una capa intermedia puede descubrir la estructura no lineal del problema. Enseguida vamos a ver esto con más detalle. La segunda, muy ligada a la primera es que si a la salida se decidiera usar una neurona de tipo lineal en lugar de una no lineal, como es el caso del ejemplo 7.2, la RNA continuará proporcionando el mapeo deseado al transformar (a través de la capa intermedia) el conjunto de datos de entrada (no linealmente separados en el caso del problema "O-exclusivo"), en uno linealmente separable. De esta forma, la neurona de salida lineal será capaz de resolver el problema en cuestión.

Para mostrar lo antes dicho, consideremos los pesos de las dos neuronas $n_1^1$ y $n_2^1$ de la capa intermedia de la RNA entrenada del ejemplo 7.2:

$$\begin{aligned}\textbf{Neurona } n_1^1: \ p_{11} &= 5.9789,\ p_{21} = -5.9956,\ p_{31} = 3.1813\\ \textbf{Neurona } n_2^1: \ p_{12} &= -4.4349,\ p_{22} = 4.7874,\ p_{32} = 2.2021.\end{aligned}$$

Para los cuatro patrones de entrada: $\boldsymbol{x}_1 = (0\ 0\ 1)^T$, $\boldsymbol{x}_2 = (0\ 1\ 1)^T$, $\boldsymbol{x}_3 = (1\ 0\ 1)^T$, $\boldsymbol{x}_4 = (1\ 1\ 1)^T$, la tabla 7.6 muestra las correspondientes salidas $s_1^1$ y $s_2^1$ de las neuronas $n_1^1$ y $n_2^1$ (ver figura 7.3):

**Tabla 7.6**: Salidas $s_1^1$ y $s_2^1$ de las neuronas $n_1^1$ y $n_2^1$ de la figura 7.3.

| $\boldsymbol{x}$ | $a_1^1$ | $s_1^1$ | $a_2^1$ | $s_2^1$ |
|---|---|---|---|---|
| $\begin{bmatrix}0\\0\\1\end{bmatrix}$ | 3.1813 | 0.96 | 2.2021 | 0.90 |
| $\begin{bmatrix}0\\1\\1\end{bmatrix}$ | -2.8143 | 0.05 | 6.9895 | 0.99 |
| $\begin{bmatrix}1\\0\\1\end{bmatrix}$ | 9.1602 | 0.99 | -2.2328 | 0.09 |
| $\begin{bmatrix}1\\1\\1\end{bmatrix}$ | 3.1646 | 0.96 | 2.5546 | 0.93 |

En la figura 7.4 se muestran los cuatro patrones mapeados al espacio de salida: $s_1^1 - s_2^1$ mediante la capa intermedia formada por las neuronas $n_1^1$ y $n_2^1$.

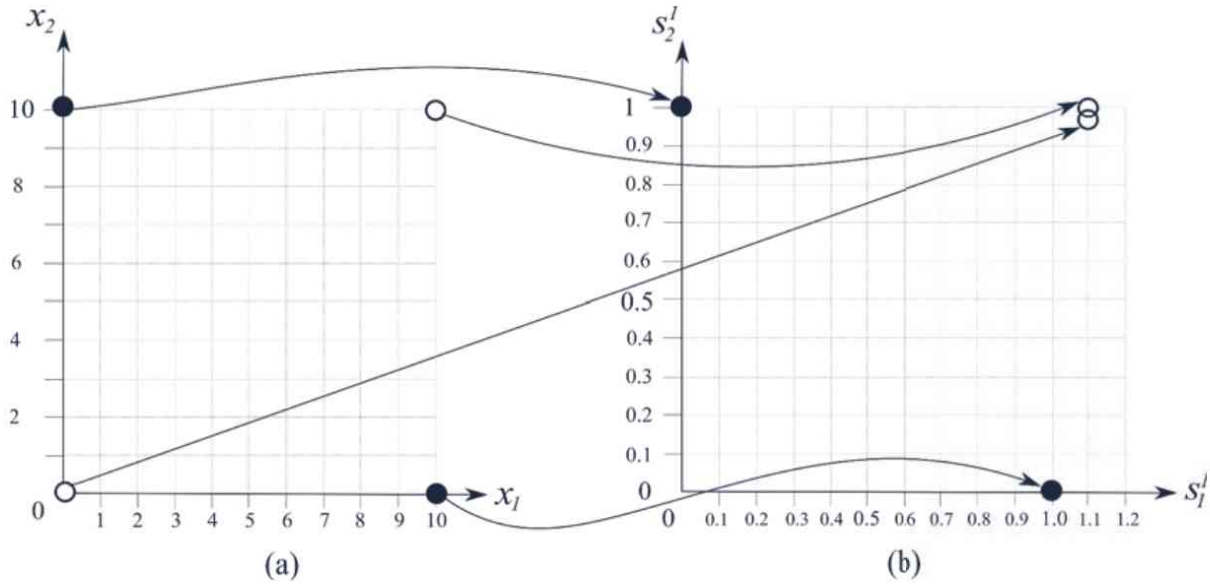



**Figura 7.4**: (a) En el espacio de entrada $x_1 - x_2$, los patrones: $\boldsymbol{x}_1 = (0\ 0)^T$, $\boldsymbol{x}_2 = (0\ 1)^T$, $\boldsymbol{x}_3 = (1\ 0)$, $\boldsymbol{x}_4 = (1\ 1)^T$ aparecen linealmente no separados. (b) En el espacio $s_1^1 - s_2^1$ los mismos patrones aparecen ahora linealmente separados, lo cual permite utilizar en la capa de salida una neurona de tipo lineal.

Note cómo los cuatro patrones linealmente no separados en el espacio de entrada $x_1 - x_2$ (figura 7.4a), en el espacio $s_1^1 - s_2^1$ aparecen ahora linealmente separados (figura 7.4b): Los patrones $(0\ 0)^T$ y $(1\ 1)^T$ son mapeados una posición cerca del uno (referirse a las flechas correspondientes), mientras que los patrones $(0\ 1)^T$ y $(1\ 0)^T$ son mapeados a posiciones similares $(0.05\ 0.99)^T$ y $(0.99\ 0.09)^T$ (ver flechas) en el espacio de salida $s_1^1 - s_2^1$.

Como ya se dijo, la RNA fue capaz de descubrir la estructura no lineal del problema (O-exclusivo en este caso), transformándolo en uno lineal. Con esto se puede ver que la neurona $n_1^2$ puede ser reemplazada por una neurona lineal más simple que la sigmoidal. Se deja al lector demostrar este hecho.

De este ejemplo se puede extraer la siguiente lección: Mediante una capa intermedia de unidades de procesamiento no lineal es posible aproximar cualquier mapeo entre la capa de entrada de la RNA y su capa de salida. Este resultado proviene del famoso teorema de Cybenco [26].

Aunque el teorema de Cybenco es una herramienta de diseño general de redes neuronales artificiales, no ayuda a establecer el número de neuronas requerido para encontrar un mapeo ni tampoco el tiempo requerido para alcanzar esta meta.

Por simple observación, el lector puede apreciar que para el caso del problema "O-exclusivo" no es difícil por prueba y error determinar que con dos neuronas en la capa intermedia es suficiente para encontrar el mapeo correspondiente entre los cuatro patrones de entrada y sus respectivas etiquetas; referirse a los ejemplos 6.9 y 6.10 del capítulo 6.

En lo general, para cualquier problema complejo, sobre todo con más de dos variables, el número deseado de neuronas en la capa intermedia no es conocido. Este número de unidades tendrá que ser encontrado por prueba y error.

### 7.2.4 Aceleramiento del proceso de entrenamiento

En la literatura especializada, varios métodos para acelerar el proceso de entrenamiento de una red neuronal han sido propuestos; referirse, por ejemplo, a [32], [41] y [47]. En esta sección se estudia uno de ellos. Este método se fundamenta en el uso de un parámetro que toma en cuenta información usada en la iteración anterior en el entrenamiento de la red neuronal. A este término se le conoce como momento.

Se basa en una heurística muy sencilla: Si una serie de pasos consecutivos $\triangle \boldsymbol{p}$ apuntan más o menos en la misma dirección, entonces probablemente el mínimo de $e$ se encuentra en esa misma dirección. Conviene, entonces, dar pasos más grandes en esa dirección.

Tomando en cuenta este hecho, el ajuste al vector de pesos $\boldsymbol{p}$ queda ahora dado como una combinación lineal entre el ajuste dado por la regla Delta generalizada y el ajuste dado por el llamado factor de momento $\lambda$, como sigue:

$$\triangle \boldsymbol{p} \quad = \quad \triangle \boldsymbol{p} + \lambda \triangle \boldsymbol{p}(n-1) \tag{7.27}$$

con $\alpha$, como ya vimos el parámetro taza de entrenamiento y, $\lambda$, el parámetro que confiere el momento deseado al sistema.

El vector de pesos $\boldsymbol{p}$ es finalmente ajustado como:

$$\boldsymbol{p}(n+1) \quad = \quad \boldsymbol{p}(n) + \triangle \boldsymbol{p} + \lambda \triangle \boldsymbol{p}(n-1). \tag{7.28}$$

La adición de momento al proceso de entrenamiento favorece un ajuste acelerado de los pesos de la red neuronal. Metafóricamente hablando, cuando el camino hacia la solución deseada, en cuanto a los pesos se refiere, se encuentra en un terreno plano o con cambios, el factor momento coadyuva a la aceleración. Por el contrario, cuando el camino hacia la solución se encuentra lleno de sinuosidades, el término momento tiende a desacelerar el ir acercándose a la solución.

Otra ventaja que se ha encontrado respecto al término de momento es que este también sirve para ayudar a la regla RGD a librar mínimos locales no tan pronunciados, contribuyendo a encontrar mejores soluciones a los pesos deseados.

### • • • Ejemplo 7.4

Para ilustrar el uso del término momento añadido a la regla de actualización de los pesos, considérese la misma configuración de red del ejemplo 7.2, esto es con $p_{11} = 1$, $p_{12} = -1$, $p_{31} = 1$, $p_{21} = -1$, $p_{22} = 1$, $p_{32} = -1$, $p_{13} = 1$, $p_{23} = -1$, $p_{33} = 1$ y $\alpha = 0.5$.

### Solución

Al usar el algoritmo RGD incorporando el factor momento con un $\lambda = 0.5$, el lector puede demostrar que en 800 épocas se obtiene un mejor conjunto de pesos:

$$\textbf{Neurona } n_1^1: \; p_{11} = 6.0296, \; p_{21} = -6.0605, \; p_{31} = 3.2130$$

$$\textbf{Neurona } n_2^1: \; p_{12} = -4.5133, \; p_{22} = 4.8364, \; p_{32} = 2.22663$$

$$\textbf{Neurona } n_1^2: \; p_{13} = -5.3738, \; p_{23} = -5.5346, \; p_{33} = 7.9656.$$

Para los cuatro patrones: $\boldsymbol{x}_1 = (0\ 0\ 1)^T$, $\boldsymbol{x}_2 = (0\ 1\ 1)^T$, $\boldsymbol{x}_3 = (1\ 0\ 1)^T$ y $\boldsymbol{x}_4 = (1\ 1\ 1)^T$, los cuatro valores de salida vienen dados por la tabla 7.7. Note cómo cada uno de los valores se acerca más al valor deseado $t_d$.

**Tabla 7.7**: Valores calculados a la salida de la red evolucionada en el ejemplo 7.4.

| $\boldsymbol{x}_d$ | $t_d$ | Valores calculados |
|---|---|---|
| $\begin{bmatrix}0\\0\\1\end{bmatrix}$ | 0 | 0.099 |
| $\begin{bmatrix}0\\1\\1\end{bmatrix}$ | 1 | 0.895 |
| $\begin{bmatrix}1\\0\\1\end{bmatrix}$ | 1 | 0.888 |
| $\begin{bmatrix}1\\1\\1\end{bmatrix}$ | 0 | 0.088 |

## 7.3 Redes neuronales convolucionales

Una red neuronal convolucional (RNC) [Web56] y [Web58] es un tipo particular de red neuronal cuyo proceso se basa en la funcionalidad de la conocida operación de la convolución [Web65]. En lo básico, una RNC combina el poder de un extractor automático de rasgos y la capacidad de clasificación de un perceptrón multicapa.

En esta sección se explicarán algunos detalles relacionados con la operación de esta potente máquina útil; en lo particular, en la detección, identificación y seguimiento de objetos en imágenes y secuencias de vídeo.

De acuerdo con lo visto, la convolución entre dos vectores: $\boldsymbol{p} = (p_1\ p_2\ \cdots\ p_n)^T$ y el vector $\boldsymbol{x} = (x_1\ x_2\ \cdots\ x_n)^T$, da como resultado un escalar $c = \boldsymbol{p}^T\boldsymbol{x}$, es decir: $c = \boldsymbol{p}^T\boldsymbol{x} = \boldsymbol{x} = p_1x_1+p_2x_2+\cdots+p_nx_n = \sum_{i=1}^{n} p_ix_i$. Este escalar para una RCN cumple con una función muy particular, abstraer mediante un número una característica o propiedad útil, para detectar o identificar un patrón específico en una señal o imagen.

Una RNC aplica un conjunto de estos convolucionadores sobre una imagen de entrada buscando encontrar en dicha imagen rasgos importantes que le permitan formar un vector descriptor; a su vez, este será usado por el clasificador, usualmente uno de tipo MLP, conectado al conjunto de convolucionadores para la tarea especificada.

Operacionalmente, una RNC funciona en cuatro etapas como sigue:

1) **Etapa de convolución**. Sobre la imagen de entrada $I$, la RNC aplica un conjunto de $m$ convolucionadores, dando como resultado tantas imágenes $J_1, J_2, \cdots, J_m$, como convolucionadores sean utilizados. Si estos convolucionadores fueron bien diseñados, estas imágenes, como ya se dijo, deberán contener rasgos a diversos niveles de abstracción de los objetos de interés, según la tarea a ejecutar.
2) **Etapa de reducción de datos**. Sobre cada una de las imágenes $J_1, J_2, \cdots, J_m$, la RNC aplica una etapa de agrupadores de valores para producir un conjunto de imágenes de menos tamaño: $K_1, K_2, \cdots, K_m$. Estas imágenes contienen la información más relevante de las imágenes $J_1, J_2, \cdots, J_m$.
3) **Etapa de formación del vector descriptor**. Con base en las imágenes $K_1, K_2, \cdots, K_m$, la RNC produce un vector descriptor $\boldsymbol{v}$. Este vector describe al objeto de interés en la imagen como un conjunto de números.
4) **Etapa de clasificación**. Durante esta etapa, la RNC recibe como entrada el vector $\boldsymbol{v}$ y emite como resultado la etiqueta de pertenencia del objeto presente en la imagen de entrada $I$.

En muchas aplicaciones, múltiples capas de convolución y de reducción, dependiendo del nivel de abstracción en los rasgos obtenidos y del nivel de reducción deseados, pueden ser utilizadas.

**•• Ejemplo 7.5**

Para ilustrar la operación básica de una RNC, suponga que sobre una imagen de 5×5 elementos se quiere determinar la presencia de un borde vertical o de un borde horizontal, como se ilustra en las figuras 7.5a y 7.5b. Sean estas imágenes $I_1$ e $I_2$.

**Solución**

**Etapa de convolución**. Para esta tarea tan sencilla, suponga que de alguna manera, que más adelante se explicará con más detalle, se lograron derivar (evolucionar) los dos siguientes convolucionadores:

$$C_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{7.29}$$

$$C_2 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \tag{7.30}$$

El lector podrá observar que estos dos operadores se corresponden con los respectivos de Prewit [42].

No es difícil ver que un convolucionador es un tipo especial de perceptrón, ya que ejecuta sobre un vector de entrada, un producto interno con su correspondiente vector de pesos. En el caso de una RNA, como función de activación a la salida del convolucionador, suele usarse una función tipo ReLU que tiende a poner en 0 los valores inferiores a 0 y dejar intactos los valores superiores a 0 [25], [29] y [30].

La respuesta a la salida de la $i$-ésima neurona convolutiva viene dada como:

$$s = f(c_i) = \max(0, c_i). \tag{7.31}$$

Al desplazar los dos convolucionadores $C_1$ y $C_2$, dados por las ecuaciones 7.29 y 7.30, sobre las imágenes 7.5a y 7.5b, el lector puede demostrar que se obtienen los resultados mostrados en las figuras 7.5c y 7.5d.

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

a) $I_1$

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

b) $I_2$

| | | |
|---|---|---|
| 2 | 0 | -2 |
| 3 | 0 | -3 |
| 2 | 0 | -2 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

c)

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

| | | |
|---|---|---|
| 2 | 3 | 2 |
| 0 | 0 | 0 |
| -2 | -3 | -2 |

d)

| | | |
|---|---|---|
| 2 | 0 | 0 |
| 3 | 0 | 0 |
| 2 | 0 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

e)

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |

| | | |
|---|---|---|
| 2 | 3 | 2 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

f)

| | |
|---|---|
| 3 | 0 |
| 3 | 0 |

| | |
|---|---|
| 1 | 1 |
| 0 | 0 |

g)

| | |
|---|---|
| 1 | 0 |
| 1 | 0 |

| | |
|---|---|
| 3 | 3 |
| 0 | 0 |

h)

**Figura 7.5**: (a) y (b) Imágenes de entrada $I_1$ y $I_2$. (c) y (d) Imágenes después de la aplicación de los dos convolucionadores (7.29) y (7.30). (e) y (f) Imágenes $J_{11}, J_{12}$ y $J_{21}, J_{22}$ después de la aplicación de ecuación (7.31) sobre las imágenes mostradas en las figuras (c) y (d). (g) y (h) Imágenes $K_{11}, K_{12}$ y $K_{21}, K_{22}$ después de la aplicación de un reductor de información de $2\times2$ sobre las imágenes $J_{11}, J_{12}$ y $J_{21}, J_{22}$.

Estos números, para la RNC codifican la presencia de un borde vertical (imagen $I_1$) o uno horizontal (imagen $I_2$). Así, por ejemplo, el primer valor arriba a la izquierda (el 2) de la primera máscara de la figura 7.5c, se obtuvo como sigue:

$$c_1 \quad = \quad -1(0) + 0(0) + 1(0) - 1(0) + 0(0) + 1(1) - 1(0) + 0(0) + 1(1) = 2.$$

Al aplicar la operación ReLU, ecuación (7.31) sobre los nueve valores de cada convolucionador, el lector puede verificar que se obtienen los resultados mostrados en las figuras 7.5e y 7.5f. En la literatura especializada se demuestra que para fines prácticos los valores negativos no tienden a aportar algo útil al problema de la detección o la clasificación.

**Etapa de reducción de datos**. En la literatura especializada se ha demostrado que la información contenida en una imagen puede ser reducida en su tamaño sin afectar la eficiencia final del clasificador.

La reducción de la información producida por los convolucionadores se puede conseguir mediante la aplicación de la conocida operación de agrupación máxima (del inglés *max pooling*). Varias formas de agrupamiento de información han sido propuestas en la literatura. Una de ellas considera un conjunto de valores sobre una ventana y selecciona el valor más alto. Si el número de valores a agrupar es por ejemplo 4, a través de una ventana de 2×2, desplazada de posición a posición sobre las imágenes de 3×3 mostradas en las figuras 7.5e y 7.5f, el lector puede demostrar que se obtienen los resultados de las figuras 7.5g y 7.5h.

Así por ejemplo, el primer valor arriba a la izquierda (el 3) de la primera máscara de la figura 7.5g, se obtuvo como sigue: max(2,0,3,0)=3.

**Etapa de formación del vector descriptor**. El vector descriptor de cada imagen se obtiene al tomar las últimas matrices, en este caso, las derivadas del proceso de agrupación máxima, convertirlas a vectores y concatenarlas; de ello resulta un solo vector. Para nuestro ejemplo, las primeras dos matrices de la figura 7.5g quedarían vectorizadas como sigue:

$$\begin{pmatrix} 3 & 0 \\ 3 & 0 \end{pmatrix} \longrightarrow (3 \quad 0 \quad 3 \quad 0) \text{ y } \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \longrightarrow (1 \quad 1 \quad 0 \quad 0).$$

El vector descriptor correspondiente quedaría como sigue al concatenar los dos vectores anteriores:

$$\boldsymbol{v}_1 = (3 \quad 0 \quad 3 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0)^T.$$

De igual forma, para las dos imágenes de la figura 7.5h, se tendrían los siguientes resultados:

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \longrightarrow (1 \quad 0 \quad 1 \quad 0) \text{ y } \begin{pmatrix} 3 & 3 \\ 0 & 0 \end{pmatrix} \longrightarrow (3 \quad 3 \quad 0 \quad 0).$$

El vector descriptor $\boldsymbol{v}_2$ concatenado correspondiente quedaría como sigue:

$$\boldsymbol{v}_2 = (1 \quad 0 \quad 1 \quad 0 \quad 3 \quad 3 \quad 0 \quad 0)^T.$$

**Etapa de clasificación**. Durante esta etapa se usan los vectores $\boldsymbol{v}_1$ y $\boldsymbol{v}_2$ para entrenar el clasificador. En este caso particular, se parte del hecho de que los dos patrones pueden ser separados linealmente. Con esta suposición en mente, el clasificador estaría formado por una sola neurona.

Esta neurona tiene 8 entradas (la dimensión de los vectores $\boldsymbol{v}_1$ y $\boldsymbol{v}_2$) y un bias. Si se supone que la salida es del tipo sigmoidal, al usar la regla Delta para una neurona sigmoidal y el siguiente conjunto inicial de pesos: $\boldsymbol{v} = (0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1)^T$ y $\alpha = 0.25$.

El lector puede demostrar que al final de 35 épocas, se obtiene el siguiente error $E$=0.0097 y el siguiente vector de pesos ajustados:

$$\boldsymbol{v}_f = (0.5742 \quad 0.1 \quad 0.5742 \quad 0.1 \quad -0.5424 \quad -0.5424 \quad 0.1 \quad 0.1 \quad 0.1420)^T.$$

En la figura 7.6 se muestra la estructura de la RNC resultante. Como se puede ver, esta estructura consta de una capa convolucionadora (des decir, dos convolucionadores) con función de activación tipo ReLU, una etapa de reducción tipo *max pooling*, una etapa de formación de vector descriptor y una red neuronal compuesta de una neurona con ocho entradas y una salida sigmoidea.

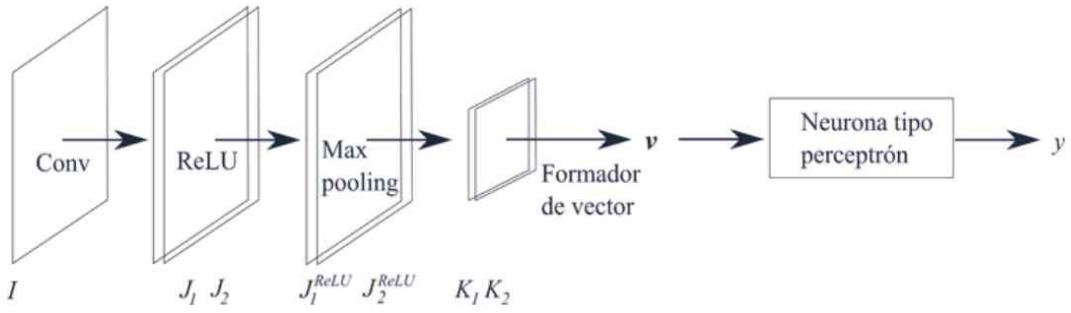



**Figura 7.6**: Arquitectura de la RNC para el ejemplo 7.5.

Enseguida se probará la capacidad de detección de la RNC diseñada. Se hará con los vectores y la neurona obtenidos. Prueba con el vector $\boldsymbol{v}_1$, representando un borde vertical. Comencemos calculando la activación de la neurona $a$:

$$a = \boldsymbol{v}_f \cdot \boldsymbol{x} = [0.5742 \quad 0.1 \quad 0.5742 \quad 0.1 \quad -0.5424 \quad -0.5424 \quad 0.1 \quad 0.1 \quad 0.1420] \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 3 \\ 3 \\ 0 \\ 0 \\ -1 \end{bmatrix} = 2.2184.$$

Ahora, calculemos la salida de la neurona. De acuerdo con lo visto, se debería obtener un valor cercano al "1". Veamos:

$$y = (1 + e^{-a})^{-1} = (1 + e^{-2.2184)})^{-1} = 0.902.$$

Procedamos ahora con la prueba con el vector $\boldsymbol{v}_2$, representando un borde horizontal. Comencemos calculando la activación de la neurona $a$:

$$a = \boldsymbol{v}_f \cdot \boldsymbol{x} = [0.5742 \quad 0.1 \quad 0.5742 \quad 0.1 \quad -0.5424 \quad -0.5424 \quad 0.1 \quad 0.1 \quad 0.1420] \begin{bmatrix} 3 \\ 0 \\ 3 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} = -2.248.$$

Finalmente, calculemos la salida de la neurona. De acuerdo con lo visto, se debería obtener un valor cercano al "0".

Es decir:

$$y = (1+e^{-a})^{-1} = (1+e^{2.248)})^{-1} = 0.094.$$

- Para el entrenamiento de una RNC como la descrita u otra más compleja, se puede usar la regla RDG con las debidas adecuaciones.

- Una de ellas es que en la etapa de propagación hacia atrás se debe tomar en cuenta, entre otras cosas, la derivada de la función ReLU.

Al igual que como en cualquier red neuronal multicapa, los pesos tanto de los convolucionadores, así como de todos los perceptrones de la capa de clasificación, generalmente son iniciados en valores aleatorios siguiendo alguna de las heurísticas obtenidas en la literatura.

Al final del entrenamiento los pesos de todos los convolucionadores y los perceptrones del clasificador habrán alcanzado valores estables según el criterio de paro elegido que, de nuevo, puede ser un valor de error o un número de épocas a alcanzar.

## 7.4 Resumen

En este capítulo primeramente, se dieron los detalles sobre el perceptrón multicapa. Se estableció un estudio de cómo acomodar esta máquina lineal en capas; y además, cómo puede ser utilizada dicha máquina para resolver problemas de clasificación convexos y no convexos.

Después, se ofrecieron todos los detalles sobre la derivación de la regla Delta generalizada, útil para el entrenamiento de arreglos de perceptrones en capas para la solución de problemas no lineales.

Se ilustró todo esto a través de varios ejemplos, que permitieron al lector asimilar de manera más sencilla los conceptos vertidos.

Enseguida, mediante un ejemplo con una red neuronal artificial de tres capas, se explicó la función de la capa intermedia para transformar un problema no lineal de clasificación en uno lineal, esto con el objeto de que el último o últimos perceptrones puedan resolver el problema.

Más adelante se explicó cómo al agregar el término de momento en la ecuación de ajuste de pesos de la red neuronal, es posible acelerar dicho proceso.

Posteriormente, se ofrecieron las bases de operación de una red neuronal convolucional. Esto se hizo a través de un ejemplo ilustrativo, lo cual permitió que el lector aprecie de manera sencilla las capacidades de esta poderosa máquina de procesamiento.

## 7.5 Problemas propuestos

Esta sección contiene un conjunto de problemas propuestos, cuyo desarrollo y solución permitirán al lector reafirmar el conjunto de conceptos vistos en la exposición del presente capítulo.

7.5.1 Para la RNA mostrada en la figura 7.3 y usada en el ejercicio 7.2, sustituir la neurona de salida $n_1^2$ sigmoidal por una neurona lineal tipo ADALINE. Re-entrene la RNA resultante tomando como base los mismos pesos y biases iniciales usados en el ejercicio 7.2 con 1000 épocas.

a) Demuestre que después de la aplicación de estos 1000 pesos se converge a una solución.

b) Clasifique los cuatro patrones de entrada y demuestre que la RNA entrenada es capaz de clasificarlos correctamente.

7.5.2 Aplique la regla Delta generalizada para entrenar una red neuronal con una capa intermedia con dos neuronas y luego sintetizar la función de separación entre las dos clases de patrones mostrados en la figura 7.7.

a) Muestre la arquitectura final de su red con pesos y bias.

b) Una vez hecho esto, clasificar los siguientes patrones:

(3,9), (11,2), (5,7) y (7,5).

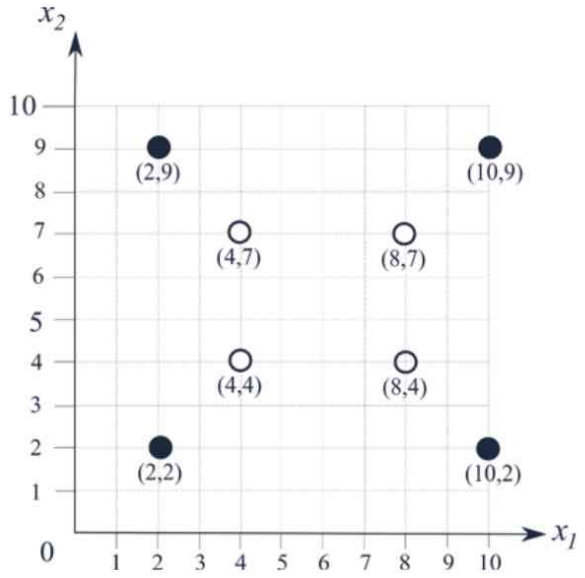



**Figura 7.7**: Datos usados en el problema 7.5.2.

7.5.3 Repita el problema anterior 7.5.2, pero con las siguientes características:

a) Utilice una red neuronal con una capa intermedia con tres neuronas.

b) Una vez hecho esto, clasificar los siguientes patrones:

(3,9), (11,2), (5,7) y (7,5).

7.5.4 Repita el problema 7.5.2, pero con las siguientes características:

a) Utilice una red neuronal con una capa intermedia con cuatro neuronas.

b) Una vez hecho esto, clasificar los siguientes patrones:

(4,10), (8,2), (5,6) y (7,5).

7.5.5 Aplique la regla Delta generalizada para entrenar una red neuronal con una capa intermedia con tres neuronas y luego sintetizar la función de separación entre las dos clases de patrones ilustrados en la figura 7.8.

a) Muestre la arquitectura final de su red con pesos y bias.

b) Una vez hecho esto, clasificar los siguientes patrones:

(9,10), (2,5), (4,4) y (6,8).

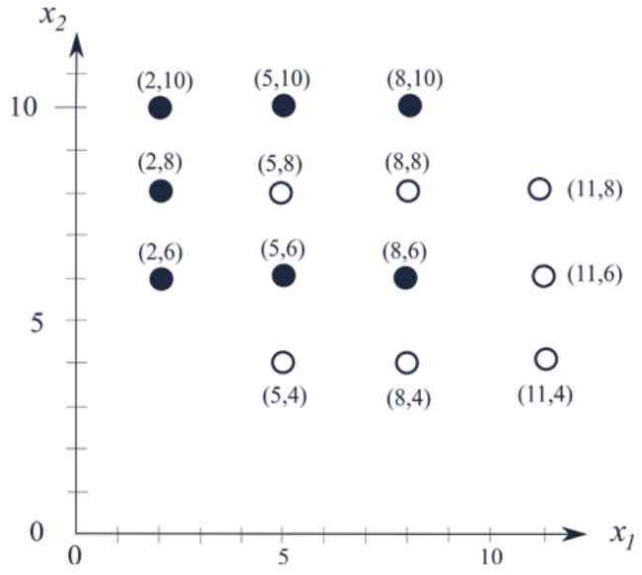



**Figura 7.8**: Datos usados en el problema 7.5.5.

7.5.6 Suponga que en una imagen $I$ de 8×8 elementos quiere detectar la presencia ya sea de una letra "E" como la mostrada en la figura 7.9a o una letra "U", como se ilustra en la figura 7.9b.

a) Diseñe una red neuronal convolucional con una sola capa de convolutiva y una capa de reducción, y una red neuronal de perceptrones que permita decir si una imagen contiene una "E" o una "U".

b) Pruebe con las cuatro imágenes mostradas en las figuras 7.9c, 7.9d, 7.9e y 7.9f.

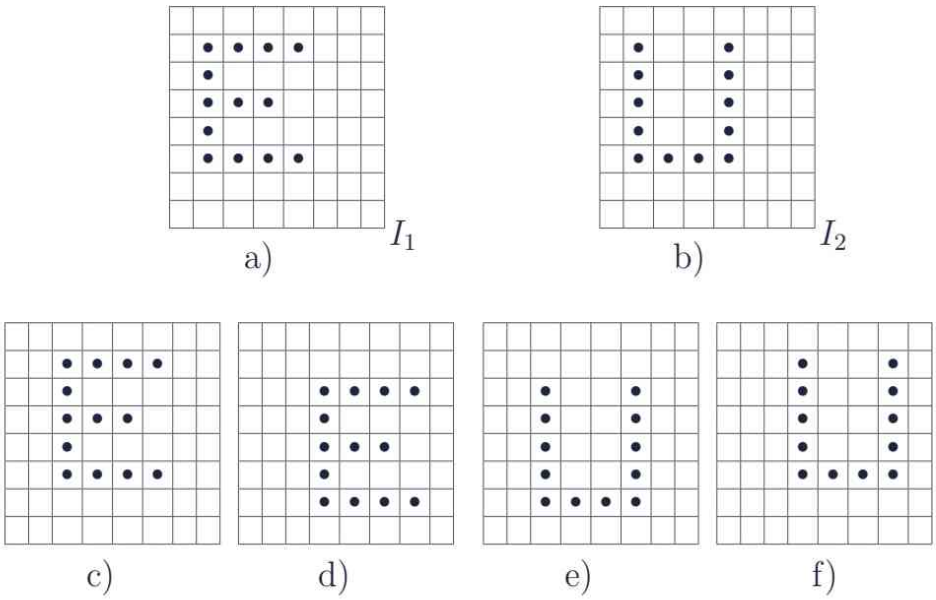



**Figura 7.9**: Datos usados en el ejercicio 7.5.6.

7.5.7 Resuelva el problema 7.5.6 con dos figuras geométricas, un rectángulo ▭ y un triángulo △.

# Referencias a sitios Web


[Web1] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.645.7760&rep=rep1&type=pdf

[Web2] https://arxiv.org/pdf/1804.06655.pdf

[Web3] https://ieeexplore.ieee.org/document/8248937

[Web4] https://medium.com/datadriveninvestor/artificial-intelligence-and-autonomous-vehicles-ae877feb6cd2

[Web5] https://www.wired.com/story/guide-self-driving-cars/

[Web6] https://spectrum.ieee.org/tech-talk/robotics/artificial-intelligence/mb

[Web7] https://spectrum.ieee.org/tech-talk/robotics/artificial-intelligence/deepminds-ai-shows-itself-to-be-a-worldbeating-world-builder?utm_source=techalert&utm_campaign=techalert-02-07-19&utm_medium=email

[Web8] https://www.britannica.com/technology/artificial-intelligence#ref739464

[Web9] https://www.pinterest.com.mx/pin/333407178640725920/

[Web10] http://noticias.universia.com.ar/cultura/noticia/2015/04/01/1122560/9-tipos-inteligencia-segun-howard-gardner.html

[Web11] https://laguiafemenina.com/psicologia/tipos-inteligencia

[Web12] https://www.eleconomista.com.mx/tecnologia/4-tipos-de-Inteligencia-Artificial-que-debes-conocer-20161115-0186.html

[Web13] https://www.quierotec.com/tipos-de-inteligencia-artificial/

[Web14] https://www.tuinteligenciaartificial.es/tipos-inteligencia-artificial/

[Web15] https://omicrono.elespanol.com/2018/10/que-es-la-inteligencia-artificial/

[Web16] https://es.wikipedia.org/wiki/Inteligencia_artificial

[Web17] https://www.wired.com/story/guide-artificial-intelligence/

[Web18] http://inteligenciaartificialgrupo7.blogspot.com/2013/02/normal-0-21-false-false-false-es-ve-x.html

[Web19] https://es.wikipedia.org/wiki/Aplicaciones_de_la_inteligencia_artificial#Lista_de_aplicaciones

[Web20] https://itcl.es/blog/donde-encontrar-inteligencia-artificial/

[Web21] https://arxiv.org/abs/1606.04671

[Web22] http://www.bbc.com/future/story/20181204-the-chef-making-fast-food-even-faster

[Web23] http://www.sciencemag.org/news/2018/12/artificial-intelligence-helps-predict-volcanic-eruptions

[Web24] https://www.nytimes.com/2018/12/18/technology/driverless-mini-car-deliver-groceries.html

[Web25] https://www.bbc.com/news/amp/technology-46987779

[Web26] http://www.xinhuanet.com/english/2018-12/24/c_137695066.htm

[Web27] https://www.bloomberg.com/news/articles/2018-12-26/toyota-wants-to-put-a-robot-in-every-home-and-make-it-your-pal?srnd=premium-asia

[Web28] https://www.nytimes.com/2018/12/26/science/chess-artificial-intelligence.html

[Web29] https://www.telegraph.co.uk/technology/2019/01/02/uk-tests-autonomous-martian-robot/

[Web30] https://www.wsj.com/articles/why-your-ice-cream-will-ride-in-a-self-driving-car-before-you-do-11546664589?mod=itp_wsj&ru=yahoo

[Web31] https://www.wired.com/story/the-clever-clumsiness-of-a-robot-teaching-itself-to-walk/

[Web32] https://www.apnews.com/6034c9ce1af347ec8da996e39b29c51b

[Web33] https://spectrum.ieee.org/the-human-os/biomedical/bionics/ai-helps-humans-walk-on-robot-prosthetic-knee?utm_source=techalert&utm_campaign=techalert-01-31-19&utm_medium=email

[Web34] https://spectrum.ieee.org/computing/software/mayhem-the-machine-thatfinds-software-vulnerabilities-then-patches-them?utm_source=circuitsandsensors&utm_campaign=circuitsandsensors-02-05-19&utm_medium=email

[Web35] https://www.bloomberg.com/news/articles/2019-02-14/the-ai-that-can-write-a-fake-news-story-from-a-handful-of-words

[Web36] https://www.fraunhofer.de/en/press/research-news/2019/february/software-that-can-automatically-detect-fake-news.html

[Web37] https://www.nytimes.com/2019/02/05/business/media/artificial-intelligence-journalism-robots.html

[Web38] https://www.sciencemag.org/news/2019/02/pictionary-playing-computer-connects-humans-deep-thoughts?r3f_986=https://www.google.com/

[Web39] https://www.dailymail.co.uk/health/article-6673115/Robbie-Robot-spot-worsening-dementia-watching-13-episodes-Emmerdale.html

[Web40] https://www.washingtonpost.com/express/2019/02/15/is-that-robot-dr-bear-bot-helps-care-kids-local-hospital/?utm_term=.4742f92ba94f

[Web41] https://spectrum.ieee.org/cars-that-think/transportation/self-driving/autonomous-parking?utm_source=roboticsnews&utm_campaign=roboticsnews-02-26-19&utm_medium=email

[Web42] https://spectrum.ieee.org/tech-talk/computing/software/artificially-intelligent-players-invent-nonverbal-languages-to-win-card-games?utm_source=techalert&utm_campaign=techalert-02-28-19&utm_medium=email

[Web43] https://www.cnbc.com/2019/02/23/new-technology-shows-promise-reducing-pedestrian-fatalities.html

[Web44] https://www.washingtonpost.com/technology/2019/02/27/your-next-fedex-delivery-could-be-pizza/?noredirect=on&utm_term=.10f2225621d2

[Web45] https://www.cnbc.com/2019/02/27/alphabets-deepmind-uses-machine-learning-to-predict-wind-power-output.html

[Web46] https://www.nature.com/articles/d41586-019-00746-1

[Web47] https://www.wsj.com/articles/tonights-dinner-in-a-cooler-sized-robot-that-knows-where-you-live-11552296601?mod=itp_wsj&ru=yahoo

[Web48] https://www.nytimes.com/2019/03/10/technology/artificial-intelligence-eye-hospital-india.html

[Web49] https://viterbischool.usc.edu/news/2019/03/a-robotic-leg-born-without-prior-knowledge-learns-to-walk/

[Web50] https://aktuelles.uni-frankfurt.de/englisch/first-machine-generated-book-published/

[Web51] https://www.newscientist.com/article/2197941-driverless-car-learns-to-perform-high-speed-turns-without-crashing/

[Web52] https://www.xataka.com/inteligencia-artificial/speedgate-nuevo-deporte-creado-inteligencia-artificial-a-partir-reglas-otras-400-modalidades-deportivas/amp

[Web53] https://www.forbes.com/sites/bernardmarr/2019/04/05/the-fascinating-ways-pepsico-uses-artificial-intelligence-and-machine-learning-to-deliver-success/amp/

[Web54] https://www.nytimes.com/2019/03/04/business/ai-technology-travel-planning.html

[Web55] https://spectrum.ieee.org/tech-talk/computing/networks/using-ai-to-make-better-ai?mkt_tok=eyJpIjoiTm1RNU0yTmxNemd4WmpKayIsInQOiI0Z2cyVFdFcEJvUTdMdDlNT0JLbjdmaGdBbFBXXC9kelBEV2diVHVMdHRTelQ1Snc0endsNDVnWDNQdUZR UkE0WkwxeDNkS2pONzhPVUdGUGlMQ2hnWVhvWUU3T0lwcm1NSHNEeWNGVEx0 TmhOWDZpM3hLcE1SNzZUTmNOVUMxWGEifQ%3D%3D

[Web56] https://blog.es.logicalis.com/analytics/inteligencia-artificial-el-tiempo-es-ahora

[Web57] https://www.techrepublic.com/article/73-of-developers-who-dont-use-ai-plan-to-learn-how-in-2018/

[Web58] https://www.washingtonpost.com/technology/2019/03/27/ artificial-intelligence-pioneers-win-turing-award/?noredirect= on&utm_term=.406cb50d8f3e

[Web59] https://www.forbes.com/sites/nicolemartin1/2019/03/27/turing-award-and-1-million-given-to-3-ai-pioneers/#254672904784

[Web60] https://www.britannica.com/topic/CYC

[Web61] https://en.wikipedia.org/wiki/Rodney_Brooks

[Web62] http://theinstitute.ieee.org/ieee-roundup/blogs/blog/understanding-the-risks-associated-with-ai

# Referencias


[1] N. Brown, T. Sandholm. "Libratus: The Superhuman AI for no-limit poker". Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. Demos. pp. 5226-5228. https://doi.org/10.24963/ijcai.2017/772. 2017.

[2] R. Han. *Math for Machine Learning: Open Doors to Data Science and Artificial Intelligence. CreateSpace Independent.* Publishing Platform. July 12, 2018.

[3] E. Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press. Third edition. August 22, 2014.

[4] E. Alpaydin. *Machine Learning: The New AI.* (The MIT Press Essential Knowledge series). October 7, 2016.

[5] A. Burkov. *The Hundred-Page Machine.* Learning Book. January 13, 2019.

[6] A. M. Legendre. *Nouvelles Méthodes Pour la Détermination des Orbites des Cométes (New Methods for the Determination of the Orbits of Comets)* (in French), Paris: F. Didot. 1805.

[7] M. Gautier and W. Khalil "Exciting trajectories for the identification of base inertial paramteres of Robots". The Int. J. of Robotics Research. 11(4). pp. 362-375. 1992.

[8] K. Kelly. *Lo inevitable: Entender las 12 Fuerzas Tecnológicas que Configurarán Nuestro Futuro.* Penguin Random House LLC. 2016.

[9] N. Bostrom. *Superinteligencia: Caminos, Peligros, estrategias.* España Teell Editorial, S.L. 2016.

# Referencias


[1] N. Brown, T. Sandholm. "Libratus: The Superhuman AI for no-limit poker". Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. Demos. pp. 5226-5228. https://doi.org/10.24963/ijcai.2017/772. 2017.

[2] R. Han. *Math for Machine Learning: Open Doors to Data Science and Artificial Intelligence. CreateSpace Independent.* Publishing Platform. July 12, 2018.

[3] E. Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press. Third edition. August 22, 2014.

[4] E. Alpaydin. *Machine Learning: The New AI.* (The MIT Press Essential Knowledge series). October 7, 2016.

[5] A. Burkov. *The Hundred-Page Machine.* Learning Book. January 13, 2019.

[6] A. M. Legendre. *Nouvelles Méthodes Pour la Détermination des Orbites des Cométes (New Methods for the Determination of the Orbits of Comets)* (in French), Paris: F. Didot. 1805.

[7] M. Gautier and W. Khalil "Exciting trajectories for the identification of base inertial paramteres of Robots". The Int. J. of Robotics Research. 11(4). pp. 362-375. 1992.

[8] K. Kelly. *Lo inevitable: Entender las 12 Fuerzas Tecnológicas que Configurarán Nuestro Futuro.* Penguin Random House LLC. 2016.

[9] N. Bostrom. *Superinteligencia: Caminos, Peligros, estrategias.* España Teell Editorial, S.L. 2016.

[10] J. M. Morel and S. Solimini. *Variational Methods in Image Segmentation: With Seven Image Processing Experiments*. Progress in Nonlinear Differential Equations and Their Applications 14 Birkhäuser. 1994.

[11] M. Ford. *El Ascenso de los Robots*. España, Paidos. 2016.

[12] N. Brealey. *New Scientist. Machines that Think: Everything you need to know about the coming age of artificial intelligence*. (Instant Expert). Massachusetts. 2017

[13] A. Wilson y J. Ledezma Millán. *El Ascenso de la Automatización: La Tecnología y los Robots Reemplazarán a los humanos*. Antioch, Tn. Adidas Wilson. 2018.

[14] D. M. West. "The Future of Work: Robots, AI, and Automation". Brookings Institution Press. 2018.

[15] E. Bribiesca. "Measuring 2D Shape Compactness using the Contact Perimeter". Computers Math. Applications, 33(11):1-9, 1997.

[16] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. New Yersey, Addison Wesley. 1992.

[17] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, Inc. Third Edition. New Jersey. 2002.

[18] M. K. Hu. "Visual Pattern Recognition by Moment Invariants". IRE Transactions on Information Theory. pp. 179-187, 1962.

[19] R. Jain, R. Kasturi and B. G. Schunck. *Machine Vision*. New York. McGraw-Hill Science/Engineering/Math. 1995.

[20] J. Kittler and J. Illingworth. "Minimun Error Thresholding." Pattern Recognition, 19(1):41-47, 1986.

[21] N. Otsu. "A Threshold Selection Method from Gray-Level Histograms". IEEE Transactions on Systems, Man, and Cybernetics, 9(1):62-66. 1979.

[22] H. Sossa, A. Carreón, R. Santiago, E. Bribiesca and A. Petrilli. "Efficient Computation of the Euler Number of a 2-D Binary of Image". Advances in Computational Intelligence LNAI 10061/10062. Springer. pp. 401-413. 2016.

[23] H. S. Yang and S. Sengupta. "Intelligent shape recognition for complex industrial tasks". IEEE Control Syst. Mag. 8(3):23-30. 1988.

[24] D. P. Bertsekas *Nonlinear Programming*. Athena Scientific 1999, 2nd edition, pp. 187.

[25] X. Glorot, A. Bordes and Y. Bengio. "Deep sparse rectifier neural networks". Journal of Machine Learning Research 15:315-323. 2011.

[26] G. Cybenko. "Approximation by superpositions of a sigmoidal function". Mathematics of Control, Signals and Systems 2(4):303-314. 1989.

[27] A. Cauchy. *Méthode générale Pour la résolution des systemes d'équations simultanées.* Comp. Rend. Sci. Paris 25.1847 (1847):536-538.

[28] M. T. Hagan, H. B Demuth and M. H. Beale. *Neural Network Design.* Second Edition. 2014.

[29] R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, H. S. Seung, H. S. *Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit.* Nature 405:947-951.2000.

[30] R. Hahnloser and H. S. Seung. *Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks.* NIPS 2001.

[31] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Singapure. Prentice Hall. Second Edition. 1998.

[32] S. Haykin. *Neural Networks and Learning Machines.* New York. Pearson. Third Edition. 2018.

[33] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory.* Oxford, England: Wiley. 1949.

[34] J. R. Hilera y V. J. Martínez. *Redes Neuronales Artificiales: Fundamentos, Modelos y Aplicaciones.* RA-MA. Madrid. España. 2000.

[35] S. Kalyanakrishnan. "The Perceptron Learning Algorithm and its Convergence". Indian Institute of Technology Bombay. pp. 1-5. 2017.

[36] R. Lippmann. "An introduction to computing with neural nets". IEEE ASSP Magazine 4(2):4-22. 1987.

[37] D. M. Gomez Allende. *Reconocimiento de Formas y Visión Artificial*. España. RA-MA. 1993.

[38] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". Bulletin of Mathematical Biophysics 5(4):115-133. 1943.

[39] J. Hodgkin. "Caenorhabditis Elegans. In Sydney Brenner and Jefferey H. Miller". Encyclopedia of Genetics. pp. 251-256. 2001.

[40] M. L. Minsky and S. A. Papert. *Perceptrons*. Boston, Mas. MIT Press, Cambridge. 1969.

[41] W. H, Press, B. P. Flannery, S. A. Teukolsky and W. T. Vitterling. *Numerical receipes in C*. Massachusetts, Cambridge University Press. 1992.

[42] J. M. S. Prewitt. Object *Enhancement and Extraction, Picture processing and Psychopictorics*. New York. Academic Press. 1970.

[43] F. Reyes Cortés. *Robótica: Control de Robots Manipuladores*. México. Grupo Editor Alfaomega. 2011.

[44] F. Reyes Cortés. ***MATLAB**: Aplicado a Robótica y Mecatrónica*. México. Grupo Editor Alfaomega. 2012.

[45] F. Rosenblatt. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. Psychological Review, 65(6), 386-408. 1958.

[46] F. Rosenblatt. *Principles of Neurodynamics: Perceptions and the Theory of Brain Mechanisms*. New York. Spartan Books. 1962.

[47] H. Wang and B. Raj. *On the Origin of Deep Learning*. ArXiv e-prints. arXiv:170207800. 2017.

[48] B. Widrow and S. D. Stearns. *Adaptive Signal Processing*. New Jersey: Prentice-Hall, Inc. 1985.

# Índice analítico

## A

## B



## C

## D



## E

[Web63] https://issues.org/perspective-should-artificial-intelligence-be-regulated/

[Web64] https://issues.org/perspective-should-artificial- intelligence-be-regulated/

[Web65] https://futureoflife.org/background/benefits-risks-of-artificial-intelligence/?cn-reloaded=1

[Web66] https://tecno.americaeconomia.com/articulos/china-concentra-el-22-de-las-patentes-de-inteligencia-artificial-nivel-mundial

[Web67] https://en.wikipedia.org/wiki/Hessian_matrix

[Web68] https://en.wikipedia.org/wiki/Second_partial_derivative_test

[Web69] https://en.wikipedia.org/wiki/Likelihood_function

[Web70] http://fourier.eng.hmc.edu/e176/lectures/NM/node21.html

[Web71] https://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf

[Web72] https://seas.ucla.edu/ kao/nndl/lectures/cnn.pdf

[Web73] https://en.wikipedia.org/wiki/Convolution

## F

## G



## H



## I

## K



## L



## M

## N

## O



## P

## R

## S

## T



## U

## V



## W



## X