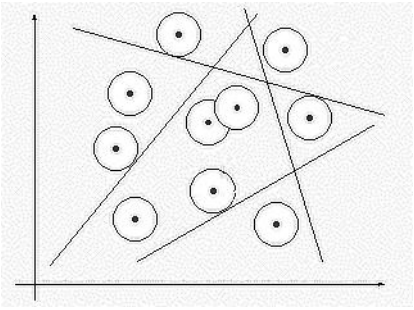
CLASIFICACIÓN CON DISCRIMINANTES: UN ENFOQUE NEURONAL



Juan Antonio Pérez Ortiz
japerez@dlsi.ua.es

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
Julio 1999

Prefacio

Este trabajo pretende ser una introducción a la utilización de distintos modelos de redes neuronales en la clasificación de patrones. Aunque no se estudian muchos de los modelos existentes cuando se redactó (ni, evidentemente, ninguno de los que con seguridad han surgido después), creo que los que aparecen son bastante representativos y pueden dar al lector una visión detallada del enfoque neuronal del reconocimiento de formas. El trabajo no puede, ni pretende, sustituir a la gran cantidad de excelentes manuales que abordan el tema, pero sí ser una guía rápida que lo describa brevemente, sin entrar en demostraciones o en aspectos de segunda fila.

El capítulo 1 realiza una pequeña introducción a las redes neuronales. Se describe el funcionamiento básico de una neurona, las posibles arquitecturas, el método de optimización de descenso por gradiente y la regla delta para modificar el peso de las conexiones presinápticas de una neurona.

El capítulo 2 describe el problema de clasificación utilizado en el resto de capítulos para evaluar los distintos algoritmos de clasificación estudiados.

El capítulo 3 aborda el problema de la clasificación desde el punto de vista de una sola neurona lineal, técnica llamada tradicionalmente «clasificación mediante funciones discriminantes lineales». Se presenta el concepto de separabilidad lineal y los algoritmos del perceptrón y de Widrow-Hoff.

El capítulo 4 se limita a mostrar los resultados obtenidos sobre el problema del capítulo 2 con un perceptrón multicapa. No se describe el algoritmo de retropropagación utilizado para entrenarlo, aunque existen miles de referencias sobre ello. En cualquier caso, su compresión no es necesaria para los capítulos posteriores.

El capítulo 5 analiza las redes neuronales basadas en funciones de base radial. Se enuncia el teorema de Cover sobre la separabilidad de patrones y las ideas básicas de interpolación con funciones de base radial. También se discute la aplicación de la teoría de la regularización para definir una generalización del modelo básico. Por último, se muestran algunos resultados

del uso de estas redes en el problema de clasificación del capítulo 2.

El capítulo 6 es una breve introducción a la minimización estructural del riesgo, una forma de aprendizaje que está demostrandose superior a muchos de los algoritmos clásicos utilizados con redes neuronales. Se define la dimensión de Vapnik-Chervonenkis y se dan las ideas esenciales para comprender las máquinas de vectores soporte, objeto del capítulo 7.

El capítulo 7 presenta una de las últimas y más importantes técnicas surgidas en la última década en el campo del reconocimiento de formas: las máquinas de vectores soporte. Se discuten sus excelentes resultados en la clasificación de muestras no linealmente separables y, de nuevo, se evalúan sobre el problema del capítulo 2.

En los apéndices puede encontrarse una pequeña introducción al clasificador de Bayes, necesario para entender algunos aspectos del capítulo 2 y del 3.

La referencia [4] ha sido la más consultada a lo largo de este trabajo. En ella se estudiaron la mayor parte de los contenidos de los capítulos 2, 5, 6 y parte del 1. [5] se ha utilizado para parte del capítulo 1 y [2] para el tercer capítulo y el apéndice A. La excelente referencia [1] es la fuente principal de los capítulos 6 y 7, aunque también se recogen en ellos ideas de [3], [4] y [6].

Este manúscrito fue desarrollado para la asignatura *Reconocimento de Formas* del programa de doctorado del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante. Los experimentos que finalizan algunos capítulos fueron desarrollados con código propio del autor o con la adaptación al programa `Octave` del código escrito por Hugh Pasika para [4].

Juan Antonio Pérez Ortiz
Alicante, 8 de julio de 1999

Índice general

Prefacio I

Listado de símbolos V

1. Redes neuronales 1
- 1.1. Modelo de neurona . . . 2
- 1.2. Arquitecturas neuronales . . . 4
- 1.3. Técnicas de optimización . . . 5
  - 1.3.1. Descenso por gradiente . . . 6
  - 1.3.2. Superficie de error . . . 7
- 1.4. La regla delta . . . 7

2. Un problema de clasificación 9
- 2.1. Frontera de decisión bayesiana . . . 10
- 2.2. Estimación del error . . . 12

3. Funciones discriminantes lineales 15
- 3.1. Clasificación mediante discriminantes . . . 15
  - 3.1.1. El umbral como un peso más . . . 16
- 3.2. Separabilidad lineal . . . 17
- 3.3. Algoritmo perceptrón . . . 19
  - 3.3.1. Incremento variable . . . 20
- 3.4. Relación entre el perceptrón y un clasificador bayesiano . . . 21
- 3.5. Algoritmos de mínimos cuadrados . . . 23
  - 3.5.1. Algoritmo de Widrow-Hoff . . . 24
  - 3.5.2. Esquemas de variación de la tasa de aprendizaje . . . 24
- 3.6. Múltiples clases . . . 25
- 3.7. Experimentos . . . 26

4. Perceptrón multicapa 29
- 4.1. Experimentos . . . 30

5. Funciones de base radial 31
- 5.1. Teorema de Cover . . . 31

5.2. Interpolación 33
5.3. Teoría de la regularización 34
5.4. Redes de regularización 37
5.5. Redes de funciones de base radial generalizadas 37
5.5.1. Estimación del parámetro de regularización 40
5.5.2. La maldición de la dimensionalidad 40
5.5.3. Comparación entre RFBR y PMC 40
5.6. Experimentos 41

6. Minimización del error **43**
6.1. Riesgo esperado y riesgo empírico 43
6.2. La dimensión de Vapnik-Chervonenkis 45
6.2.1. Dimensión VC del conjunto de hiperplanos orientados 45
6.2.2. Acotación del riesgo 46
6.3. Minimización estructural del riesgo 47

7. Máquinas de vectores soporte **49**
7.1. Máquinas de vectores soporte lineales 50
7.1.1. Fase de evaluación 52
7.2. El caso no separable 53
7.3. Máquinas no lineales de vectores soporte 54
7.3.1. Ejemplo de núcleo 55
7.3.2. Condición de Mercer 57
7.3.3. Algunas MVS no lineales 57
7.4. La dimensión VC de las MVS 58
7.5. Limitaciones 60
7.6. Experimentos 61

A. Clasificador de Bayes **63**

Bibliografía **67**

Listado de símbolos

b	peso umbral
$\boldsymbol{C}$	matriz de covarianza
d_i	salida esperada para la i-ésima muestra
$d(n)$	salida esperada en el instante n
$\det(\boldsymbol{A})$	determinante de la matriz $\boldsymbol{A}$
$e(n)$	diferencia entre la salida esperada y la real $e(n) = d(n) - y(n)$
$\mathcal{E}$	función de error
$\boldsymbol{g}$	gradiente del error, $\boldsymbol{g} = \boldsymbol{\nabla}\mathcal{E}(\boldsymbol{w})$
G	función de Green
$\boldsymbol{G}$	matriz de funciones de Green, $\boldsymbol{G} = \{G_{ji} = G(\boldsymbol{x}_j, \boldsymbol{x}_i)\}$
h	dimensión de Vapnik-Chervonenkis
$\mathcal{H}$	espacio de características
$\boldsymbol{I}$	matriz identidad
K	función nucleo
L_P	Lagrangiano primario
L_D	Lagrangiano dual
m	dimensión del espacio de entrada
m_0	dimensión del espacio de entrada
N	número de muestras de entrenamiento
P_e	probabilidad de error del clasificador
R	riesgo esperado o riesgo real
R_{emp}	riesgo empírico
sgn	función signo
$\boldsymbol{w}$	vector de pesos
$\boldsymbol{w}(n)$	vector de pesos en el instante n
$\boldsymbol{w}^*$	valor óptimo de los pesos para una $\mathcal{E}$ concreta
$\boldsymbol{x}_i$	i-ésima muestra de entrenamiento

$\boldsymbol{x}(n)$	vector de entrada aplicado en el instante n
$\mathcal{X}$	conjunto de patrones de entrenamiento
$\mathcal{X}_i$	i-ésima clase en un problema de clasificación
y	salida de la neurona
$y(n)$	salida real en el instante n
z_i	salida espera para la i-ésima muestra
α_i	multiplicador de Lagrange
η	tasa de aprendizaje
η_0	valor inicial de la tasa de aprendizaje
$\eta(n)$	tasa de aprendizaje en el instante n
θ	función escalón
λ	parámetro de regularización
$\Lambda(\boldsymbol{x})$	ratio de verosimilitud
$\boldsymbol{\mu}$	vector de medias
ξ	umbral para el clasificador de Bayes
σ	desviación estándar
σ^2	varianza
τ	constante de tiempo de búsqueda
$\boldsymbol{\Phi}$	matriz de interpolación, $\boldsymbol{\Phi} = \{\varphi_{ji} = \varphi(\Vert\boldsymbol{x}_j - \boldsymbol{x}_i\Vert)\}$
φ_i	i-ésima función de base radial
$\boldsymbol{\nabla}$	operador gradiente
$\boldsymbol{A}^T$	transpuesta de la matriz $\boldsymbol{A}$
$\boldsymbol{A}^+$	pseudoinversa de la matriz $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	inversa de la matriz $\boldsymbol{A}$

Capítulo 1

Redes neuronales

Una *red neuronal* puede verse como una *máquina* designada para modelizar la forma en que el cerebro humano realiza una determinada tarea. Para lograr este objetivo, una red neuronal está formada por un conjunto de unidades de procesamiento interconectadas llamadas *neuronas*.

Cada neurona recibe como entrada un conjunto de señales discretas o continuas, las pondera e integra, y transmite el resultado a las neuronas conectadas a ella. Cada conexión entre dos neuronas tiene una determinada importancia asociada denominada *peso sináptico* o, simplemente, *peso*. En los pesos se guarda la mayor parte del conocimiento que la red neuronal tiene sobre el problema en cuestión. El proceso mediante el cual se ajustan estos pesos para lograr un determinado objetivo se denomina *aprendizaje* o *entrenamiento* y el procedimiento concreto utilizado para ello se conoce como *algoritmo de aprendizaje* o *algoritmo de entrenamiento*.

El ajuste de pesos es la principal forma de aprendizaje de las redes neuronales, aunque hay otras formas posibles como la modificación de la topología de la propia red neuronal.

Los algoritmos de aprendizaje pueden dividirse en dos categorías: *supervisados* y *no supervisados*. En los primeros la red neuronal cuenta con el apoyo de un *maestro* que informa de la corrección de la salida producida por la red de acuerdo con la salida considerada correcta. En los segundos, no existe tal maestro y la red neuronal debe extraer sin ayuda características de los datos que se le suministra. Este trabajo se centra en redes neuronales para la clasificación de patrones; por ello, todos los algoritmos de aprendizaje estudiados serán supervisados. Durante el entrenamiento, las entradas al algoritmo serán los patrones a clasificar y las salidas deseadas la clase a la que pertenezca cada uno. En la fase de *evaluación* (utilización) de la red

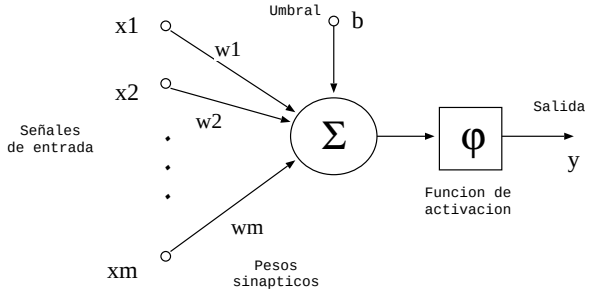


Figura 1.1: Modelo no lineal de una neurona.

neuronal, se suministran nuevos patrones a la red neuronal y se observa su salida para determinar la clase a la que pertenece el nuevo patrón.

Las redes neuronales destacan por su estructura fácilmente paralelizable y por su elevada capacidad de *generalización* (capacidad de producir salidas correctas ante entradas no vistas durante el entrenamiento). Otras propiedades interesantes son:

No linealidad Una red neuronal puede ser lineal o no lineal. La no linealidad es muy importante, en especial si el mecanismo físico responsable de la generación de la señal de entrada lo es.

Adaptatividad Las redes neuronales son capaces de reajustar sus pesos para adaptarse a cambios en el entorno. Esto es especialmente útil cuando el entorno que suministra los datos de entrada es no estacionario.

Confianza de la respuesta En el contexo del reconocimiento de formas, una red neuronal puede diseñarse para dar no solo información sobre la clase a la que pertenece un determinado patrón, sino también para dar información sobre la confianza de la decisión.

Tolerancia a fallos Una red neuronal, implementada en hardware, es tolerante a fallos en el sentido de que fallos operacionales en partes de la red pueden afectar débilmente al rendimiento de esta. Esta propiedad es debida a la naturaleza distribuida de la información almacenada en la red neuronal.

1.1. Modelo de neurona

La figura 1.1 muestra el modelo más común de neurona. En él se identifican cinco elementos básicos:

- Un conjunto de m señales de entrada, que suministran a la red los datos del entorno.
- Un conjunto de *sinapsis*, caracterizada cada una por un peso propio w_i. El peso w_i está asociado a la sinapsis que conecta la entrada i-ésima con la neurona.
- Un *umbral* o *sesgo* b, que aumenta la capacidad de procesamiento de la neurona y que eleva o reduce la entrada a la neurona, según sea su valor positivo o negativo.
- Un *sumador* o *integrador* que suma las señales de entrada, ponderadas con sus respectivos pesos, y el umbral.
- Una *función de activación* φ que suele limitar la amplitud de la salida de la neurona.

En términos matemáticos podemos describir la operación de una neurona mediante la ecuación

$$z = \varphi\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{1.1}$$

o en forma vectorial

$$z = \varphi(\boldsymbol{w}^T \boldsymbol{x} + b) \tag{1.2}$$

donde

$$\boldsymbol{w} = [w_1, w_2, \ldots, w_m]^T \tag{1.3}$$

y

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T \tag{1.4}$$

La función de activación es la que define en última instancia la salida de la neurona. En este trabajo se utilizan tres tipos básicos de funciones de activación:

1. *Función lineal.* Tiene la forma $\varphi(x) = x$ y se utiliza cuando no se desea acotar la salida de la neurona.
2. *Función escalón.* Adopta la forma

$$\varphi(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases} \tag{1.5}$$

y proporciona una salida bivaluada.

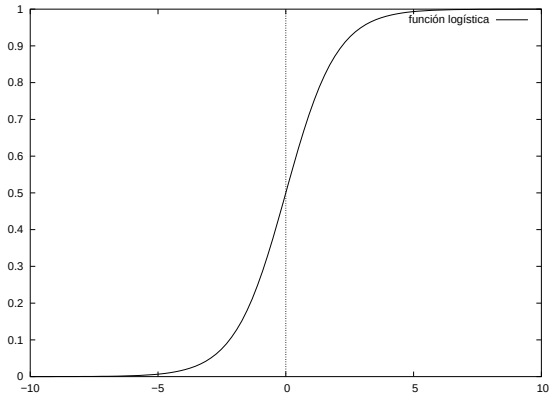


Figura 1.2: Representación gráfica de la función logística, $1/(1+e^{-x})$.

3. *Función sigmoidea.* Aunque en este trabajo prácticamente no aparece, la función sigmoidea es habitual en muchos modelos neuronales y provoca una transformación no lineal de su argumento. Una de las más utilizadas es la *función logística* definida por

$$\varphi(x) = \frac{1}{1+\exp(-ax)} \tag{1.6}$$

donde a es un *parámetro de inclinación* que ajusta la inclinación de la función. La función logística para $a = 1$ se muestra en la figura 1.2.

1.2. Arquitecturas neuronales

La forma en la que se estructuran las neuronas de una red neuronal está íntimamente ligada con el algoritmo de aprendizaje utilizado para entrenarlas. En general, podemos identificar tres clases diferentes de arquitecturas neuronales:

Redes de una capa sin ciclos Las neuronas se disponen en paralelo en una sola capa. Cada señal de entrada está unida con una sinapsis distinta (con peso sináptico propio) a todas las neuronas de la red. Los pesos de estas conexiones y los umbrales (uno por neurona) son los únicos pesos de la red, es decir, no hay interconexiones entre las distintas neuronas (ciclos). Se dice que las entradas constituyen la *capa de entrada* y que las neuronas conforman la *capa de salida.*

Redes multicapa sin ciclos Se distinguen de las anteriores por la presencia adicional de una o más *capas ocultas*, cuyas neuronas se denominan

neuronas ocultas. Cada neurona solo está conectada con neuronas de las capas siguientes (no hay ciclos), aunque normalmente estas conexiones solo se producen con las de la capa inmediatamente contigua.

Redes recurrentes Se distinguen de las dos anteriores en que tienen un ciclo como mínimo. Estos ciclos tienen un impacto profundo en la capacidad de aprendizaje de la red y la hacen especialmente indicada para el procesamiento de secuencias temporales. No se utilizan en este trabajo y no hablaremos más de ellas.

1.3. Técnicas de optimización

Consideremos una *función de error* $\mathcal{E}(\boldsymbol{w})$ (también llamada *función de coste*), continuamente diferenciable, donde $\boldsymbol{w}$ es un vector con los pesos (parámetros) de una determinada red neuronal. La función $\mathcal{E}(\boldsymbol{w})$ proporciona un número real que es una medida de la corrección de los pesos $\boldsymbol{w}$ para resolver un determinado problema de forma óptima. Nos interesa encontrar una solución óptima $\boldsymbol{w}^*$ que satisfaga la condición

$$\mathcal{E}(\boldsymbol{w}^*) \leq \mathcal{E}(\boldsymbol{w}) \tag{1.7}$$

es decir, queremos minimizar la función de coste $\mathcal{E}(\boldsymbol{w})$ con respecto a $\boldsymbol{w}$. Se trata en definitiva de un problema de optimización sin restricciones.

La condición necesaria para el óptimo es

$$\boldsymbol{\nabla}\mathcal{E}(\boldsymbol{w}^*) = \mathbf{0} \tag{1.8}$$

donde ∇ es el operador de gradiente

$$\nabla = \left(\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \ldots, \frac{\partial}{\partial w_m}\right) \tag{1.9}$$

Una clase de algoritmos de optimización que suelen adaptarse bien a las redes neuronales son los basados en la idea de *descenso iterativo*:

> Comenzar con un valor inicial $\boldsymbol{w}(0)$ (aleatorio, si no se tiene más información) y generar una secuencia de vectores de pesos $\boldsymbol{w}(1), \boldsymbol{w}(2), \ldots$, tal que la función de error $\mathcal{E}(\boldsymbol{w})$ se reduzca en cada iteración del algoritmo, esto es,
>
> $$\mathcal{E}(\boldsymbol{w}(n+1)) < \mathcal{E}(\boldsymbol{w}(n)) \tag{1.10}$$

1.3.1. Descenso por gradiente

En esta forma de descenso iterativo, los sucesivos ajustes realizados al vector de pesos $\boldsymbol{w}$ se hacen en dirección opuesta al vector de gradiente $\boldsymbol{\nabla}\mathcal{E}(\boldsymbol{w})$:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \eta\boldsymbol{\nabla}\,\mathcal{E}(\boldsymbol{w}(n)) \tag{1.11}$$

donde η es la *tasa de aprendizaje*. Al pasar de la iteración n a la $n+1$, el algoritmo aplica la corrección

$$\Delta\boldsymbol{w}(n) = \boldsymbol{w}(n+1) - \boldsymbol{w}(n) = -\eta\,\boldsymbol{\nabla}\mathcal{E}(\boldsymbol{w}(n)) \tag{1.12}$$

En la formulación anterior se asume que el umbral b se trata como un peso más del vector $\boldsymbol{w}$.

Demostraremos ahora que la formulación del algoritmo de descenso por gradiente satisface la condición (1.10) del descenso iterativo. Por conveniencia en la notación, consideremos

$$\boldsymbol{g} = \boldsymbol{\nabla}\mathcal{E}(\boldsymbol{w}) \tag{1.13}$$

Mediante una expansión en series de Taylor de primer orden alrededor de $\boldsymbol{w}(n)$, podemos aproximar $\mathcal{E}(\boldsymbol{w}(n+1))$ como

$$\mathcal{E}(\boldsymbol{w}(n+1)) \approx \mathcal{E}(\boldsymbol{w}(n)) + \boldsymbol{g}^T(n)\Delta\boldsymbol{w}(n) \tag{1.14}$$

expresión justificada para valores pequeños de η. Sustituyendo la ecuación (1.12) en esta aproximación, obtenemos

$$\begin{aligned} \mathcal{E}(\boldsymbol{w}(n+1)) &\approx \mathcal{E}(\boldsymbol{w}(n)) - \eta\boldsymbol{g}^T(n)\boldsymbol{g}(n) && (1.15)\\ &= \mathcal{E}(\boldsymbol{w}(n)) - \eta\|\boldsymbol{g}(n)\|^2 && (1.16) \end{aligned}$$

que demuestra que para valores positivos de η la función de error se decrementa en cada iteración. El razonamiento presentado aquí es aproximado, ya que el resultado final es solo cierto para valores lo suficientemente pequeños de la tasa de aprendizaje.

La tasa de aprendizaje η tiene una enorme influencia en la convergencia del método de descenso por gradiente. Si η es pequeño, el proceso de aprendizaje se desarrolla suavemente, pero la convergencia del sistema a una solución estable puede llevar un tiempo excesivo. Si η es grande, la velocidad de aprendizaje aumenta, pero existe el riesgo de que el proceso de aprendizaje diverja y el sistema se vuelva inestable.

El método de descenso por gradiente tiene en su sencillez uno de sus mayores enemigos: cuando la superficie de error tiene mínimos locales, existe el riesgo de que el algoritmo quede atrapado en uno de ellos. Existen

otros métodos de optimización más sofisticados (por ejemplo, métodos que consideran la información suministrada por las derivadas de segundo orden), que, en general, proporcionan mejores resultados que el descenso por gradiente. Algunos de ellos son el método de Gauss-Newton, el algoritmo de Levenberg-Marquardt o el método de los gradientes conjugados.

1.3.2. Superficie de error

La representación gráfica de la función de error $\mathcal{E}$ en función de los pesos sinápticos $\boldsymbol{w}$ es una superficie multidimensional conocida como *superficie de error*. Según el tipo de neuronas utilizadas en la red neuronal, podemos identificar dos situaciones posibles:

- La red neuronal está formada en su totalidad por unidades lineales: en ese caso, la superficie de error es una función cuadrática con forma de cuenco y con un único punto mínimo.
- La red neuronal contiene unidades de procesamiento no lineales: en este caso, la superficie de error tiene un mínimo global (quizá múltiples mínimos globales) y mínimos locales.

El segundo caso es mucho más complejo de tratar. Por desgracia, la presencia de no linealidad es necesaria para resolver una enorme cantidad de problemas.

En ambos casos, el objetivo de un algoritmo de descenso por gradiente es comenzar en un punto arbitrario de la superficie de error y moverse paso a paso hacia un mínimo global. En el segundo caso, este objetivo puede no ser realizable con un algoritmo de entrenamiento determinado.

1.4. La regla delta

Consideremos una red monocapa con una única salida como la de la figura 1.1. Con una función de activación lineal, la salida de la red viene dada por

$$y = \sum_{j=1}^{m} w_j x_j + b \tag{1.17}$$

Esta sencilla red es capaz de representar una relación lineal entre el valor de la unidad de salida y el valor de las entradas. Supongamos que queremos entrenar la red neuronal para una tarea de aproximación de funciones de

forma que aprenda la asociación definida por el conjunto de entrenamiento $\{\boldsymbol{x}_i, d_i\}$, donde $\boldsymbol{x}_i$ es el i-ésimo vector de entrada y d_i es la salida deseada para este vector.

Supongamos que en cada iteración n se aplica a la red un vector $\boldsymbol{x}(n)$ del conjunto de entrenamiento. El orden en que se apliquen los distintos vectores es indiferente siempre que cualquier vector del conjunto de entrenamiento sea eventualmente utilizado; una de las formas más habituales es recorrer cíclicamente el conjunto de entrenamiento. Para este vector de entrada $\boldsymbol{x}(n)$ la salida proporcionada por la red difiere de la deseada en $e(n) = d(n) - y(n)$.

La regla delta utiliza una función de error en cada iteración definida por

$$\mathcal{E}(n) = \frac{1}{2}(d(n) - y(n))^2 \tag{1.18}$$

Aplicando el método de descenso por gradiente de la ecuación (1.12), el ajuste realizado sobre el peso w_j en la iteración n es

$$\Delta w_j(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_j(n)} \tag{1.19}$$

La derivada es, aplicando la regla de la cadena,

$$\frac{\partial \mathcal{E}(n)}{\partial w_j(n)} = \frac{\partial \mathcal{E}(n)}{\partial y(n)} \frac{\partial y(n)}{\partial w_j(n)} \tag{1.20}$$

A partir de la ecuación (1.17),

$$\frac{\partial y(n)}{\partial w_j(n)} = x_j(n) \tag{1.21}$$

donde $x_j(n)$ es la j-ésima componente del vector $\boldsymbol{x}(n)$ y

$$\frac{\partial \mathcal{E}(n)}{\partial y(n)} = -(d(n) - y(n)) \tag{1.22}$$

de manera que

$$\Delta w_j(n) = \eta(d(n) - y(n))x_j(n) \tag{1.23}$$

En cada iteración todos los pesos de la red se actualizan según la regla anterior. En el capítulo siguiente se analiza con un poco más de profundidad la regla delta.

Capítulo 2

Un problema de clasificación

En los próximos capítulos se realizan experimentos con cada técnica de clasificación propuesta. Para que los resultados sean comparables, se utiliza el mismo problema de clasificación en todos ellos. Aquí se plantea dicho problema.

El objetivo del experimento es distinguir entre dos clases de patrones bidimensionales distribuidos según dos gaussianas de distintas medias y varianzas. Las clases, por lo tanto, se solapan y este solapamiento será mayor o menor en función de esos parámetros.

La función de densidad de probabilidad condicionada para la clase $\mathcal{X}_1$ es (en este punto puede ser necesario echar un vistazo al apéndice A)

$$p(\boldsymbol{x}|\mathcal{X}_1) = \frac{1}{2\pi\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2}\|\boldsymbol{x}-\boldsymbol{\mu}_1\|^2\right) \tag{2.1}$$

donde

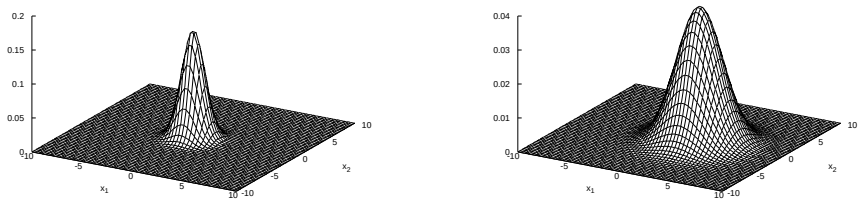
$$\begin{aligned} \boldsymbol{\mu}_1 &= \text{vector de medias} = [0,0]^T & (2.2)\\ \sigma_1^2 &= \text{varianza} = 1 & (2.3) \end{aligned}$$

y para la clase $\mathcal{X}_2$

$$p(\boldsymbol{x}|\mathcal{X}_2) = \frac{1}{2\pi\sigma_2^2} \exp\left(-\frac{1}{2\sigma_2^2}\|\boldsymbol{x}-\boldsymbol{\mu}_2\|^2\right) \tag{2.4}$$

donde

$$\begin{aligned} \boldsymbol{\mu}_2 &= \text{vector de medias} = [2,0]^T & (2.5)\\ \sigma_2^2 &= \text{varianza} = 4 & (2.6) \end{aligned}$$



(a) f.d.p. $p(\boldsymbol{x}|\mathcal{X}_1)$ (b) f.d.p. $p(\boldsymbol{x}|\mathcal{X}_2)$

Figura 2.1: Funciones de densidad de probabilidad de las dos clases del problema de clasificación.

Se supone que las dos clases son equiprobables, es decir, $p_1 = p_2 = 1/2$. El vector de entrada es $\boldsymbol{x} = [x_1, x_2]^T$ y la dimensión del espacio de entrada es $m = 2$.

En la figura 2.1 se muestran las distribuciones de probabilidad anteriores. En la figura 2.2, por otra parte, se muestran conjuntos de puntos tomados de ambas distribuciones. Las dos distribuciones se solapan significativamente, indicando que existe una probabilidad significativa de error en la clasificación.

2.1. Frontera de decisión bayesiana

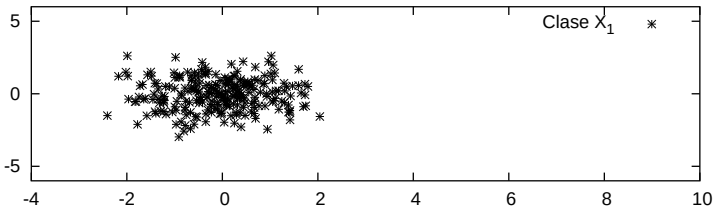
Siguiendo el apéndice A, tenemos que la frontera de decisión óptima se encuentra aplicando

$$\Lambda(\boldsymbol{x}) \underset{\mathcal{X}_2}{\overset{\mathcal{X}_1}{\gtrless}} \xi \tag{2.7}$$

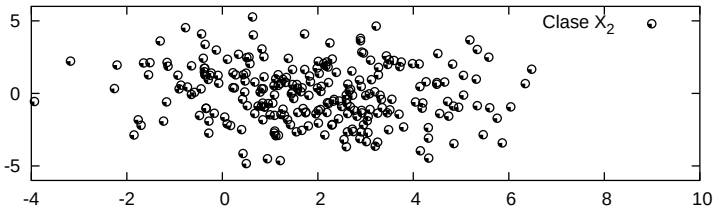
Para nuestro ejemplo, tenemos

$$\Lambda(\boldsymbol{x}) = \frac{\sigma_2^2}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2}\|\boldsymbol{x} - \boldsymbol{\mu}_1\|^2 + \frac{1}{2\sigma_2^2}\|\boldsymbol{x} - \boldsymbol{\mu}_2\|^2\right) \tag{2.8}$$

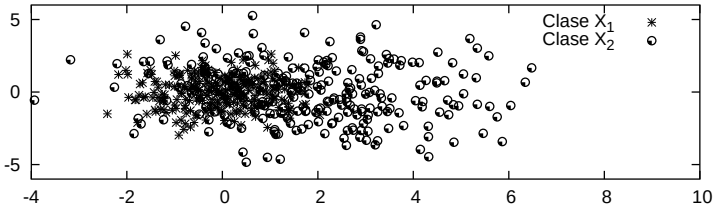
$$\xi = 1 \tag{2.9}$$



(a)



(b)



(c)

Figura 2.2: Puntos de las clases $\mathcal{X}_1$ y $\mathcal{X}_2$ generados bajo sus respectivas funciones de densidad de probabilidad. (a) Puntos de la clase $\mathcal{X}_1$. (b) Puntos de la clase $\mathcal{X}_2$. (c) Las dos gráficas anteriores superpuestas.

La frontera de decisión óptima está, por tanto, definida por

$$\frac{\sigma_2^2}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2}\|\boldsymbol{x}-\boldsymbol{\mu}_1\|^2 + \frac{1}{2\sigma_2^2}\|\boldsymbol{x}-\boldsymbol{\mu}_2\|^2\right) = 1 \tag{2.10}$$

o, de forma equivalente,

$$\frac{1}{\sigma_2^2}\|\boldsymbol{x}-\boldsymbol{\mu}_2\|^2 - \frac{1}{\sigma_1^2}\|\boldsymbol{x}-\boldsymbol{\mu}_1\|^2 = 4\log\left(\frac{\sigma_1}{\sigma_2}\right) \tag{2.11}$$

Operando, podemos redefinir la frontera de decisión óptima como

$$\|\boldsymbol{x}-\boldsymbol{x}_c\|^2 = r^2 \tag{2.12}$$

donde

$$\boldsymbol{x}_c = \frac{\sigma_2^2 \boldsymbol{\mu}_1 - \sigma_1^2 \boldsymbol{\mu}_2}{\sigma_2^2 - \sigma_1^2} \tag{2.13}$$

y

$$r^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_2^2 - \sigma_1^2} \left[\frac{\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2}{\sigma_2^2 - \sigma_1^2} + 4 \log \left(\frac{\sigma_2}{\sigma_1} \right) \right] \tag{2.14}$$

La ecuación (2.12) representa un círculo con centro $\boldsymbol{x}_c$ y radio r. Para los parámetros del experimento, la frontera de decisión circular tiene como centro

$$\boldsymbol{x}_c = [-2/3, 0]^T \tag{2.15}$$

y como radio

$$r \approx 2{,}34 \tag{2.16}$$

La probabilidad de error en la clasificación P_e puede obtenerse a partir de esta región circular mediante la ecuación (A.6), que puede expresarse como

$$P_e = p_1 P(e|\mathcal{X}_1) + p_2 P(e|\mathcal{X}_2) \tag{2.17}$$

donde cada $P(e|\mathcal{X}_i)$ es la probabilidad de error dado que la entrada al clasificador se tomó de la distribución de la clase $\mathcal{X}_i$. El cálculo de estas integrales es complejo y, normalmente, se realiza mediante métodos numéricos. Para nuestro problema, la evaluación numérica de las integrales de probabilidad nos da

$$P(e|\mathcal{X}_1) \approx 0{,}10545 \qquad P(e|\mathcal{X}_2) \approx 0{,}26433 \tag{2.18}$$

Con $p_1 = p_2 = 1/2$, tenemos $P_e \approx 0{,}1849$ y que la probabilidad óptima de clasificar correctamente es $P_c = 1 - P_e \approx 0{,}8151$.

2.2. Estimación del error

Después de entrenar un determinado clasificador neuronal, la probabilidad de acierto se obtiene simplemente evaluando la red neuronal entrenada sobre otro conjunto independiente de muestras tomadas aleatoriamente de las distribuciones. Sigamos considerando el caso de dos clases en el que ambas son equiprobables. Sea A una variable aleatoria que cuenta el número de patrones de los N patrones de evaluación que son clasificados correctamente. Entonces la razón

$$p_N = \frac{A}{N} \tag{2.19}$$

es una variable aleatoria que proporciona una estimación insesgada de máxima verosimilitud del rendimiento real p del clasificador. Si suponemos que

p es constante durante los N patrones, podemos aplicar la *cota de Chernoff* al estimador p_N de p para obtener

$$P(|p_N - p| > \epsilon) < 2\exp(-2\epsilon^2 N) = \delta \tag{2.20}$$

La aplicación de esta fórmula da $N \approx 26\,500$ para $\epsilon = 0{,}01$ y $\delta = 0{,}01$ (es decir, 99 % de certeza de que el estimador esté dentro de la tolerancia permitida). En los experimentos usaremos $N = 32\,000$ muestras de evaluación y mostraremos los resultados de la media de varias pruebas independientes de cada experimento.

Capítulo 3

Funciones discriminantes lineales

Aunque se trata de un modelo sencillo, una única neurona es capaz de resolver eficientemente algunos problemas de clasificación, en concreto aquellos en los que las muestras de dos clases pueden separarse mediante un hiperplano. Este capítulo estudia dos algoritmos clásicos de entrenamiento para este modelo de una neurona: el algoritmo perceptrón y el algoritmo de Widrow-Hoff. Además, se explica cómo extender los resultados para manejar problemas de clasificación de más de dos clases.

Los distintos algoritmos mostrados en este capítulo no son adecuados cuando el problema no es linealmente separable. Para estos casos, hay algoritmos, como los de Ho-Kashyap, capaces de detectar la no separabilidad, pero no se discutirán debido a que en capítulos posteriores se muestran algoritmos que no solo son capaces de detectarla, sino que, además, la manejan eficientemente.

3.1. Clasificación mediante discriminantes

Una neurona como la de la figura 1.1 implementa una función $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}+b$, si la función de activación es lineal, donde $\boldsymbol{x}$ es el vector de entrada, $\boldsymbol{w}$ es el vector de pesos y b es el umbral. Este sistema puede entrenarse para un problema de clasificación de dos clases, tomando φ como $\varphi(\boldsymbol{x}) = \boldsymbol{x}$ y

asignando un patrón de entrada a una de las dos clases según la regla

$$\begin{array}{rl} \text{decidir } \mathcal{X}_1 & \text{si } g(\boldsymbol{x}) > 0 \\ \text{decidir } \mathcal{X}_2 & \text{si } g(\boldsymbol{x}) < 0 \\ \text{cualquier clase} & \text{en otro caso} \end{array} \tag{3.1}$$

La ecuación $g(\boldsymbol{x}) = 0$ define la frontera de decisión que separa la clase $\mathcal{X}_1$ de la clase $\mathcal{X}_2$. Por ser $g(\boldsymbol{x})$ lineal, la frontera de decisión es un hiperplano H. Si $\boldsymbol{x}_1$ y $\boldsymbol{x}_2$ están sobre este hiperplano

$$\boldsymbol{w}^T\boldsymbol{x}_1 + b = \boldsymbol{w}^T\boldsymbol{x}_2 + b \tag{3.2}$$

o, lo que es lo mismo,

$$\boldsymbol{w}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0 \tag{3.3}$$

Por lo tanto, el vector de pesos $\boldsymbol{w}$ es normal a cualquier vector sobre el hiperplano. El hiperplano divide el espacio de entrada en dos regiones R_1 y R_2 (para $\mathcal{X}_1$ y $\mathcal{X}_2$). Como $g(\boldsymbol{x}) > 0$ si $\boldsymbol{x}$ está en R_1, se deduce que $\boldsymbol{w}$ apunta hacia R_1, llamado *lado positivo*.

Puede demostrarse que la función $g(\boldsymbol{x})$ proporciona una medida algebraica de la distancia r de $\boldsymbol{x}$ al hiperplano

$$g(\boldsymbol{x}) = r\|\boldsymbol{w}\| \tag{3.4}$$

o

$$r = \frac{g(\boldsymbol{x})}{\|\boldsymbol{w}\|} \tag{3.5}$$

En particular, la distancia del origen al hiperplano H está dada por $g(\mathbf{0})/\|\boldsymbol{w}\| = b/\|\boldsymbol{w}\|$. Si el umbral es cero, la función $g(\boldsymbol{x})$ adopta la forma homogénea $\boldsymbol{w}^T\boldsymbol{x}$ y el hiperplano pasa por el origen. La figura 3.1 muestra gráficamente las ideas anteriores.

3.1.1. El umbral como un peso más

El problema de tratar con un vector de pesos $\boldsymbol{w}$ y un umbral b puede simplificarse para manejar únicamente el primero. Para ello el umbral se trata como si fuera un peso sináptico asociado a una entrada fija de valor $+1$ como se muestra en la figura 3.2.

A partir de ahora, por tanto, consideraremos que el vector de entrada es un vector $m + 1$ dimensional

$$\boldsymbol{x} = [+1, x_1, x_2, \ldots, x_m]^T \tag{3.6}$$

y que el vector de pesos es

$$\boldsymbol{w} = [b, w_1, w_2, \ldots, w_m]^T \tag{3.7}$$

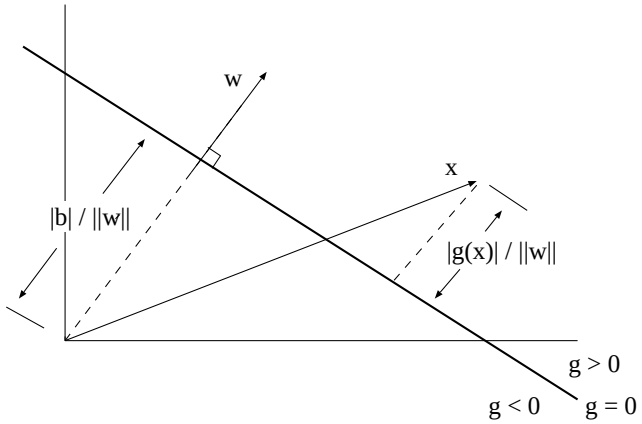


Figura 3.1: El hiperplano de separación $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$ y las distancias del hiperplano al origen y a un punto del espacio.

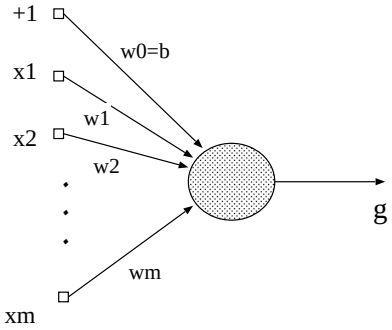
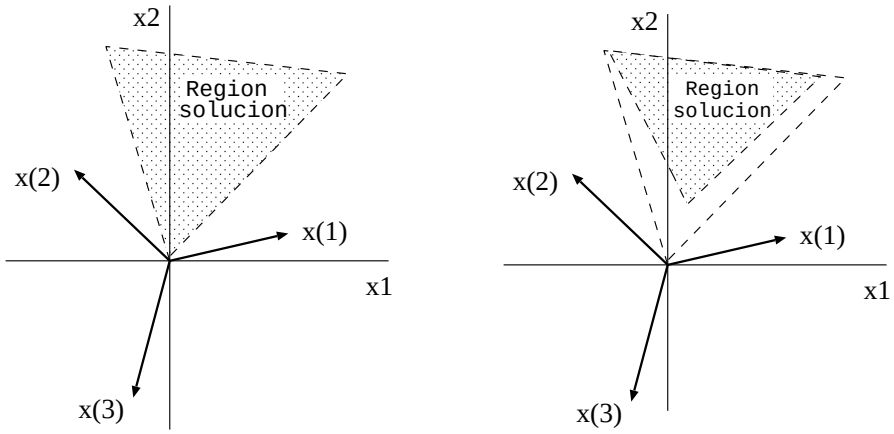


Figura 3.2: El ocultamiento del umbral b como un peso más del vector $\boldsymbol{w}$ se logra considerándolo como el peso de la sinapsis entre una entrada constante de valor $+1$ y la neurona.

3.2. Separabilidad lineal

Supongamos que tenemos un conjunto de muestras $\{\boldsymbol{x}_i, d_i\}$, $i = 1, \ldots, N$, donde d_i es una etiqueta que indica la clase a la que pertenece el patrón $\boldsymbol{x}_i$, por ejemplo, $d_i = +1$ si $\boldsymbol{x}_i$ pertenece a la clase $\mathcal{X}_1$ y $d_i = -1$ si $\boldsymbol{x}_i$ pertenece a la clase $\mathcal{X}_2$. El objetivo de las siguientes secciones es encontrar un vector de pesos $\boldsymbol{w}$ que haga que la neurona clasifique correctamente las muestras de entrenamiento según la regla (3.1). Si existe un vector de pesos (llamado *vector solución* o *vector separador*) que clasifica correctamente todas las muestras del entrenamiento, diremos que las muestras son *linealmente separables*.



(a) $c = 0$ (b) $c > 0$

Figura 3.3: Efecto del margen en la región solución. En este ejemplo, $\boldsymbol{x}_1$ y $\boldsymbol{x}_2$ pertenecen a $\mathcal{X}_1$ y $\boldsymbol{x}_3$ a $\mathcal{X}_2$.

Si $d_i \in \{-1, +1\}$, podemos decir que la neurona clasifica correctamente todas las muestras del entrenamiento con la condición más simple

$$d_i(\boldsymbol{w}^T \boldsymbol{x}_i) \geq 0 \qquad \text{para todo } i \tag{3.8}$$

El vector solución, si existe, no es único. Existe una región, denominada *región solución*, de posibles valores de $\boldsymbol{w}$ como se observa en la figura 3.3(a). Hay varias formas de imponer restricciones adicionales sobre el vector solución. Por ejemplo,

- podemos buscar un vector unitario que maximice la mínima distancia de las muestras al hiperplano separador;
- podemos buscar el vector de menor longitud que satisfaga

  $$d_i(\boldsymbol{w}^T \boldsymbol{x}_i) \geq c \qquad \text{para todo } i \tag{3.9}$$

  donde c es una constante positiva denominada *margen*. Normalmente suele buscarse solo la condición de esta ecuación y se relaja el requerimiento de longitud mínima. Esta solución, como se observa en la figura 3.3(b), está en el interior de la región solución anterior y alejada de la antigua frontera una distancia $c/\|\boldsymbol{x}_i\|$.

El motivo de estas restricciones es intentar encontrar un vector solución lo más cercano posible a la mitad de la región solución, con la esperanza

de que clasifique con mayor corrección muestras que no aparezcan durante el entrenamiento. Por ahora, nos preocuparemos únicamente por que los algoritmos de entrenamiento no converjan a una solución justo en el límite de la región solución. Este problema puede evitarse siempre con la introducción de un margen $c > 0$.

Como veremos al estudiar las máquinas de vectores soporte, es posible combinar las dos ideas anteriores para obtener un clasificador con un excelente rendimiento.

3.3. Algoritmo perceptrón

La mayor parte de los algoritmos de entrenamiento de una neurona presentados en este capítulo actúan bajo un enfoque de descenso por gradiente, ya explicado en 1.3.1. El algoritmo perceptrón es uno de los más sencillos que pueden utilizarse sobre una neurona para clasificar patrones linealmente separables. Este algoritmo apareció por primera vez en un trabajo desarrollado por Rosenblatt en el que describía un modelo de cerebro al que denominó perceptrón.

El algoritmo perceptrón es un algoritmo de descenso por gradiente y puede necesitar que se le presente más de una vez un determinado patrón del entrenamiento. Para adecuar la notación a este requerimiento, consideraremos una secuencia infinita de las muestras de entrenamiento en la que cada muestra aparezca infinitas veces, por ejemplo, repitiendo cíclicamente las muestras de entrenamiento. Con esto, consideraremos como conjunto de entrenamiento $\{\boldsymbol{x}(i), d(i) : i = 1, 2, \ldots, n, \ldots\}$ donde $\boldsymbol{x}_i$ es el vector de entrada (ampliado con una entrada fija a $+1$ para considerar el umbral como un peso más) aplicado en la iteración i-ésima del algoritmo y $d(i)$ es la salida esperada. Evidentemente, si en algún momento el algoritmo de entrenamiento converge, no es necesario continuar la presentación de patrones.

El algoritmo perceptrón para adaptar el vector de pesos puede formularse como sigue:

1. Si el vector $\boldsymbol{x}(n)$ es clasificado correctamente con el vector de pesos $\boldsymbol{w}(n)$, no se realiza ninguna corrección sobre el vector de pesos:

$$\Delta \boldsymbol{w}(n+1) = 0 \tag{3.10}$$

2. En otro caso, $\boldsymbol{x}(n)$ no es clasificado correctamente y el vector de pesos

se actualiza según la regla:

$$\begin{aligned} \Delta \boldsymbol{w}(n+1) &= -\eta(n)\boldsymbol{x}(n) \qquad \text{si } \boldsymbol{x}(n) \text{ pertenece a la clase } \mathcal{X}_2 \\ \Delta \boldsymbol{w}(n+1) &= +\eta(n)\boldsymbol{x}(n) \qquad \text{si } \boldsymbol{x}(n) \text{ pertenece a la clase } \mathcal{X}_1 \end{aligned} \tag{3.11}$$

donde la tasa de aprendizaje $\eta(n)$ controla el ajuste realizado sobre el vector $\boldsymbol{w}$ en la iteración n.

La actualización se produce únicamente cuando una muestra no se clasifica correctamente. Para decidir este condición, puede utilizarse la definición inicial de 3.8 o considerar la inclusión de un margen c como en 3.9.

Si $\eta(n) = \eta > 0$, donde η es una constante, tenemos la *la regla de adaptación con incremento fijo* del perceptrón. En el caso $\eta = 1$ la regla del perceptrón es muy sencilla: si un patrón $\boldsymbol{x}(n)$ no es clasificado correctamente, sumárselo o restárselo a $\boldsymbol{w}(n)$, según la clase a la que pertenezca sea $\mathcal{X}_1$ o $\mathcal{X}_2$, respectivamente.

El *teorema de convergencia del perceptrón* demuestra que si las muestras son linealmente separables, el algoritmo converge a una solución adecuada. No lo demostraremos aquí, pero daremos unas ideas generales sobre el porqué de esa convergencia.

Consideremos la regla de adaptación con incremento fijo con $\eta = 1$ y, sin pérdida de generalidad, consideremos una muestra mal clasificada $\boldsymbol{x}(n)$ perteneciente a la clase $\mathcal{X}_1$. Geométricamente, la suma de $\boldsymbol{x}(n)$ a $\boldsymbol{w}(n)$ mueve el vector de pesos hacia el hiperplano $\boldsymbol{w}(n)^T\boldsymbol{x}(n)$ e, incluso, puede ser que hasta el otro lado (véase la figura 3.4) del hiperplano. Como $\boldsymbol{w}(n)$ no clasifica bien $\boldsymbol{x}(n)$, $\boldsymbol{w}(n)$ no está en el lado positivo del hiperplano $\boldsymbol{w}(n)^T\boldsymbol{x}(n)$.

Independientemente de que se cruce el hiperplano, el nuevo producto escalar $\boldsymbol{w}(n+1)^T\boldsymbol{x}(n)$ es mayor que el viejo producto $\boldsymbol{w}(n)^T\boldsymbol{x}(n)$ en $\|\boldsymbol{x}(n)\|^2$ (recuérdese que $\eta = 1$) y, claramente, la corrección ha movido el vector de pesos en la dirección correcta.

Aunque un problema linealmente separable puede complicarse arbitrariamente haciendo que las muestras sean casi coplanares, el algoritmo encontrará siempre una solución en un número finito de pasos.

3.3.1. Incremento variable

Si utilizamos un valor variable para la tasa de aprendizaje $\eta(n)$, el algoritmo perceptrón convergerá si las muestras son linealmente separables y si

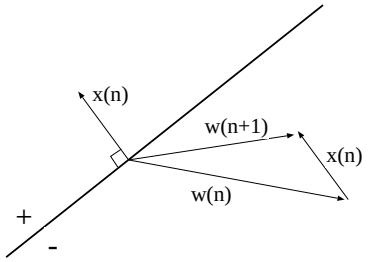


Figura 3.4: Una iteración de la regla de adaptación con incremento fijo. El vector de pesos se mueve en la dirección adecuada para corregir la clasificación errónea del vector $\boldsymbol{x}(n)$.

se cumplen las siguientes condiciones:

$$\eta(n) \geq 0 \tag{3.12}$$

$$\lim_{m\to\infty} \sum_{n=1}^{m} \eta(n) = \infty \tag{3.13}$$

$$\lim_{m\to\infty} \frac{\sum_{n=1}^{m} \eta(n)^2}{\left(\sum_{n=1}^{m} \eta(n)\right)^2} = 0 \tag{3.14}$$

En particular, estas condiciones se cumplen tanto si $\eta(n)$ es una constante positiva como si decrece con $1/n$.

Normalmente es aconsejable que $\eta(n)$ vaya decreciendo conforme aumenta n. Esto es especialmente útil si existen razones para creer que el conjunto de muestras no es linealmente separable.

3.4. Relación entre el perceptrón y un clasificador bayesiano

Cuando el entorno es gaussiano, un clasificador de Bayes como el estudiado en el apéndice A, se reduce a un clasificador lineal (la misma forma que la tomada por el perceptrón).

Consideremos un problema de clasificación con dos clases como el del apéndice en el que las funciones de densidad de probabilidad condicionales toman la forma

$$p(\boldsymbol{x}|\mathcal{X}_i) = \frac{1}{(2\pi)^{m/2}(\det(\boldsymbol{C}))^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right) \quad i = 1, 2 \tag{3.15}$$

donde cada clase tiene una media distinta $\boldsymbol{\mu}_i$, pero la misma matriz de covarianza $\boldsymbol{C}$, y m es la dimensión del vector de observación $\boldsymbol{x}$. Supongamos que la matriz de covarianza $\boldsymbol{C}$ es no diagonal y no singular (las muestras de ambas clases están correlacionadas).

Sustituyendo la ecuación (3.15) en la ecuación (A.2), tomando logaritmos y simplificando, obtenemos

$$\begin{aligned} \log \Lambda(\boldsymbol{x}) &= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + (\boldsymbol{x}-\boldsymbol{\mu}_2)^T\boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) \\ &= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\boldsymbol{C}^{-1}\boldsymbol{x} + \frac{1}{2}(\boldsymbol{\mu}_2^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_1) \end{aligned} \tag{3.16}$$

Si, además, suponemos que las dos clases $\mathcal{X}_1$ y $\mathcal{X}_2$ son equiprobables, $p_1 = p_2 = 1/2$, de la ecuación (A.3) obtenemos

$$\log \xi = 0 \tag{3.17}$$

Las dos ecuaciones anteriores muestran que el clasificador de Bayes que estamos considerandio es un *clasificador lineal* como el descrito por

$$z = \boldsymbol{w}^T\boldsymbol{x} + b \tag{3.18}$$

donde

$$z = \log \Lambda(\boldsymbol{x}) \tag{3.19}$$

$$\boldsymbol{w} = \boldsymbol{C}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{3.20}$$

$$b = \frac{1}{2}(\boldsymbol{\mu}_2^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\boldsymbol{C}^{-1}\boldsymbol{\mu}_1) \tag{3.21}$$

Por lo tanto, siguiendo la ecuación (A.5), podemos definir nuestro clasificador como: si la salida z del combinador lineal (incluido el umbral b) es positiva, asignar la observación $\boldsymbol{x}$ a la clase $\mathcal{X}_1$; en otro caso, asignarla a la clase $\mathcal{X}_2$.

A pesar de este resultado, hay diferencias importantes entre el clasificador de Bayes para un entorno gaussiano y el perceptrón:

- El perceptrón opera bajo la suposición de la separabilidad lineal; si esta no existe, la frontera de decisión puede oscilar continuamente. Por otra parte, las distribuciones gaussianas se solapan y no son separables; como se vio en el apéndice (figura A.1), un clasificador bayesiano minimiza siempre la probabilidad del error de clasificación.

- El algoritmo de entrenamiento del perceptrón es no paramétrico en el sentido de que no supone ninguna forma de las distribuciones involucradas. El clasificador gaussiano es paramétrico y tiene por ello un área de aplicación limitada.

- Finalmente, el perceptrón es adaptativo y sencillo de implementar. El diseño del clasificador gaussiano es fijo; puede hacerse adaptativo a costa de incrementar el número de cálculos y el espacio en memoria necesario.

3.5. Algoritmos de mínimos cuadrados

Una neurona constituye también la base de un *filtro lineal adaptativo*, un bloque funcional básico en el campo del *procesamiento de señales*. El desarrollo del filtrado adaptativo debe mucho al trabajo de Widrow y Hoff en el que presentaron la *regla delta* o *algoritmo de mínimos cuadrados*.

El problema del filtrado adaptativo asume que los datos $\{\boldsymbol{x}(n), d(n) : i = 1, 2, \ldots, n \ldots\}$ han sido generados en instantes discretos de tiempo por un sistema dinámico cuya caracterización matemática es desconocida. El problema es cómo diseñar un modelo del sistema desconocido con una única neurona lineal.

Definimos el vector de error $\boldsymbol{e}(n)$ como

$$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{X}(n)\boldsymbol{w}(n) \tag{3.22}$$

donde $\boldsymbol{d}(n)$ es el vector de respuestas deseadas

$$\boldsymbol{d}(n) = [d(1), d(2), \ldots, d(n)]^T \tag{3.23}$$

y $\boldsymbol{X}(n)$ es la matriz de datos $n \times m$ definida por

$$\boldsymbol{X}(n) = [\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(n)]^T \tag{3.24}$$

Puede demostrarse (aplicando el método de minimización de Gauss-Newton) que el vector de pesos que minimiza el error $\boldsymbol{e}(n)$ viene dado por la expresión

$$\begin{aligned} \boldsymbol{w}(n+1) &= (\boldsymbol{X}^T(n)\boldsymbol{X}(n))^{-1}\boldsymbol{X}^T(n)\boldsymbol{d}(n) \\ &= \boldsymbol{X}^+(n)\boldsymbol{d}(n) \end{aligned} \tag{3.25}$$

donde $\boldsymbol{X}^+$ es la pseudoinversa de $\boldsymbol{X}$. Si $\boldsymbol{X}$ es cuadrada y no singular, la pseudoinversa coincide con la inversa. Nótese también que $\boldsymbol{X}^+\boldsymbol{X} = \boldsymbol{I}$, pero que, en general, $\boldsymbol{X}\boldsymbol{X}^+ \neq \boldsymbol{I}$.

La ecuación 3.25 es una forma conveniente de expresar la idea de que el vector de pesos $\boldsymbol{w}(n+1)$ resulve el problema lineal de mínimos cuadrados definido sobre un intervalo de observación de duración n.

No obstante, el cálculo de la matriz pseudoinversa puede ocasionar problemas si la matriz $\boldsymbol{X}^T\boldsymbol{X}$ es singular. Además, su uso implica trabajar con grandes matrices lo que repercute negativamente en el coste de algoritmo. En la siguiente sección resolveremos el problema con un método de descenso por gradiente.

3.5.1. Algoritmo de Widrow-Hoff

La regla delta es una forma sencilla de abordar el problema del filtrado lineal basada en la técnica del descenso por gradiente. La regla delta, también conocida como *algoritmo de Widrow-Hoff*, fue discutida en la sección 1.4 y no se repetirá aquí. La formulación allí empleada utilizaba una tasa de aprendizaje η constante. Dando valores bajos a η, el proceso adaptativo progresa lentamente y el algoritmo recuerda gran parte de los datos del pasado. Para valores grandes de η, las nuevas observaciones influyen con más fuerza en el nuevo vector de pesos obtenido. En otras palabras, el valor inverso de η es una medida de la *memoria* del algoritmo.

La regla delta obtiene un buen rendimiento conforme el número de iteraciones se acerca a infinito, ya que en ese caso $\boldsymbol{w}(n)$ realiza un paseo aleatorio en torno a la solución de Wiener (una solución óptima, si se conoce ciertas estadísticas del entorno en forma de matrices de correlación y vectores de correlación cruzada, lo cual no suele ser el caso). En este sentido, representa un buen compromiso cuando los datos pueden ser no linealmente separables, ya que minimiza el error cuadrático medio.

3.5.2. Esquemas de variación de la tasa de aprendizaje

Algunos de los problemas que pueden aparecer en el uso de la regla delta pueden atribuirse al hecho de que la tasa de aprendizaje se mantiene constante

$$\eta(n) = \eta_0 \qquad \text{para todo } n \tag{3.26}$$

Sin embargo, nada impide que la tasa de aprendizaje varíe con el tiempo. Una alternativa es

$$\eta(n) = \frac{c}{n} \tag{3.27}$$

donde c es una constante. Esta elección es suficiente para garantizar la convergencia, pero cuando c es grande y n pequeño, existe el peligro de que los

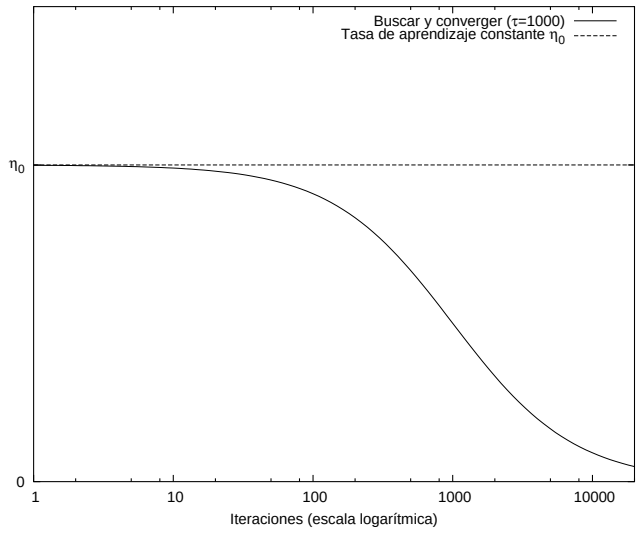


Figura 3.5: Comparación de la evolución de la tasa de aprendizaje en el esquema *buscar y converger* con una tasa de aprendizaje constante η_0.

pesos se inflen excesivamente.

Una alternativa es el *esquema buscar y converger*, que se define con

$$\eta(n) = \frac{\eta_0}{1 + (n/\tau)} \tag{3.28}$$

donde η_0 y τ son constantes ajustadas por el usuario. En las primeras fases del algoritmo, con un valor de n pequeño comparado con la *constante de tiempo de búsqueda* τ, la tasa de aprendizaje $\eta(n)$ es aproximadamente igual a η_0 y el algoritmo opera como la regla delta tradicional. Cuando $\eta \gg \tau$, $\eta(n)$ tiende a c/n donde $c = \tau\eta_0$. La evolución de la tasa de aprendizaje con este esquema puede observarse en la figura 3.5.

3.6. Múltiples clases

Todos los resultados presentados en este capítulo utilizaban un problema de clasificación con dos clases posibles, $\mathcal{X}_1$ y $\mathcal{X}_2$. Para resolver un problema de clasificación con M clases, $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$ necesitamos M neuronas de salida.

El conjunto de datos de entrenamiento es ahora $\{\boldsymbol{x}_i, \boldsymbol{d}_i\}$, donde $\boldsymbol{x}_i$ es como antes, pero la salida esperada $\boldsymbol{d}_i$ es ahora un vector de M componentes. Supongamos que las funciones de activación de la capa de salida (la única

					Media
65,93	65,40	67,81	75,66	50,74	65,11

Cuadro 3.1: Porcentajes de aciertos del clasificador de una única neurona.

capa en este capítulo, aunque se habla de forma genérica porque la técnica de representación aquí dada será valida en posteriores capítulos) están acotadas en $[s_0, s_1]$ (por ejemplo, $s_0 = -1$ y $s_1 = 1$ en el caso de la función escalón). Entonces, si el vector de entrada $\boldsymbol{x}_i$ pertenece a la clase k con $1 \leq k \leq M$, la salida esperada $\boldsymbol{d}_i$ se codifica con valores binarios como

$$d_{ji} = \begin{cases} s_0 & \text{si } j \neq k \\ s_1 & \text{si } j = k \end{cases} \tag{3.29}$$

Con esta codificación, la salida de la red neuronal puede interpretarse como una aproximación a las probabilidades a posteriori de las distintas clases, siempre y cuando el aprendizaje se haya realizado con éxito. Siguiendo la regla de Bayes con estas probabilidades aproximadas, durante la evaluación de la red neuronal diremos que un patrón $\boldsymbol{x}$ pertenece a la clase $\mathcal{X}_k$ si la salida de la k-ésima neurona de salida es mayor que la de las $M-1$ neuronas restantes.

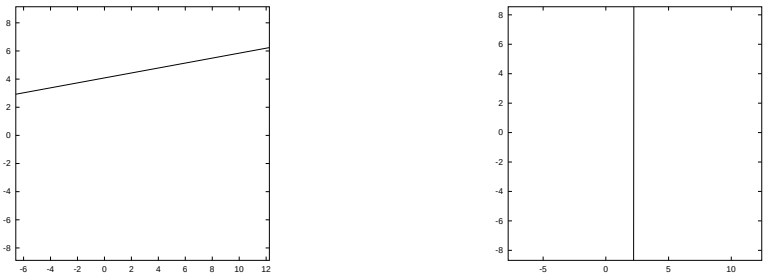
3.7. Experimentos

En esta sección aplicaremos el algoritmo perceptrón al problema de clasificación del capítulo 2. Los datos de este problema son claramente no separables, por lo que las correciones que realiza el algoritmo no cesarán nunca si se utiliza una tasa de aprendizaje constante. Para superar este inconveniente, hacemos que $\eta(n)$ tienda a cero y paramos el algoritmo tras un determinado número de iteraciones.

El cuadro 3.1 muestra el porcentaje de aciertos al clasificar 32 000 muestras pertenecientes de forma equiprobable a las clases $\mathcal{X}_1$ y $\mathcal{X}_2$ en 5 experimentos independientes. En cada experimento se utilizó una neurona entrenada bajo un conjunto distinto de entrenamiento de 200 patrones. Las muestras de evaluación, sin embargo, eran las mismas en todos los casos. Todos los algoritmos se detuvieron tras 1000 iteraciones.

Se utilizó el esquema *buscar y converger* para variar la tasa de aprendizaje con $\eta_0 = 0{,}1$ y $\tau = 100$. Además, se consideró un margen nulo, por no ser este demasiado útil en un problema de estas características.

Los resultados demuestran que la tasa media de aciertos conseguida



(a) 50,74 % (b) 72,63 %

Figura 3.6: Fronteras de decisión bayesiana (círculo) y neuronal en 2 experimentos distintos. Bajo cada figura se indica el porcentaje de aciertos asociado al clasificador neuronal.

(65, 11 %) está muy lejos de la de Bayes (81,51 por ciento), lo cual era de esperar por el alto grado de no separabilidad del problema.

En la figura 3.6 se compara la frontera de decisión circular obtenida con el clasificador de Bayes en el capítulo 2 con la frontera lineal definida por modelos de una neurona entrenados mediante el algoritmo perceptrón.

Capítulo 4

Perceptrón multicapa

Como vimos en el capítulo 3, una red neuronal monocapa sin ciclos puede aprender sin problemas un número muy limitado de tareas. En 1969, Minsky y Papert demostraron que una red sin ciclos con una capa oculta y funciones de activación no lineales (por ejemplo, la función logística) podía superar muchas de estas restricciones, pero no presentaron una solución al problema de cómo ajustar los pesos de las capas ocultas. Casi 20 años depués, en 1986, Rumelhart, Hinton y Willians publicaron el algoritmo de *retropropagación*, que indicaba una forma de ajustar estos pesos.

La idea central del algoritmo de retropropagación es ajustar los pesos que conectan la última capa oculta con la capa de salida mediante la regla delta discutida en el capítulo 1 y propagar el error *hacia atrás* para actualizar los pesos de las restantes capas. Por esta razón también se conoce al método con el nombre de *regla delta generalizada*.

Se ha demostrado que, si las funciones de activación de las unidades ocultas son no lineales, una única capa oculta es suficiente para aproximar cualquier función con una precisión arbitraria (teorema de aproximación universal). El gran problema es encontrar los pesos para conseguirlo.

En este capítulo no presentaremos ningún desarrollo teórico relativo al perceptrón multicapa (PMC). Hay abundante bibliografía sobre el tema y solo nos interesa el poder de clasificación del perceptrón multicapa para compararlo con el de otros métodos.

				Media
79,94	80,05	80,45	80,35	80,20

Cuadro 4.1: Porcentajes de aciertos del clasificador basado en un perceptrón multicapa.

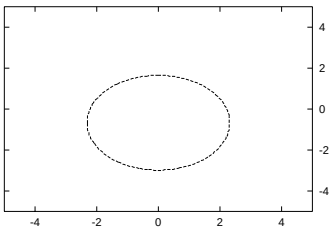


Figura 4.1: La frontera de decisión circular del clasificador óptimo de Bayes y la de un perceptrón multicapa entrenado según se describe en la sección de experimentos. El porcentaje de aciertos sobre el conjunto de evaluación fue de 79,94 %.

4.1. Experimentos

En el cuadro 4.1 se muestran los resultados obtenidos al utilizar un perceptrón multicapa para el problema de clasificación del capítulo 2. Puede observarse cómo la tasa de acierto media (80,20 %) se acerca mucho a la óptima, sobre todo si se compara con la obtenida en el caso de los discriminantes lineales.

Los conjuntos de entrenamiento estaban formados por 1 000 patrones y el conjunto de evaluación por 32 000. Se utilizaron los parámetros sugeridos en [4], esto es, una tasa de aprendizaje de 0,1, un momento de 0,5 y 2 neuronas en la capa oculta. El algoritmo de retropropagación se ejecutó durante 600 épocas sobre 4 conjuntos independientes de entrenamiento.

En la figura 4.1 se observa la frontera de decisión (no lineal) definida por uno de los perceptrones multicapa. La no linealidad de la frontera de decisión permite mejorar notablemente el rendimiento con respecto al modelo lineal de una neurona.

Capítulo 5

Funciones de base radial

En este capítulo se ve el aprendizaje como la búsqueda en un espacio multidimensional de la superficie que mejor se adapte al conjunto de datos de entrenamiento, donde «mejor» se interpreta según una determinada medida. De esta manera, la generalización es equivalente al uso de esta superficie para interpolar los datos de evaluación. El método de funciones de base radial se alimenta, por tanto, de los trabajos en interpolación estricta clásica sobre un espacio multidimensional.

Una red de funciones de base radial se muestra en la figura 5.1. La capa oculta, formada por m_1 funciones de base radial, aplica una transformación no lineal desde el espacio de entrada m dimensional al espacio oculto (espacio de características), que es normalmente de mayor dimensión. La capa de salida opera linealmente con los vectores de este nuevo espacio.

La justificación matemática de la idoneidad de una transformación no lineal se encuentra en trabajo de Cover de 1965. Básicamente, este venía a decir que un problema de clasificación realizado en un espacio de dimensión alta tiene más posibilidades de ser linealmente separable que uno llevado a cabo en un espacio de dimensión menor. Este resultado, dado aquí en términos cualitativos, se conoce como teorema de Cover sobre la separabilidad de patrones y se aborda con más detalle a continuación.

5.1. Teorema de Cover

Por lo que se vio en el capítulo 3, sabemos que cuando los patrones son linealmente separables, el problema de la clasificación es relativamente sencillo de resolver.

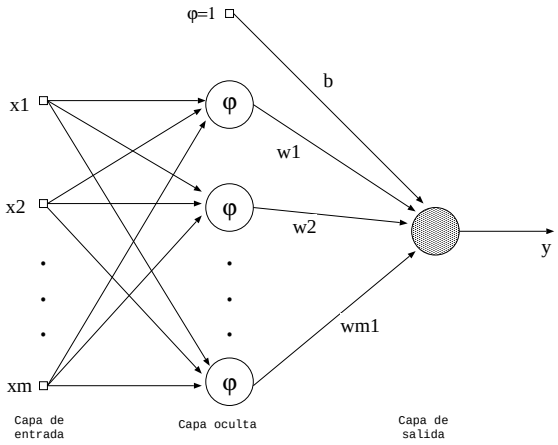


Figura 5.1: Una red de funciones de base radial.

Sea $\mathcal{X}$ un conjunto de N muestras (vectores) de entrenamiento $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, cada una de los cuales pertenece a una de las clases $\mathcal{X}_1$ o $\mathcal{X}_2$. Para cada muestra $\boldsymbol{x} \in \mathcal{X}$ definamos un vector formado por un conjunto de funciones reales $\{\varphi_i(\boldsymbol{x}) : i = 1, \ldots, m_1\}$ como sigue:

$$\boldsymbol{\varphi}(\boldsymbol{x}) = [\varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}), \ldots, \varphi_{m_1}(\boldsymbol{x})]^T \tag{5.1}$$

Si $\boldsymbol{x}$ es un vector del espacio de entrada m_0 dimensional, la transformación $\boldsymbol{\varphi}(\boldsymbol{x})$ lo lleva a un nuevo espacio de dimensión m_1. Diremos que cada $\varphi_i(\boldsymbol{x})$ es una *función oculta* y al espacio generado por $\boldsymbol{\varphi}(\boldsymbol{x})$ lo llamaremos *espacio de características espacio oculto.*

Se dice que los vectores de $\mathcal{X}$ son separables-φ si existe un vector m_1 dimensional $\boldsymbol{w}$ que cumple

$$\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}) > 0 \qquad \boldsymbol{x} \in \mathcal{X}_1 \tag{5.2a}$$
$$\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}) < 0 \qquad \boldsymbol{x} \in \mathcal{X}_2 \tag{5.2b}$$

El hiperplano definido por la ecuación $\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}) = 0$ describe la frontera de decisión en el espacio oculto. La imagen inversa de este hiperplano, esto es,

$$\boldsymbol{x} : \quad \boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}) = 0 \tag{5.3}$$

define la frontera de decisión en el espacio de entrada.

Supongamos que los patrones de entrada $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ se eligen independientemente según alguna distribución de probabilidad sobre el espacio de entrada. Supongamos también que todas las posibles *dicotomías* (asignaciones a los patrones de etiquetas que determinen la clase a la que

pertenecen) de $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ son equiprobables y que las funciones $\varphi_i(\boldsymbol{x})$ son polinómicas en $\boldsymbol{x}$. Si $P(N, m_1)$ es la probabilidad de que una dicotomía elegida al azar sea separable-φ, entonces el teorema de Cover afirma que

$$P(N, m_1) = \left(\frac{1}{2}\right)^{N-1} \sum_{k=0}^{m_1-1} \binom{N-1}{k} \tag{5.4}$$

Aunque la ecuación anterior necesita que las unidades ocultas sean polinómicas y, por tanto, diferentes de las que se utilizan habitualmente en las redes de funciones de base radial, el contenido esencial de la ecuación tiene aplicación general: cuanto mayor sea la dimensión m_1, más cercana estará a uno la probabilidad $P(N, m_1)$. En definitiva, el teorema de Cover destaca dos aspectos:

1. la formulación no lineal de $\varphi_i(\boldsymbol{x})$ con $i = 1, \ldots, m_1$,
2. la elevada dimensión del espacio oculto respecto del espacio de entrada.

En algunos casos, el uso de transformaciones no lineales (punto 1) es suficiente para producir la separabilidad.

5.2. Interpolación

Consideremos la red neuronal de la figura 5.1. Esta red representa una transformación

$$s : \mathbb{R}^{m_0} \longmapsto \mathbb{R}^1 \tag{5.5}$$

Podemos imaginar la transformación s como una hipersuperficie $\Gamma \subset \mathbb{R}^{m_0+1}$. En la práctica Γ es desconocida y los datos de entrenamiento suelen estar contaminados con ruido. Por tanto, el entrenamiento y la generalización del proceso de aprendizaje de la red puede verse como un problema de interpolación. El problema de la interpolación (estricta) puede formularse como: *dado un conjunto de N puntos distintos $\{\boldsymbol{x}_i \in \mathbb{R}^{m_0} : i = 1, \ldots, N\}$ y un conjunto asociado de N números $\{d_i \in \mathbb{R}^1 : i = 1, \ldots, N\}$, encontrar una función $F : \mathbb{R}^{m_0} \longrightarrow \mathbb{R}^1$ que satisfaga la condición de interpolación*

$$F(\boldsymbol{x}_i) = d_i \quad i = 1, \ldots, N \tag{5.6}$$

La técnica basada en funciones de base radial consiste en elegir una función F que tenga la forma

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \tag{5.7}$$

donde $\{\varphi(\|\boldsymbol{x}-\boldsymbol{x}_i\|), i = 1, \ldots, N\}$ es un conjunto de N funciones no lineales llamadas *funciones de base radial*. Los puntos $\boldsymbol{x}_i \in \mathbb{R}^{m_o}$ se toman como *centros* de las funciones de base radial.

Las dos ecuaciones anteriores pueden combinarse en la forma compacta

$$\boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{d} \tag{5.8}$$

donde

$$\boldsymbol{d} = [d_1, \ldots, d_N]^T \tag{5.9}$$

$$\boldsymbol{w} = [w_1, \ldots, w_N]^T \tag{5.10}$$

y $\boldsymbol{\Phi}$ es una matriz $N \times N$ llamada *matriz de interpolación* con elementos

$$\boldsymbol{\Phi} = \{\varphi_{ji} = \varphi(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|) : (j, i) = 1, \ldots, N\} \tag{5.11}$$

Si $\boldsymbol{\Phi}$ no es singular, podemos resolver la ecuación (5.8) y obtener $\boldsymbol{w}$ como

$$\boldsymbol{w} = \boldsymbol{\Phi}^{-1}\boldsymbol{d} \tag{5.12}$$

El problema ahora es cómo asegurarnos de que la matriz de interpolación $\boldsymbol{\Phi}$ no es singular. El teorema de Micchelli demuestra que si $\{\boldsymbol{x}_i\}_{i=1}^N$ es un conjunto de puntos *distintos*, entonces la matriz de interpolación es no singular si φ es, entre otras, una de las siguientes funciones:

1. Multicuádrica

$$\varphi(r) = (r^2 + c^2)^{1/2} \qquad \text{con } c > 0 \text{ y } r \in \mathbb{R} \tag{5.13}$$

2. Multicuádrica inversa

$$\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \qquad \text{con } c > 0 \text{ y } r \in \mathbb{R} \tag{5.14}$$

3. Función gaussiana

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad \text{con } \sigma > 0 \text{ y } r \in \mathbb{R} \tag{5.15}$$

5.3. Teoría de la regularización

El procedimiento de interpolación descrito anteriormente puede no ser una buena estrategia para el entrenamiento de redes de funciones de base radial (RFBR) debido a la pobre generalización que presentan con nuevos

datos en ciertos problemas y al sobreajuste. El problema que se trata de resolver es lo que se denomina en matemáticas un problema *mal planteado*, lo que implica básicamente que en una gran cantidad de datos se puede encontrar una cantidad de información sorprendentemente pequeña sobre la solución deseada. El problema de la reconstrucción de una superficie Γ está mal planteado por razones tales como:

- no continuidad (*inestabilidad*) de la transformación,
- falta de información en las muestras de entrenamiento para reconstruir correctamente la transformación de entrada-salida,
- inexistencia de una salida diferente para cada entrada.

Muy especialmente, si un problema carece de la propiedad de continuidad, entonces la transformación de entrada-salida calculada no tiene nada que ver con la solución correcta. No hay forma de superar este inconveniente a menos que se disponga de alguna información a priori sobre la transformación de entrada-salida.

En 1963, Tikhonov propuso un nuevo método denominado *regularización* para resolver problemas mal planteados convirtiéndolos en *bien planteados*. En el contexto de nuestro problema de recontrucción de hipersuperficies, la idea básica de la regularización es *estabilizar* la solución a través de algún funcional auxiliar no negativo que guarde información a priori sobre la solución: la suposición más habitual es que la función de transformación es *suave* en el sentido de que a similares entradas les corresponden salidas similares.

No se presentará aquí el desarrollo de la teoría de la regularización y su aplicación a la reconstrucción de hipersuperficies. El resultado final que se obtiene es que la solución al problema de la regularización viene dada por la expansión

$$F_\lambda(\boldsymbol{x}) = \sum_{i=1}^{N} w_i G(\boldsymbol{x}, \boldsymbol{x}_i) \tag{5.16}$$

donde $G(\boldsymbol{x}, \boldsymbol{x}_i)$ es una *función de Green* asociada a la suposición a priori realizada sobre el modelo. Las funciones de Green que son interesantes en la práctica son de la forma

$$G(\boldsymbol{x}, \boldsymbol{x}_i) = G(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \tag{5.17}$$

En este caso las funciones de Green se denominan *funciones de base radial* y la solución regularizada de la ecuación (5.16) se convierte en

$$F_\lambda(\boldsymbol{x}) = \sum_{i=1}^{N} w_i G(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \tag{5.18}$$

Un ejemplo de función de Green de este tipo es la gaussiana multivariante definida por

$$G(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\left(-\frac{1}{2\sigma_i^2}\|\boldsymbol{x} - \boldsymbol{x}_i\|^2\right) \tag{5.19}$$

donde para simplificar normalmente se toma $\sigma_i = \sigma$.

Si definimos $\boldsymbol{d}$ como en la ecuación (5.9), $\boldsymbol{w}$ como en la ecuación (5.10) y $\boldsymbol{G}$ como

$$\boldsymbol{G} = \begin{bmatrix} G(\boldsymbol{x}_1, \boldsymbol{x}_1) & G(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & G(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ G(\boldsymbol{x}_2, \boldsymbol{x}_1) & G(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & G(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \cdots & \cdots & \cdots & \cdots \\ G(\boldsymbol{x}_N, \boldsymbol{x}_1) & G(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & G(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \tag{5.20}$$

entonces los coeficientes desconocidos de las ecuaciones (5.16) y (5.18) se pueden determinar con

$$\boldsymbol{w} = (\boldsymbol{G} + \lambda \boldsymbol{I})^{-1}\boldsymbol{d} \tag{5.21}$$

Después de decidir la función de Green a utilizar, podemos usar la ecuación anterior para obtener el vector de pesos para una respuesta deseada concreta $\boldsymbol{d}$ y para un valor apropiado del *parámetro de regularización* λ.

El parámetro de regularización λ puede verse como un indicador de la suficiencia de los datos del entrenamiento como ejemplos que especifican la solución correcta de $F_\lambda(\boldsymbol{x})$. En particular, el caso límite $\lambda \to 0$ implica que el problema no tiene restricciones y que la solución $F_\lambda(\boldsymbol{x})$ se determina completamente a partir de los ejemplos. El otro caso extremo, $\lambda \to \infty$, implica que la restricción de suavidad inicial es por sí misma suficiente para especificar la solución $F_\lambda(\boldsymbol{x})$ o, lo que es lo mismo, que las muestras no son nada fiables. En situaciones prácticas se le asigna a λ un valor intermedio para que tanto los datos de entrenamiento como la información a priori contribuyan a la solución $F_\lambda(\boldsymbol{x})$.

Como puede verse, la matriz de Green $\boldsymbol{G}$ juega en la teoría de regularización un papel similar al de $\boldsymbol{\Phi}$ en la teoría de interpolación con funciones de base radial. Tanto $\boldsymbol{G}$ como $\boldsymbol{\Phi}$ son matrices simétricas de tamaño $N \times N$.

La solución de la ecuación (5.18) se denomina *interpolación estricta*, ya que todos los puntos disponibles en el entrenamiento se usan en la generación de la función de interpolación $F(\boldsymbol{x})$. Es importante señalar que esta solución difiere de la de la ecuación (5.6) en un aspecto fundamental: la solución de la ecuación (5.18) está regularizada por efecto de la definición dada en la ecuación (5.21) para el vector de pesos $\boldsymbol{w}$. Las dos soluciones son idénticas solo cuando el parámetro de regularización se iguala a cero.

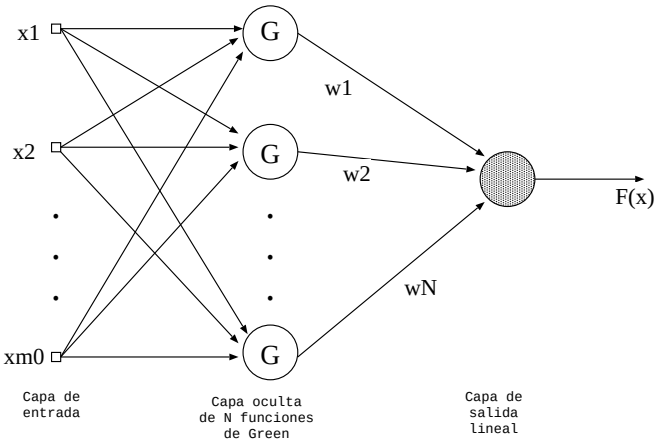


Figura 5.2: Una red de regularización.

5.4. Redes de regularización

La expansión de la función de aproximación regularizada $F_\lambda(\boldsymbol{x})$ de la ecuación (5.16), dada en términos de las funciones de Green $G(\boldsymbol{x}, \boldsymbol{x}_i)$ centradas en $\boldsymbol{x}_i$, sugiere la arquitectura neuronal de la figura 5.2. Por razones obvias, a esta red se le denomina *red de regularización*. La salida de la i-ésima unidad oculta es $G(\boldsymbol{x}, \boldsymbol{x}_i)$.

Claramente, esta arquitectura puede extenderse para acoger cualquier número de salidas (en el caso de clasificación, debería seguirse la codificación sugerida en 3.6). En la red de regularización de la figura 5.2 se asume que todas las funciones de Green son definidas positivas. Si esta condición se cumple (lo cual es el caso si $G(\boldsymbol{x}, \boldsymbol{x}_i)$ tiene la forma gaussiana de la ecuación (5.19), por ejemplo), entonces la solución producida por la red sera óptima en el sentido de que minimiza un funcional que mide cuánto se aleja la solución de su valor real (G está asociada con dicho funcional). La red de regularización es un *aproximador universal* porque puede aproximar bien cualquier función continua sobre un subconjunto compacto de $\mathbb{R}^{m_0}$, dado un número suficiente grande de unidades ocultas.

5.5. Redes de funciones de base radial generalizadas

Las redes de regularización pueden ser muy costosas de implementar para valores grandes de N (inversión de una matriz $N \times N$). Para evitar

este problema se sigue un enfoque distinto en el que se busca una solución subóptima en un espacio de dimensión menor, que aproxime la solución regularizada de la ecuación (5.16). Según esta técnica, conocida como *método de Galerkin*, la solución aproximada $F^*(\boldsymbol{x})$ se expande como

$$F^*(\boldsymbol{x}) = \sum_{i=1}^{m_1} w_i \varphi_i(\boldsymbol{x}) \tag{5.22}$$

donde $\{\varphi_i(\boldsymbol{x}) : i = 1, 2, \ldots, m_1\}$ es un nuevo conjunto de funciones base. Habitualmente el número de funciones base es menor que el número de muestras de entrenamiento, es decir, $m_1 < N$. Con las funciones de base radial en mente podemos hacer

$$\varphi_i(\boldsymbol{x}) = G(\|\boldsymbol{x} - \boldsymbol{t}_i\|) \quad i = 1, 2, \ldots, m_1 \tag{5.23}$$

donde debe determinarse el conjunto de centros $\{\boldsymbol{t}_i : i = 1, 2, \ldots, m_1\}$. Con esto podemos redefinir $F^*(\boldsymbol{x})$ como

$$F^*(\boldsymbol{x}) = \sum_{i=1}^{m_1} w_i G(\|\boldsymbol{x} - \boldsymbol{t}_i\|) \tag{5.24}$$

Si definimos

$$\boldsymbol{G} = \begin{bmatrix} G(\boldsymbol{x}_1, \boldsymbol{t}_1) & G(\boldsymbol{x}_1, \boldsymbol{t}_2) & \cdots & G(\boldsymbol{x}_1, \boldsymbol{t}_{m_1}) \\ G(\boldsymbol{x}_2, \boldsymbol{t}_1) & G(\boldsymbol{x}_2, \boldsymbol{t}_2) & \cdots & G(\boldsymbol{x}_2, \boldsymbol{t}_{m_1}) \\ \cdots & \cdots & \cdots & \cdots \\ G(\boldsymbol{x}_N, \boldsymbol{t}_1) & G(\boldsymbol{x}_N, \boldsymbol{t}_2) & \cdots & G(\boldsymbol{x}_N, \boldsymbol{t}_{m_1}) \end{bmatrix} \tag{5.25}$$

$$\boldsymbol{d} = [d_1, d_2, \ldots, d_N]^T \tag{5.26}$$

$$\boldsymbol{w} = [w_1, w_2, \ldots, w_{m_1}]^T \tag{5.27}$$

y

$$\boldsymbol{G}_0 = \begin{bmatrix} G(\boldsymbol{t}_1, \boldsymbol{t}_1) & G(\boldsymbol{t}_1, \boldsymbol{t}_2) & \cdots & G(\boldsymbol{t}_1, \boldsymbol{t}_{m_1}) \\ G(\boldsymbol{t}_2, \boldsymbol{t}_1) & G(\boldsymbol{t}_2, \boldsymbol{t}_2) & \cdots & G(\boldsymbol{t}_2, \boldsymbol{t}_{m_1}) \\ \cdots & \cdots & \cdots & \cdots \\ G(\boldsymbol{t}_{m_1}, \boldsymbol{t}_1) & G(\boldsymbol{t}_{m_1}, \boldsymbol{t}_2) & \cdots & G(\boldsymbol{t}_{m_1}, \boldsymbol{t}_{m_1}) \end{bmatrix} \tag{5.28}$$

puede deducirse el equivalente de la ecuación (5.21) como

$$(\boldsymbol{G}^T \boldsymbol{G} + \lambda \boldsymbol{G}_0)\boldsymbol{w} = \boldsymbol{G}^T \boldsymbol{d} \tag{5.29}$$

Conforme el parámetro de regularización λ tiende a cero, el vector de pesos $\boldsymbol{w}$ converge (si $m_1 < N$) a una solución basada en el cálculo de la pseudoinversa de $\boldsymbol{G}$:

$$\boldsymbol{w} = \boldsymbol{G}^+ \boldsymbol{d}, \qquad \lambda = 0 \tag{5.30}$$

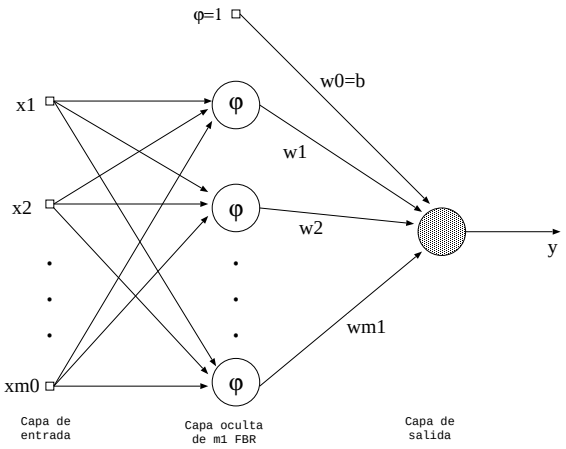


Figura 5.3: Red neuronal generalizada de funciones de base radial.

Los resultados anteriores se plasman en las redes de funciones de base radial generalizadas, cuya estructura es la que se muestra en la figura 5.3.

Aunque similares, hay ciertas diferencias entre las estructuras de las figuras 5.2 y 5.3:

- El número de nodos ocultos es igual al número de muestras de entrenamiento N en la redes de regularización; en el caso de las RFBR generalizadas este número es menor, $m_1 < N$.
- En la figura 5.3 los parámetros desconocidos a aprender son los pesos lineales asociados con la capa de salida y las posiciones de los centros. En las redes de regularización los únicos parámetros desconocidos son los pesos lineales de la capa de salida.

El enfoque más sencillo para la selección de los centros es asumir que las funciones de activación de las unidades ocultas son funciones de base radial fijas. Los centros pueden tomarse aleatoriamente a partir del conjunto de entrenamiento. Este enfoque suele ser acertado siempre que los datos de entrenamiento se distribuyan de una forma representativa para el problema en cuestión. La desviación estándar (el *ancho*) de todas las FBR gaussianas se fija a

$$\sigma = \frac{d_{máx}}{\sqrt{2m_1}} \tag{5.31}$$

donde $d_{máx}$ es la máxima distancia entre los centros elegidos y m_1 es el número de centros. Esta fórmula asegura que las FBR individuales no son ni demasiado afiladas ni demasiado planas, condiciones ambas que deben

evitarse. Los únicos parámetros que deben aprenderse con este enfoque son los pesos lineales de la capa de salida de la red.

Métodos más elaborados para la selección de los centros son la *selección autoorganizativa* y la *selección supervisada* de los centros.

5.5.1. Estimación del parámetro de regularización

El parámetro de regularización λ juega un papel central en la teoría de la regularización de las RFBR mostrada aquí. El método denominado *validación cruzada generalizada* es uno de los más utilizados para determinar un valor adecuado del parámetro de regularización.

5.5.2. La maldición de la dimensionalidad

El espacio de funciones de aproximación aprendible con perceptrones multicapa o con redes de funciones de base radial se va restringiendo cada vez más conforme aumenta la dimensión m_0. Un resultado importante es que la maldición de la dimensionalidad no puede combatirse con redes neuronales ya sean perceptrones multicapa o RFBR, ni con cualquier otra técnica de naturaleza similar.

5.5.3. Comparación entre RFBR y PMC

Los siguientes puntos comparan brevemente las redes de funciones de base radial (RFBR) y los perceptrones multicapa (PMC):

- Ambos tipos de redes son redes sin ciclos no lineales. Ambas son aproximadores universales.
- Las RFBR tienen una única capa oculta, mientras que los PMC tienen una o más.
- La capa de salida es lineal en las RFBR y no lineal en el caso de los PMC.
- En las RFBR el argumento de la función de activación de cada unidad oculta computa la norma euclídea entre el vector de entrada y el centro correspondiente. En los PMC se calcula el producto escalar del vector de entrada y el vector de pesos sinápticos de la unidad correspondiente.

λ	0	0,1	1	100
	80,40	81,06	81,03	79,28
	81,20	81,27	80,92	73,77
	81,29	81,42	81,05	69,83
	79,64	81,45	81,08	78,00
	81,37	81,51	80,48	56,47
Media	80,78	81,34	80,91	71,47

Cuadro 5.1: Porcentajes de aciertos de redes de funciones de base radial con 30 centros para distintos valores de λ. Para cada valor de λ se muestran los resultados de 5 experimentos independientes y su media.

(a) $m_1 = 30$ (b) $m_1 = 100$

Figura 5.4: Fronteras de decisión de dos redes de funciones de base radial con $\lambda = 100$ junto a la frontera de decisión óptima (círculo interior). La tasa de aciertos en la evaluación de la red fue de 78,36 % para el caso (a) y de 81,16 % para el caso (b).

Las características lineales de la capa de salida de una RFBR la situan más cerca del perceptrón de Rosenblatt que del PMC. Sin embargo, las RFBR difieren del perceptrón en que son capaces de implementar transformaciones no lineales arbitrarias del espacio de entrada (como ejemplo más simple, son capaces de resolver el problema de la función O exclusiva).

5.6. Experimentos

Para evaluar el rendimiento de las redes de funciones de base radial generalizadas, volvemos a recurrir al problema del capítulo 2. Si codificamos las salidas como se sugirió en el capítulo 3, podemos considerarlas como

λ	0	0,1	1	100
	78,95	81,61	81,32	77,85
	80,21	81,45	80,58	78,74
	80,21	81,36	81,07	71,52
	80,72	81,52	81,22	67,73
	79,57	81,45	81,28	75,58
Media	79,93	81,48	81,09	74,28

Cuadro 5.2: Porcentajes de aciertos de redes de funciones de base radial con 100 centros para distintos valores de λ. Para cada valor de λ se muestran los resultados de 5 experimentos independientes y su media.

probabilidades, de manera que la salida i-ésima indica la probabilidad de que el símbolo de entrada pertenezca a la clase $\mathcal{X}_i$.

Como funciones de base radial se utilizaron gaussianas con $\sigma = 4$. Los conjuntos de entrenamiento estaban formados por 500 muestras y el de evaluación por 32 000. Se realizaron varios experimentos con distintos valores de m_1 y λ. Los resultados de 5 experimentos independientes en cada caso y el valor medio de la tasa de aciertos se muestran en los cuadros 5.1 y 5.2.

Los resultados son mejores que los obtenidos con el perceptrón multicapa. Lo anterior se confirma observando las fronteras de decisión de las figuras 5.4(a) y 5.4(b); en el segundo caso, la frontera de decisión óptima y la obtenida con la red de funciones de base radial casi coinciden.

Capítulo 6

Minimización del error

Para una determinada tarea de aprendizaje, con una cantidad finita de datos de entrenamiento, el mejor rendimiento al generalizar se conseguirá si se equilibra la balanza entre la exactitud obtenida sobre ese conjunto de entrenamiento y la *capacidad* de la máquina, esto es, la habilidad de la máquina de aprender cualquier conjunto de entrenamiento sin error.

Un ejemplo de desequilibrio de la balanza hacía las muestras de entrenamiento sería la de un clasificador que dijera «un árbol no es un árbol porque tiene un número distinto de hojas». Si la balanza se inclinara hacia el otro lado, con una capacidad demasiado baja, el resultado sería «si es verde, es un árbol». Ninguna de las dos máquinas puede generalizar bien. La exploración y formalización de estos conceptos ha abierto una de las líneas más prometedoras dentro de la teoría del aprendizaje estadístico.

6.1. Riesgo esperado y riesgo empírico

Supongamos que tenemos N muestras $\{(\boldsymbol{x}_i, z_i) : \boldsymbol{x}_i \in \mathbb{R}^m, i = 1, \ldots, N\}$. Para simplificar las fórmulas siguientes, la salida esperada z_i será 1 o -1.

Ahora supongamos que existe una distribución de probabilidad desconocida $P(\boldsymbol{x}, z)$ de la que se han tomado esas muestras. Esto es más general que asignar un z fijo a cada $\boldsymbol{x}$. Con esta distribución, las etiquetas z_i se asignan según una distribución de probabilidad condicional en $\boldsymbol{x}_i$. Más adelante, sin embargo, asumiremos un z fijo para un $\boldsymbol{x}$ dado.

Supongamos que tenemos una máquina para aprender la transformación $\boldsymbol{x}_i \longmapsto z_i$. La máquina está definida por $\boldsymbol{x} \longmapsto f(\boldsymbol{x}, \zeta)$ donde ζ es un conjunto de parámetros ajustables (por ejemplo, en una red neuronal los pesos y los

umbrales). Una elección particular de ζ genera lo que llamamos una *máquina entrenada*.

La esperanza del error de evaluación para una máquina entrenada, llamada también *riesgo real*, *riesgo esperado* o simplemente *riesgo*, es

$$R(\zeta) = \int \frac{1}{2} |z - f(\boldsymbol{x}, \zeta)| \, dP(\boldsymbol{x}, z) \tag{6.1}$$

que, si existe una función de densidad de probabilidad, puede escribirse como

$$R(\zeta) = \int \frac{1}{2} |z - f(\boldsymbol{x}, \zeta)| \, p(\boldsymbol{x}, d) \, d\boldsymbol{x}, dz \tag{6.2}$$

Parece una forma secilla de calcular el error medio real, pero normalmente no tenemos ni siquiera una estimación de la forma de $P(\boldsymbol{x}, z)$.

El *riesgo empírico* es el error medio medido sobre el conjunto de evaluación:

$$R_{emp}(\zeta) = \frac{1}{2N} \sum_{i=1}^{N} |z_i - f(\boldsymbol{x}_i, \zeta)| \tag{6.3}$$

Nótese cómo aquí no aparece ninguna distribución de probabilidad. A la cantidad

$$\frac{1}{2}|z_i - f(\boldsymbol{x}_i, \zeta)| \tag{6.4}$$

se le denomina *pérdida*. Ahora elijamos un ρ tal que $0 \leq \rho \leq 1$. Entonces, la siguiente cota se cumple con probabilidad $1 - \rho$:

$$R(\zeta) \leq R_{emp}(\zeta) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\rho/4)}{N}} \tag{6.5}$$

donde h es un entero no negativo llamado *dimensión de Vapnik-Chervonenkis* (VC) y es una medida de la idea de capacidad mencionada al principio del capítulo. A la parte derecha de la desigualdad se le llama *cota del riesgo* y al segundo término de la cota del riesgo se le llama *confianza de VC*.

Algunas notas sobre esta cota:

- es independiente de $P(\boldsymbol{x}, z)$,
- normalmente no es posible calcular la parte izquierda de la desigualdad,
- si conociéramos h, podríamos calcular la parte derecha. De esta forma, dadas diferentes máquinas de aprendizaje (familias de funciones $f(\boldsymbol{x}, \zeta)$) y eligiendo un valor de ρ lo suficientemente pequeño, si seleccionamos la máquina que minimice la parte derecha de la desigualdad, estaremos eligiendo la máquina que da la cota superior más pequeña del riesgo real.

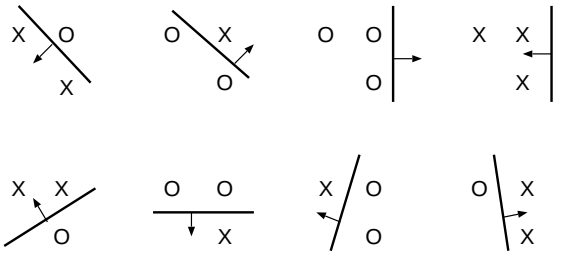


Figura 6.1: Conjunto de tres puntos de $\mathbb{R}^2$ que son rotos por las funciones del conjunto de hiperplanos orientados.

El último punto da la idea básica para la minimización del riesgo estructural discutida más adelante.

6.2. La dimensión de Vapnik-Chervonenkis

La dimensión VC es una propiedad de un conjunto de funciones $\{f(\zeta)\}$ y puede definirse para distintas clases de funciones f. Aquí nos centraremos en funciones de la forma $f(\boldsymbol{x}, \zeta) \in \{-1, 1\}\ \forall\, \boldsymbol{x}, \zeta$.

Si un conjunto de N puntos puede etiquetarse de 2^N formas distintas, y para cada etiquetado puede encontrarse un elemento del conjunto $\{f(\zeta)\}$ que asigne correctamente estas etiquetas, diremos que el conjunto de puntos es *roto* por ese conjunto de funciones. La dimensión VC del conjunto de funciones $\{f(\zeta)\}$ se define como el máximo número de puntos de entrenamiento que pueden ser rotos por $\{f(\zeta)\}$. Nótese que si la dimensión VC es h, entonces existe como mínimo un conjunto de h puntos que puede romperse, pero en general no todos los conjuntos de h puntos podrán romperse.

6.2.1. Dimensión VC del conjunto de hiperplanos orientados

Supongamos que el espacio en que viven los datos es $\mathbb{R}^2$ y que $\{f(\zeta)\}$ está formado por líneas orientadas (las fronteras de decisión que aparecían el capítulo 3). Aunque es posible encontrar tres puntos que pueden romperse con este conjunto de funciones (figura 6.1), es imposible encontrar cuatro. Por lo tanto, la dimensión VC del conjunto de líneas orientadas en $\mathbb{R}^2$ es 3.

Nótese de paso que, aunque los puntos de la figura 6.1 pueden romperse, no todos los conjuntos de tres puntos en $\mathbb{R}^2$ pueden ser rotos. Por ejemplo, si los tres puntos son colineales, es imposible romperlos con líneas orientadas.

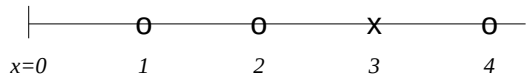


Figura 6.2: Pese a tener dimensión VC infinita, la máquina de un solo parámetro de la ecuación (6.6) no es capaz de romper este conjunto de cuatro puntos.

El resultado anterior se generaliza en el siguiente teorema y corolario.

Teorema 6.1 *Consideremos un conjunto de N puntos de $\mathbb{R}^n$ y tomemos cualquiera de ellos como origen. Los N puntos pueden romperse con hiperplanos orientados si y solo si los vectores de posición de los restantes puntos son linealmente independientes.*

Corolario 6.1 *La dimensión VC del conjunto de hiperplanos orientados en $\mathbb{R}^n$ es $n+1$, ya que siempre pueden encotrarse n puntos linealmente independientes (dado un origen cualquiera), pero nunca $n+1$.*

La dimensión VC formaliza la idea de la capacidad de un conjunto de funciones. Intuitivamente cabe esperar que máquinas con muchos parámetros tendrán una dimensión VC alta y que máquinas con pocos parámetros la tendrán baja. Aunque, esto suele ser así, hay un contraejemplo: una máquina con un único parámetro y con dimensión VC infinita (puede romper cualquier conjunto de N puntos, independientemente de N):

$$f(x,\zeta) = \theta(\text{sen}(\zeta x)), \quad x,\zeta \in \mathbb{R} \tag{6.6}$$

donde θ es la función escalón definida como

$$\theta(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases} \tag{6.7}$$

Como curiosidad, aunque con esta función podemos romper un número arbitrario de puntos, podemos encontrar cuatro que no pueden ser rotos. Estos cuatro puntos son los que se muestran en la figura 6.2.

6.2.2. Acotación del riesgo

La figura 6.3 muestra la evolución con un nivel de confianza del 95 % ($\rho = 0{,}05$) de la confianza VC al aumentar la dimensión VC. Como puede observarse, la confianza VC es monótona creciente con h. Aunque la gráfica

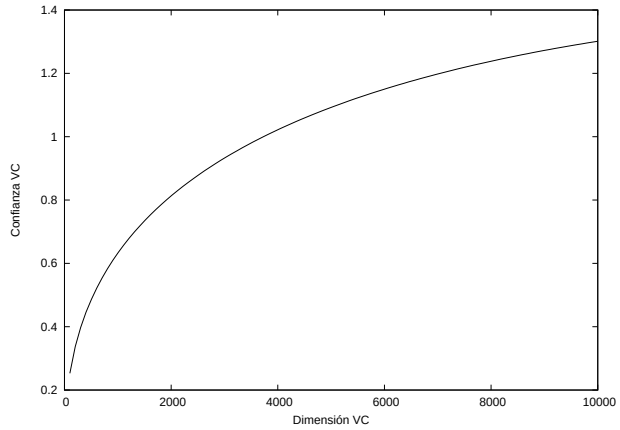


Figura 6.3: Evolución de la confianza VC en función de la dimensión VC con $\rho = 0{,}05$ y $N = 10\,000$.

se generó para una valor de $N = 10\,000$, lo anterior es cierto para cualquier valor de N.

Por lo tanto, si tenemos un conjunto de máquinas de aprendizaje cuyo riesgo empírico es cero, nos quedaremos con aquella de menor dimensión VC, es decir, con aquella que minimiza la cota del riesgo.

De cualquier forma, es importante tener en cuenta que la ecuación (6.5) da (con una determinada probabilidad) una cota superior del riesgo real, y que esto no impide que una determinada máquina con el mismo R_{emp} y cuyo conjunto de funciones asociado presente una dimensión VC mayor tenga mejor rendimiento. Un ejemplo de sistema de este tipo que da un buen rendimiento pese a tener dimensión VC infinita es el de un clasificador basado en los k vecinos más cercanos con $k = 1$. Este conjunto de funciones tiene dimensión VC infinita y riesgo empírico cero, ya que cualquier conjunto de puntos etiquetados arbitrariamente será aprendido correctamente por el algoritmo. En este caso la cota no da ninguna información. Con todo los clasificadores basados en el vecino más cercano funcionan bien. Así, una capacidad infinita no implica un rendimiento pobre.

6.3. Minimización estructural del riesgo

La confianza VC de la ecuación (6.5) depende de la clase de funciones escogidas, mientras que el riesgo empírico y el riesgo real dependen de la función particular elegida por el algoritmo de entrenamiento.

El objetivo es encontrar un subconjunto del conjunto de funciones que minimice la cota del riesgo. Para ello dividimos la clase completa de funciones en subconjuntos anidados. Para cada conjunto deberíamos poder calcular h o, al menos, establecer una cota de su valor. La *minimización estructural del riesgo* consiste en encontrar el subconjunto de funciones que minimiza la cota del error actual. Entonces se toma aquella máquina entrenada de la serie con menor valor para la suma del riesgo empírico y la confianza VC.

Capítulo 7

Máquinas de vectores soporte

Muchas redes neuronales se diseñan para encontrar un hiperplano que separe las muestras de las distintas clases. Normalmente se comienza con un hiperplano aleatorio que se va desplazando hasta que todos los datos del entrenamiento quedan en la parte correcta. Esto deja, inevitablemente, algunos puntos del entrenamiento muy cerca del hiperplano lo cual no es un resultado óptimo a la hora de generalizar.

El inicio de la teoría en la que se basan las máquinas de vectores soporte data de los últimos setenta con los trabajos de Vapnik. Esta teoría fue discutida en el capítulo anterior.

Al igual que los perceptrones multicapa y que las redes de funciones de base radial, las máquinas de vectores soporte pueden usarse tanto para clasificación de patrones como para regresión no lineal. Aquí nos centraremos en el uso de las máquinas de vectores soporte para el reconocimiento de formas.

La formulación de las máquinas de vectores soporte se basa en el principio de minimización estructural del riesgo que ha demostrado ser superior al principio de minimización del riesgo empírico, que es el empleado por muchas de las redes neuronales convencionales. Las máquinas de vectores soporte presentan un buen rendimiento al generalizar en problemas de clasificación, pese a no incorporar conocimiento específico sobre el dominio.

7.1. Máquinas de vectores soporte lineales

Comenzaremos con máquinas lineales entrenadas con datos linealmente separables (el caso general, máquinas no lineales entrenadas con datos no separables, se resuelve con un problema de programación cuadrática muy similar).

Consideremos el conjunto de entrenamiento

$$\{\boldsymbol{x}_i, z_i\} \quad i = 1, \ldots, N, \quad z_i \in \{-1, 1\}, \quad \boldsymbol{x}_i \in \mathbb{R}^m \tag{7.1}$$

Supongamos que existe un hiperplano que separa los puntos de ambas clases. Como vimos en el capítulo 3, los puntos $\boldsymbol{x}$ sobre el hiperplano satisfacen $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ donde el vector $\boldsymbol{w}$ es normal al hiperplano, $|b|/\|\boldsymbol{w}\|$ es la distancia perpendicular del hiperplano al origen y $\|\boldsymbol{w}\|$ es la norma euclídea de $\boldsymbol{w}$.

Sea d_+ (d_-) la distancia más corta del hiperplano de separación a la muestra positiva (negativa, respectivamente) más cercana. Definamos el *margen* del hiperplano como la suma $d_+ + d_-$. En el caso linealmente separable, el algoritmo buscará simplemente el hiperplano con mayor margen. A continuación formulamos esta idea.

Supongamos que todos los datos de entrenamiento satisfacen

$$\boldsymbol{w}^T\boldsymbol{x}_i + b \geq +1 \qquad \text{para } z_i = +1 \tag{7.2a}$$

$$\boldsymbol{w}^T\boldsymbol{x}_i + b \leq -1 \qquad \text{para } z_i = -1 \tag{7.2b}$$

esto es

$$z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geq 0 \quad \forall\, i \tag{7.3}$$

Ahora consideremos los puntos para los que se cumple la igualdad en (7.2a) (que este punto exista es equivalente a elegir una escala adecuada para $\boldsymbol{w}$ y b). Estos puntos están sobre el hiperplano $H_1 : \boldsymbol{w}^T\boldsymbol{x}_i + b = 1$ con normal $\boldsymbol{w}$ y distancia al origen $|1-b|/\|\boldsymbol{w}\|$. De forma similar, para el hiperplano H_2 la distancia al origen es $|-1-b|/\|\boldsymbol{w}\|$. Por lo tanto, $d_+ = d_- = 1/\|\boldsymbol{w}\|$ y el margen es simplemente $2/\|\boldsymbol{w}\|$. Nótese que H_1 y H_2 son paralelos (tienen la misma normal) y que no hay puntos de entrenamiento entre ellos. Podemos, en definitiva, encontrar el par de hiperplanos que dan el máximo margen minimizando la función de coste $1/2\,\|\boldsymbol{w}\|^2$ con las restricciones (7.3).

Ahora pasaremos a una formulación lagrangiana del problema. Hay dos razones importantes para hacer esto:

- la primera es que las restricciones de la ecuación (7.3) se sustituirán por restricciones sobre los multiplicadores de Lagrange, que serán más fáciles de manejar;

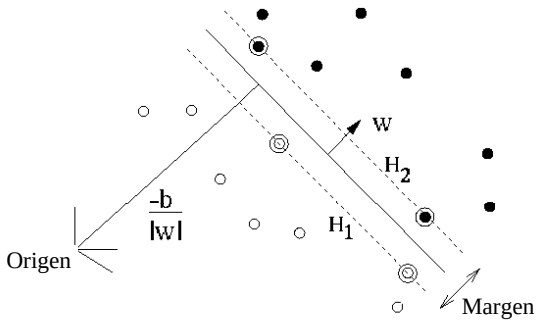


Figura 7.1: Ilustración de la idea de hiperplano de separación óptimo para el caso de patrones linealmente separables. Los vectores soporte (aquellos que yacen sobre H_1, H_2 y cuya eliminación cambiaría la solución encontrada) se muestran rodeados por un círculo. *Figura tomada de [1].*

- la segunda es que con esta reformulación del problema, los datos del entrenamiento solo aparecen en forma de productos escalares entre vectores. Esta propiedad es crucial para generalizar el procedimiento al caso no lineal como veremos.

Por lo tanto, introduzcamos N multiplicadores de Lagrange que denotaremos por $\alpha_1, \alpha_2, \ldots, \alpha_N$, uno para cada una de las restricciones de (7.3). La regla es que para restricciones de la forma $c_i \geq 0$ las ecuaciones que las definen se multiplican por multiplicadores de Lagrange positivos y se restan de la función objetivo para formar el lagrangiano. En el caso de restricciones de la forma $c_i = 0$, los multiplicadores de Lagrange no tienen restricciones. Lo anterior da el lagrangiano

$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i z_i (\boldsymbol{w}^T \boldsymbol{x}_i + b) + \sum_{i=1}^{N} \alpha_i \tag{7.4}$$

Ahora debemos minimizar L_P con respecto a $\boldsymbol{w}$, b y a la vez exigir que las derivadas de L_P con respecto a todos los α_i se anulen, todo sujeto a las restricciones $\alpha_i \geq 0$ (restricciones $\mathcal{C}_1$). Puede demostrarse que se trata de un problema de programación cuadrática convexo. Esto quiere decir que podemos resolver de forma equivalente el siguiente problema *dual*: *maximizar* L_P sujeto a la restricción de que el gradiente de L_P con respecto a $\boldsymbol{w}$ y b se anule y sujeto también a la restricción de que $\alpha_i \geq 0$ (restricciones $\mathcal{C}_2$). Esta formulación particular del problema se conoce como *dual de Wolfe* y presenta la propiedad de que el máximo de L_P con las restricciones $\mathcal{C}_2$ ocurre en los mismos valores de $\boldsymbol{w}$, b y α que el mínimo de L_P con las restricciones $\mathcal{C}_1$.

Al requerir que se anule el gradiente de L_P con respecto a $\boldsymbol{w}$ y b, obtenemos las condiciones:

$$\boldsymbol{w} = \sum_i \alpha_i z_i \boldsymbol{x}_i \tag{7.5}$$

$$\sum_i \alpha_i z_i = 0 \tag{7.6}$$

Ya que en estas restricciones aparecen igualdades, podemos sustituirlas en la ecuación (7.4) para obtener

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{7.7}$$

La solución se obtiene minimizando L_P o maximizando L_D y viene dada por la ecuación (7.5).

Nótese que hay un multiplicador de Lagrange α_i para cada punto de entrenamiento. Tras obtener una solución, aquellos puntos para los que $\alpha_i > 0$ se denominan *vectores soporte* y yacen sobre los hiperplanos H_1, H_2. El resto de puntos tienen $\alpha_i = 0$ y satisfacen la ecuación (7.3), quizá con la igualdad. Con estas máquinas, los vectores soporte son los elementos críticos del conjunto entrenamiento: son los más cercanos a la frontera de decisión y si el resto de puntos no se consideraran en un nuevo entrenamiento, el algoritmo encontraría el mismo hiperplano de separación.

Nótese de paso como $\boldsymbol{w}$ está determinado explícitamente por el algoritmo de entrenamiento, pero no es este el caso del umbral b, aunque su obtención es inmediata: basta tomar en la ecuación (7.3) cualquier i para el que $\alpha_i \neq 0$ y despejar b; por ejemplo, si $z_i = 1$ (vector positivo):

$$b = 1 - \boldsymbol{w}^T \boldsymbol{x}_i \tag{7.8}$$

En cualquier caso, siempre es más seguro tomar el valor medio de b que resulte de todas las ecuaciones.

7.1.1. Fase de evaluación

Una vez que hayamos entrenado una máquina de vectores soporte (MVS), para clasificar un patrón de evaluación $\boldsymbol{x}$ basta determinar en qué parte de la frontera de decisión se encuentra y asignarle la etiqueta de la clase correspondiente, es decir, asignamos a $\boldsymbol{x}$ la clase $\text{sgn}(\boldsymbol{w}^T \boldsymbol{x} + b)$, donde sgn es la función signo.

7.2. El caso no separable

En el caso no linealmente separable nos interesa poder relajar las restricciones de (7.2), pero únicamente cuando sea necesario, es decir, nos interesa añadir un nuevo coste a la función objetivo. Una posible forma de hacerlo es introduciendo variables débiles ξ_i, $i = 1, \ldots, N$, en las restricciones para convertirlas en

$$\begin{aligned} \boldsymbol{w}^T\boldsymbol{x}_i + b &\geq +1 - \xi_i \qquad \text{para } z_i = +1 && \text{(7.9a)} \\ \boldsymbol{w}^T\boldsymbol{x}_i + b &\leq -1 + \xi_i \qquad \text{para } z_i = -1 && \text{(7.9b)} \\ \xi_i &\geq 0 \qquad\qquad \forall\, i && \text{(7.9c)} \end{aligned}$$

Con esto, para que una muestra quede mal clasificada, el correspondiente ξ_i debe ser mayor que la unidad. La suma $\sum_i \xi_i$ es, por tanto, una cota superior en el número de errores sobre el entrenamiento. Una forma de añadir el coste de los errores a la función objetivo es intentar minimizar $1/2\,\|\boldsymbol{w}\|^2 + \gamma(\sum_i \xi_i)$, donde γ es un parámetro ajustable; un valor grande de γ equivale a una mayor penalización de los errores.

El problema anterior es de nuevo un problema de programación cuadrática con la agradable propiedad de que ni los ξ_i ni sus multiplicadores de Lagrange asociados, aparecen en el problema dual de Wolfe, que es ahora

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j z_i z_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{7.10}$$

sujeto a las restricciones

$$0 \leq \alpha_i \leq \gamma \tag{7.11}$$

$$\sum_i \alpha_i z_i = 0 \tag{7.12}$$

La solución viene nuevamente dada por

$$\boldsymbol{w} = \sum_{i=1}^{N^S} \alpha_i z_i \boldsymbol{s}_i \tag{7.13}$$

donde ahora N^S es el número de vectores soporte y $\boldsymbol{s}_i$ es el i-ésimo vector soporte. Por lo tanto, la única diferencia con el caso del hiperplano óptimo es que los valores de α_i están ahora acotados superiormente por γ. La figura 7.2 ilustra gráficamente algunas de las ideas anteriores.

Aunque no se demostrará, el cálculo del umbral b sigue de nuevo la expresión $b = 1 - \boldsymbol{w}^T\boldsymbol{x}_i$ con $0 < \alpha_i < \gamma$. Al igual que en el caso linealmente separable, es más adecuado tomar la media sobre todos los vectores soporte.

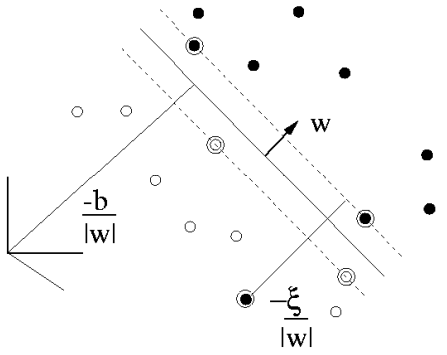


Figura 7.2: Las variables débiles ξ_i permiten relajar la condición de separabilidad para el caso no linealmente separable. *Figura tomada de [1].*

En cuanto al parámetro γ, conforme $\gamma \to 0$, la solución converge a la obtenida con el hiperplano óptimo (sobre el problema no separable). Por otro lado, conforme $\gamma \to \infty$, la solución converge hacia una en la que domina el término de maximización del margen y se hace menos énfasis en minimizar el error de clasificación.

7.3. Máquinas no lineales de vectores soporte

Mediante un viejo truco, generalizaremos en esta sección los métodos anteriores al caso en que la función de decisión no es lineal con los datos.

En primer lugar, observemos que la única forma en que aparecen los datos en las ecuaciones (7.10)-(7.12) es como productos escalares $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$. Supongamos entonces que llevamos los datos a un nuevo espacio euclídeo $\mathcal{H}$ (posiblemente de dimensión infinita), mediante una transformación $\aleph$ (véase la figura 7.3)

$$\aleph : \mathbb{R}^m \longmapsto \mathcal{H} \tag{7.14}$$

Esta operación se realiza de acuerdo con el teorema de Cover sobre la separabilidad de patrones, que ya se discutió en 5.1. Con esto, el algoritmo de entrenamiento solo dependerá de los datos a través de productos escalares en $\mathcal{H}$, es decir, de funciones de la forma $\aleph(\boldsymbol{x}_i) \cdot \aleph(\boldsymbol{x}_j)$. Si existiera una *función núcleo* K tal que $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \aleph(\boldsymbol{x}_i) \cdot \aleph(\boldsymbol{x}_j)$, podríamos utilizar únicamente K en el algoritmo de entrenamiento sin tener que conocer explícitamente $\aleph$.

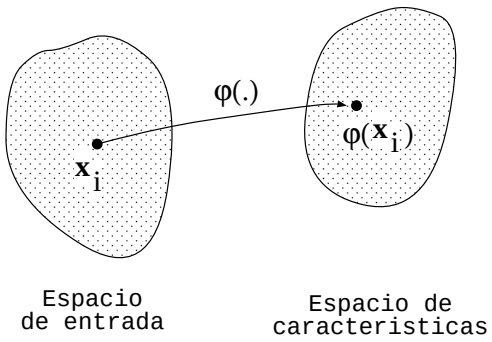


Figura 7.3: Transformación no lineal $\varphi(\cdot)$ del espacio de entrada al espacio de características $\mathcal{H}$.

Un ejemplo de función núcleo es

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{7.15}$$

En este ejemplo $\mathcal{H}$ es de dimensión infinita y no sería sencillo trabajar explícitamente con $\aleph$. Sin embargo, si sustituimos $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ por $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ a lo largo de todo el algoritmo de entrenamiento, obtendremos una MVS que vive en un espacio de dimensión infinita y que opera en prácticamente la misma cantidad de tiempo. La separación sigue siendo lineal, pero en un espacio diferente.

En la fase de evaluación la MVS se usa calculando el producto escalar de un punto de evaluación dado $\boldsymbol{x}$ con $\boldsymbol{w}$ y calculando el signo de

$$f(\boldsymbol{x}) = \sum_{i=1}^{N^S} \alpha_i z_i \aleph(\boldsymbol{s}_i) \cdot \aleph(\boldsymbol{x}) + b \tag{7.16}$$

$$= \sum_{i=1}^{N^S} \alpha_i z_i K(\boldsymbol{s}_i, \boldsymbol{x}) + b \tag{7.17}$$

donde $\boldsymbol{s}_i$ son los vectores soporte. De nuevo, por tanto, podemos evitar el cálculo directo de $\aleph(\boldsymbol{x})$.

7.3.1. Ejemplo de núcleo

Veamos ahora un ejemplo muy sencillo de núcleo para el que puede construirse una transformación $\aleph$.

Supongamos que los datos son vectores de $\mathbb{R}^2$ y que tomamos $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^2$. Es fácil encontrar un espacio $\mathcal{H}$ y una transformación $\aleph$ de $\mathbb{R}^2$ en

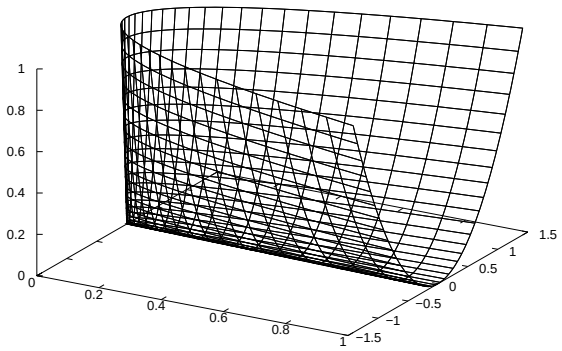


Figura 7.4: La transformación $\aleph(\boldsymbol{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$.

$\mathcal{H}$ tal que $(\boldsymbol{x} \cdot \boldsymbol{y})^2 = \aleph(\boldsymbol{x}) \cdot \aleph(\boldsymbol{y})$: tomemos $\mathcal{H} = \mathbb{R}^3$ y

$$\aleph(\boldsymbol{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \tag{7.18}$$

Para los datos del espacio de entrada en el rango $[-1, 1] \times [-1, 1] \in \mathbb{R}^2$, la imagen de $\aleph$ se muestra en la figura 7.4. La imagen de $\aleph$ puede vivir en un espacio de dimensión muy alta, pero se trata esencialmente de una superficie cuya dimensión intrínseca es la misma que la del espacio de entrada. Es importante saber que ni la transformación $\aleph$ ni el espacio $\mathcal{H}$ son únicos para un determinado núcleo. Podríamos haber elegido de nuevo $\mathcal{H} = \mathbb{R}^3$, pero

$$\aleph(\boldsymbol{x}) = \frac{1}{\sqrt{2}} \begin{bmatrix} (x_1^2 - x_2^2) \\ 2x_1x_2 \\ (x_1^2 + x_2^2) \end{bmatrix} \tag{7.19}$$

o $\mathcal{H} = \mathbb{R}^4$ y como transformación

$$\aleph(\boldsymbol{x}) = \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_1x_2 \\ x_2^2 \end{bmatrix} \tag{7.20}$$

En la bibliografía sobre máquinas de vectores soporte se denomina normalmente a $\mathcal{H}$ como *espacio de Hilbert*.

7.3.2. Condición de Mercer

La condición de Mercer permite determinar para qué núcleos existe un par $\{\mathcal{H}, \aleph\}$ con las propiedades descritas anteriormente y para cuáles no.

Teorema 7.1 (Condición de Mercer) *Existe una transformación* $\aleph$ *y una expansión en series*

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \aleph(\boldsymbol{x})_i \aleph(\boldsymbol{y})_i \tag{7.21}$$

si y solo si para cualquier $g(\boldsymbol{x})$ *para la que la integral*

$$\int g(\boldsymbol{x})^2 d\boldsymbol{x} \tag{7.22}$$

sea finita, se tiene

$$\int K(\boldsymbol{xy})\, g(\boldsymbol{x}) g(\boldsymbol{y})\, d\boldsymbol{x} d\boldsymbol{y} \geq 0 \tag{7.23}$$

Además, debe tenerse en cuenta que el truco de la transformación implícita de espacios utilizado aquí funcionará en cualquier algoritmo en el que los datos únicamente aparezcan en forma de productos escalares (por ejemplo, el algoritmo de los vecinos más cercanos). Este hecho ha permitido ya derivar una versión no lineal del análisis de las componentes principales y probablemente encontrará nuevas aplicaciones en los próximos años.

7.3.3. Algunas MVS no lineales

Los primeros núcleos utilizados para el reconocimiento de formas son los siguientes:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^p \tag{7.24}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right) \tag{7.25}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\kappa \boldsymbol{x} \cdot \boldsymbol{y} - \delta) \qquad \text{para ciertos } \kappa \text{ y } \delta \tag{7.26}$$

La ecuación (7.24) prporciona un clasificador que es un polinomio de grado p sobre los datos; la ecuación (7.25) da un clasificador basado en funciones de base radial gaussianas y la ecuación (7.26) proporciona un tipo especial de red neuronal de una capa oculta con funciones de activación sigmoideas.

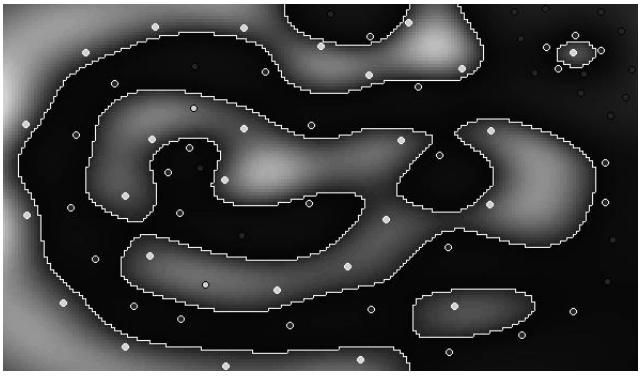


Figura 7.5: Fronteras de decisión obtenidas con una MVS con funciones de base radial gaussianas con $\sigma = 10$ y $\gamma = 10\,000$. Los vectores soporte se muestran encerrados en un círculo.

- En el caso de las FBR (ecuación (7.25)), el número de centros (N^S), los centros en sí (los $\boldsymbol{s}_i$), los pesos (α_i) y el umbral (b) son determinados automáticamente en el entrenamiento de la MVS y dan resultados excelentes en comparación con las FBR gaussianas clásicas.
- En el caso de la red neuronal (ecuación (7.26)), la primera capa está formada por N^S conjuntos de pesos, cada uno formado por m (la dimensión del espacio de entrada) pesos (las componentes de los vectores soporte), y la segunda capa está formada por N^S pesos (los α_i) de manera que el uso de la red pasa simplemente por el cálculo de una suma ponderada de sigmoideas, evaluadas a su vez sobre productos escalares de los datos de evaluación y los vectores soporte. Por lo tanto, en el caso de la red neuronal, la arquitectura (número de pesos) es determinada por el algoritmo de entrenamiento de la MVS.

La figura 7.5 muestra las fronteras de decisión obtenidas con una MVS sobre un problema no separable con un núcleo como el de la ecuación (7.25). Como puede observarse, las fronteras de decisión obtenidas por la MVS pueden ser complejas.

7.4. La dimensión VC de las MVS

Normalmente llevar los datos a un espacio de características con un elevado número de dimensiones provoca un bajo rendimiento en la máquina resultante. A fin de cuentas, los hiperplanos del conjunto $\{\boldsymbol{w}, b\}$ están parametrizados por $\dim(\mathcal{H})+1$ números. La mayor parte de los sistemas de reconocimien-

to de formas ni siquiera pasarían de la línea de salida con un número enorme de parámetros. ¿Por qué las MVS lo hacen tan bien?

La dimensión VC de una MVS puede ser muy grande (incluso infinita). A pesar de ello, las MVS suelen generalizar correctamente. Todavía no existe una respuesta definitiva, pero hay indicios de que el requerir hiperplanos con margen máximo puede estar jugando un papel importante. Por otro lado, también es importante en la lucha contra la maldición de la dimensionalidad el hecho de que, gracias a las funciones núcleo y a la definición del lagrangiano, nos evitamos tener que calcular los parámetros de un hiperplano óptimo en un espacio posiblemente de elevada dimensión. Hoy en día, no existe ninguna teoría que garantice que una determinada familia de MVS se comportarán correctamente con un problema determinado.

Las MVS basan su funcionamiento en el principio de minimización estructural del riesgo, del que son una aproximación. Para entender cómo la minimización de $1/2\|\boldsymbol{w}\|^2$ es equivalente a la implementación del principio de minimización estructural del riesgo, supongamos que se cumple

$$\|\boldsymbol{w}\| \leq A \tag{7.27}$$

donde la variación de la cota A nos servirá para definir una estructura anidada de hiperplanos.

Ahora a partir de la ecuación (7.3), repetida aquí por conveniencia,

$$z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \tag{7.28}$$

y como la distancia de un punto $\boldsymbol{x}$ al hiperplano definido por $\boldsymbol{w}$ y b es

$$d(\boldsymbol{w}, b; \boldsymbol{x}) = \frac{|\boldsymbol{w}^T\boldsymbol{x} + b|}{\|\boldsymbol{w}\|} \tag{7.29}$$

tenemos, entonces, que

$$d(\boldsymbol{w}, b; \boldsymbol{x}) \geq \frac{1}{A} \tag{7.30}$$

Según este resultado, la distancia de cualquiera de los puntos de entrenamiento a los hiperplanos no puede ser menor que $1/A$ e, intuitivamente, esto reduce los posibles hiperplanos y, por tanto, la capacidad. La figura 7.6 ilustra esta idea y el siguiente teorema la concreta más.

Teorema 7.2 *La dimensión VC del conjunto de hiperplanos canónicos en un espacio m dimensional es*

$$h \leq \text{mín}\,\{R^2A^2, m\} \tag{7.31}$$

donde R es el radio de la hiperesfera que envuelve todos los datos.

Por tanto, la minimización de $\|\boldsymbol{w}\|$ es equivalente a la minimización de una cota superior de la dimensión VC.

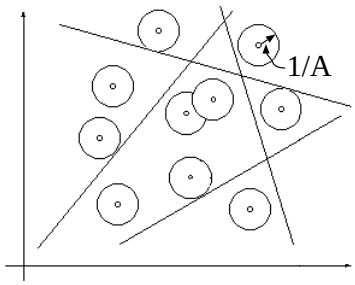


Figura 7.6: La minimización de $(1/2)\|\boldsymbol{w}\|^2$ reduce la dimensión VC de la MVS correspondiente.

7.5. Limitaciones

Cabe señalar que aunque los clasificadores basado en MVS descritos en este capítulo son todos clasificadores binarios, la combinación de varios de ellos permite manejar el caso multiclase. Una combinación efectiva para c clases se obtiene entrenando c clasificadores, dando al clasificador i-ésimo, $i = 1, \ldots, c$, las muestras de la clase i como positivas y las restantes como negativas, y clasificando una muestra de evaluación como perteneciente a aquella clase cuya MVS asociada da la mayor distancia positiva.

Pese a los excelentes resultados obtenidos, las MVS tienen algunas limitaciones:

- La elección de un núcleo adecuado es todavía un área abierta de investigación. Una vez elegido un núcleo, los clasificadores basados en MVS tienen un único parámetro a ajustar por el usuario: la penalización del error γ.
- La complejidad temporal y espacial, tanto en el entrenamiento como en la evaluación, son también una seria limitación. Es un problema sin resolver el entrenamiento con grandes conjuntos de datos (del orden de millones de vectores soporte).
- Aunque hay ya algunos trabajos que estudian el entrenamiento de MVS multiclase en una sola pasada, aún se está lejos de diseñar un clasificador multiclase óptimo basado en MVS.

Con todo, el entrenamiento de una MVS es básicamente un problema de programación cuadrática, que es atractivo por dos motivos: su eficiente computación y la garantía de encontrar un extremo global de la superficie de error.

γ	0,1		1		100		10 000	
	233	81,24	228	81,05	320	80,93	321	81,04
	252	30,92	204	80,73	325	81,43	329	80,58
	227	80,84	190	81,31	307	80,71	332	80,88
	232	80,86	241	80,78	325	81,39	314	80,68
	244	80,71	205	80,96	343	80,90	310	80,12
Media		80,91		80,97		81,07		80,66

Cuadro 7.1: Número de vectores soporte (del total de 500) y porcentajes de aciertos de máquinas de vectores soporte con distintos valores de γ. Para cada valor de γ se muestran los resultados de 5 experimentos independientes y su media.

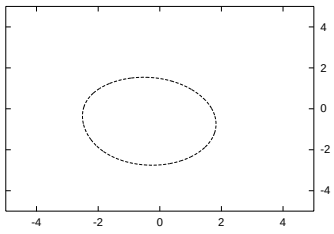


Figura 7.7: Confrontación de la frontera de decisión bayesiana con la obtenida con una máquina de vectores soporte con $\gamma = 0{,}1$ (tasa de aciertos 80,72 %).

7.6. Experimentos

Se evaluó el problema del capítulo 2 sobre una máquina de vectores soporte con núcleos como los de la ecuación (7.25) con $\sigma^2 = 4$. Los conjuntos de entrenamiento tenían 400 muestras y el de evaluación (único) 32 000.

El cuadro 7.1 y la figura 7.7 muestran que el error obtenido con las máquinas de vectores soporte está muy cercano al óptimo.

Apéndice A

Clasificador de Bayes

La teoría de decisión de Bayes es fundamental en la clasificación de patrones. Este enfoque asume que se conocen todos los valores de probabilidad relevantes del problema en cuestión.

Supongamos que tenemos dos clases, que denotaremos por $\mathcal{X}_1$ y $\mathcal{X}_2$, y que existe una *probabilidad a priori* asociada a cada una de ellas, $P(\mathcal{X}_1) = p_1$ y $P(\mathcal{X}_2) = p_2$ con $p_1 + p_2 = 1$. Estas probabilidades a priori reflejan nuestro conocimiento previo acerca de la posibilidad de encontrarnos con una muestra de una u otra clase. Si no tuvieramos más información, la *regla de decisión* sería sencilla: decidir siempre $\mathcal{X}_1$ si $p_1 > p_2$ y decidir siempre $\mathcal{X}_2$ en caso contrario.

Normalmente, sin embargo, podemos tener más información que la suministrada por las probabilidades a priori. Esta información es el resultado de una serie de *mediciones* realizadas en el entorno y representadas mediante un vector de características $\boldsymbol{x}$ sobre un determinado espacio. Sea, entonces, $p(\boldsymbol{x}|\mathcal{X}_i)$, $i = 1, 2$, la función de densidad de probabilidad condicionada a la clase, esto es, la función de densidad de probabilidad de $\boldsymbol{x}$ dado que la clase es $\mathcal{X}_i$. A este factor se le denomina *verosimilitud* de $\mathcal{X}_i$ con respecto a $\boldsymbol{x}$.

Si conocemos todas las probabilidades anteriores y tenemos un vector de datos $\boldsymbol{x}$ resultado de una cierta medición, la regla de decisión cambia. La *regla de Bayes*,

$$P(\mathcal{X}_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{X}_i)P(\mathcal{X}_i)}{\sum_{j=1}^{2} p(\boldsymbol{x}|\mathcal{X}_j)P(\mathcal{X}_j)} \tag{A.1}$$

determina cómo la observación de $\boldsymbol{x}$ cambia la probabilidad a priori $P(\mathcal{X}_i)$ a la probabilidad a posteriori $P(\mathcal{X}_i|\boldsymbol{x})$. La nueva regla de decisión, por tanto, será decidir $\mathcal{X}_1$ si $P(\mathcal{X}_1|\boldsymbol{x}) > P(\mathcal{X}_2|\boldsymbol{x})$ y decidir $\mathcal{X}_2$ en caso contrario.

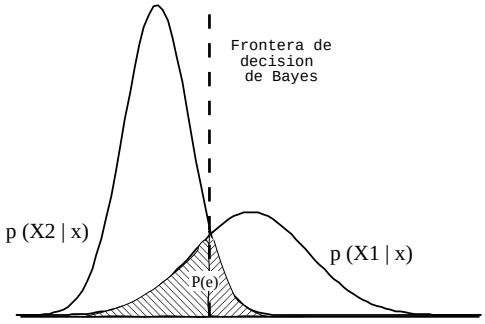


Figura A.1: Componentes de la probabilidad de error en un clasificador bayesiano.

Nótese que el denominador de la ecuación (A.1) es irrelevante desde el punto de vista de la toma de decisión; se trata de un fáctor de escala que puede eliminarse para obtener la siguiente regla de decisión equivalente: *decidir* $\mathcal{X}_1$ *si* $p(\boldsymbol{x}|\mathcal{X}_1)p_1 > p(\boldsymbol{x}|\mathcal{X}_2)p_2$ *y decidir* $\mathcal{X}_2$ *en otro caso.*

Definamos ahora

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{X}_1)}{p(\boldsymbol{x}|\mathcal{X}_2)} \tag{A.2}$$

y

$$\xi = \frac{p_2}{p_1} \tag{A.3}$$

La cantidad $\Lambda(\boldsymbol{x})$, la razón de dos funciones de densidad de probabilidad condicionales, se denomina *razón de verosimilitud*. La cantidad ξ se denomina *umbral* de la prueba. Ambas cantidades son siempre positivas. Con esta nueva notación el clasificador de Bayes es

$$\Lambda(\boldsymbol{x}) \underset{\mathcal{X}_2}{\overset{\mathcal{X}_1}{\gtrless}} \xi \tag{A.4}$$

En la práctica es más conveniente trabajar con el logaritmo de la razón de verosimilitud. Por tanto,

$$\log \Lambda(\boldsymbol{x}) \underset{\mathcal{X}_2}{\overset{\mathcal{X}_1}{\gtrless}} \log \xi \tag{A.5}$$

Es posible hacer esto por dos razones: el logaritmo es una función monótona, y tanto $\Lambda(\boldsymbol{x})$ como ξ son positivos.

La regla de decisión de Bayes divide el espacio en dos regiones $\mathcal{R}_1$ y $\mathcal{R}_2$. Hay dos formas de clasificar erróneamente una observación $\boldsymbol{x}$: la clase

correcta es $\mathcal{X}_1$ y $\boldsymbol{x} \in \mathcal{R}_2$ o bien la clase correcta es $\mathcal{X}_2$ y $\boldsymbol{x} \in \mathcal{R}_1$. La probabilidad de error P_e puede, por tanto, obtenerse como:

$$P_e = \int_{\mathcal{R}_2} p(\boldsymbol{x}|\mathcal{X}_1)P(\mathcal{X}_1)d\boldsymbol{x} + \int_{\mathcal{R}_1} p(\boldsymbol{x}|\mathcal{X}_2)P(\mathcal{X}_2)d\boldsymbol{x} \tag{A.6}$$

Las integrales anteriores sobre un clasificador de Bayes se muestran en la figura A.1 para un sencillo ejemplo bidimensional. El resultado importante es que no solo en este caso la regla de decisión de Bayes obtiene el mínimo error de clasificación posible. Un clasificador de Bayes es siempre óptimo en el sentido de que ningún otro clasificador puede dar una probabilidad de error P_e menor. Este resultado no es tan sorprendente si se observa que un clasificador de Bayes tiene a su disposición toda la información sobre el problema en forma de probabilidades; en general, esta información no suele estar disponible en la práctica y los clasificadores deben encarar el problema desde otros enfoques.

Bibliografía

[1] Burges, Cristopher J. C., «A tutorial on support vector machines for pattern recognition», `http://svm.research.bell-labs.com/papers/tutorial_web_page.ps.gz`.

[2] Duda, Richard O. y Peter E. Hart, *Pattern recognition and scene analysis*, Jon Wiley & Sons, Nueva York, 1973.

[3] Gunn, Steve, «Support vector machines for classification and regression», inf. téc., Image Speech & Intelligent Systems Group, University of Southampton, mayo 1998, `http://www.isis.ecs.soton.ac.uk/resources/svminfo/svm.ps.gz`.

[4] Haykin, S., *Neural networks: a comprehensive foundation*, 2.ª ed., Prentice Hall, 1998.

[5] Kröse, Ben y Patrick van der Smagt, «An introduction to neural networks», nov. 1996, `ftp://ftp.fwi.uva.nl/pub/computer-systems/aut-sys/reports/neuro-intro/neuro-intro.ps.gz`.

[6] Stitson, M. O., J. A. E. Weston, A. Gammerman, V. Vovk y V. Vapnik, «Theory of support vector machines», inf. téc., Royal Holloway, University of London, dic. 1996, `http://www.dcs.rhbnc.ac.uk/research/compint/areas/comp_learn/sv/pub/report17.ps`.