# Fundamentos matemáticos del aprendizaje profundo

# (Notas de clase)

Julián Fernández Bonder

(1er cuatrimestre 2025)

IC - CONICET y Departamento de Matemática, FCEyN - Universidad de Buenos Aires, Ciudad Universitaria, Pabellón 0+infinito (1428) Buenos Aires, Argentina.

*Email address*: jfbonder@dm.uba.ar

*URL*: http://mate.dm.uba.ar/~jfbonder

# Índice general

# Introducción

En este curso intentaremos cubrir la base matemática que se esconde detrás de los algoritmos usados hoy en día basados en lo que se conoce como *machine learning, deep learning, redes neuronales, etc.*

Este tipo de algoritmos se encuentra presente en infinidad de aplicaciones que usamos en la vida cotidiana y son utilizados de manera habitual por las compañías tecnológicas más importantes de la actualidad, como Google, Microsoft, Facebook, IBM, Apple, Adobe, Netflix, NVIDIA, y Baidu.

¿Qué es una *red neuronal*? Una red neuronal, se puede definir como una colección de unidades de cómputo, llamadas *neuronas*, donde cada una produce como resultado un número real que se lo denomina *activación*. Las neuronas iniciales son activadas por sensores que perciben el entorno, mientras que otras neuronas son activadas por las precedentes activaciones. Este tipo de estructuras permite a las neuronas enviar mensajes entre ellas y, en consecuencia, reforzar las conexiones que llevan a resultados positivos en resolver el problema en cuestión y disminuyendo aquellas que llevan al fracaso.

Lo que intentaremos realizar en este curso es describir como funcionan estas redes neuronales desde un punto de vista matemático, teniendo en cuenta el éxito que estos métodos basados en redes neuronales no se determina por ensayo y error o por puro azar, sino que es un análisis matemático preciso.

El objetivo principal de este curso es el de presentar las principales ideas y conceptos detrás de las redes neuronales que habitualmente se utilizan de manera intuitiva, como conceptos matemáticos precisos. Para eso, haremos una mezcla (¡espero que divertida!) de matemática clásica junto con conceptos modernos de aprendizaje profundo (o deep learning).

Quiero enfatizar que el principal objetivo de este curso se centra en la matemática detrás de las redes neuronales (haciendo conexiones con MECÁNICA LAGRANGIANA, ANÁLISIS FUNCIONAL, CÁLCULO, GEOMETRÍA, ETC) pero no en la implementación de estos algoritmos. Los interesados en las implementaciones de estas ideas, pueden bien cursar los excelentes cursos dictados por el Departamento de Computación y/o consultar la excelente bibliografía disponible sobre el tema, p.ej. [**6, 7, 18, 2, 17, 13, 8**].

En este curso seguiremos el libro [**5**] pero existen otros excelentes textos que pueden consultarse y ser complementarios de este. Por ejemplo [**14, 16, 19**], etc.

Quiero resaltar que estas notas son una traducción de partes de [**5**] y recomiendo fuertemente abandonar estas notas y leer directamente ese libro.

# Parte 1

# Introducción a las redes neuronales

Capítulo 1

# Problemas introductorios

Empecemos discutiendo algunos problemas de la vida cotidiana que nos llevarán al concepto de *neurona abstracta* y *red neuronal*. Estos ejemplos, están basados en el problema de ajustar un cierto número de parámetros que detonan una cierta función de activación. Estos parámetros se ajustan para minimizar una *función de error*.

## 1.1. Agua en una tanque

Una cañería lleva agua a una presión dada $P$ (ver la Figura 1.1 **a**). La canilla $K$ puede ajustar la presión del agua, produciendo una flujo variable a una tasa $r$. La canilla influencia la tasa introduciendo un control $w$ de manera tal que $P = r/w$.

Un cierto número de cañerías de este tipo son usadas para verter agua en un tanque (ver Figura 1.1 **b**). Simultáneamente, el tanque es drenado a una tasa $R$. El objetivo de este problema es: Dadas las presiones de agua $P_1, \ldots, P_n$, cómo se pueden ajustar las canillas $K_1, \ldots, K_n$ de manera tal que luego de un período de tiempo fijo $t$ haya un determinado volumen $V$ de agua en el tanque.

Denotemos por $r_1, \ldots, r_n$ a los flujos de agua ajustados por las canillas $K_1, \ldots, K_n$. Si el flujo saliente $R$ excede la totalidad del flujo entrate (i.e. $R > \sum_{i=1}^{n} r_i$), entonces no se acumulará agua en el tanque (i.e. $V = 0$). Caso contrario, si el total del flujo entrante es mayor que el saliente (i.e. $R < \sum_{i=1}^{n} r_i$), entonces el agua se acumulará en el tanque a

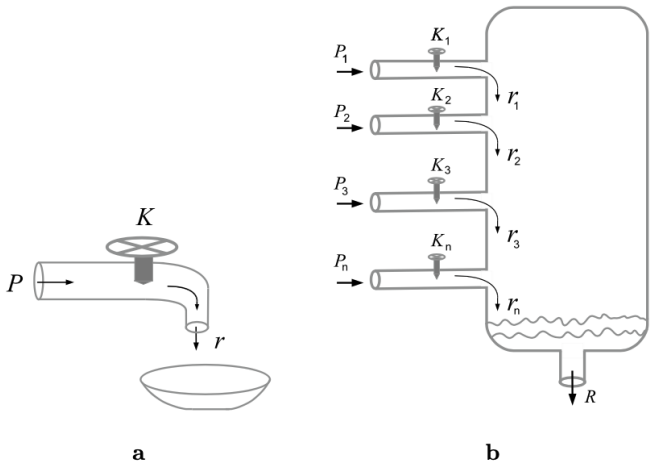



FIGURA 1.1. **a.** La canilla $K$ ajusta la presión $P$ en la cañería produciendo un flujo $r$. **b.** Las $n$ canillas $K_1, \ldots, K_n$ ajustan las presiones en las cañerías que vierten agua en el tanque con un desagüe teniendo un tasa de desagote $R$

una tasa igual a la diferencia de la tasa entrante y la saliente, con lo que la acumulación de agua sobre le intervalo de tiempo $t$ viene dado por $V = t\left(\sum_{i=1}^{n} r_i - R\right)$. Luego, la función que determina el volumen de agua total en el tanque luego de $t$ segundos, viene dado por la fórmula

$$V = \begin{cases} 0, & \text{si } R > \sum_{i=1}^{n} r_i \\ t\left(\sum_{i=1}^{n} r_i - R\right), & \text{caso contrario.} \end{cases}$$

Podemos escribir esta función de manera equivalente. Si notamos $w_i$ al control provisto por la canilla $K_i$, entonces la $i-$ésima cañería provee agua a una tasa $r_i = P_i w_i$. Consideramos también la *función de activación*

$$\varphi_t(x) = \begin{cases} 0, & \text{si } x < 0 \\ tx, & \text{caso contrario,} \end{cases}$$

que depende del parámetro temporal $t > 0$. Luego, el volumen de agua $V$ se puede escribir como

$$V = \varphi_t\left(\sum_{i=1}^{n} P_i w_i - R\right). \tag{1.1.1}$$

La tasa de salida $R$ produce una traslación horizontal al gráfico de $\varphi_t$ que en el lenguaje de las redes neuronales se denomina como *bias* o *sesgo*. El problema se reduce entonces en encontrar una solución para la ecuación (1.1.1) teniendo como incógnitas $w_i$ y $R$. En la práctica alcanza con obtener una buena solución aproximada eligiendo para eso controles $w_i$ y el sesgo $R$ de manera tal que la *función de error*

$$L(w_1, \ldots, w_n, R) = \frac{1}{2}\left(\varphi_t\left(\sum_{i=1}^{n} P_i w_i - R\right) - V\right)^2$$

sea minimizada. Otras funciones de error pueden ser utilizadas, como por ejemplo

$$\tilde{L}(w_1, \ldots, w_n, R) = \left|\varphi_t\left(\sum_{i=1}^{n} P_i w_i - R\right) - V\right|,$$

pero esta presenta el inconveniente de que no es diferenciable.

En la práctica queremos aprender de los datos disponibles. Para eso disponemos de $N$ datos de entrenamiento que consisten en presiones $\mathbf{P}^k = (P_1^k, \ldots, P_n^k)$ y volúmenes $V_k$, $1 \leq k \leq N$. Usando estos datos, deberíamos poder pronosticar el volumen $V$ para cualquier otro vector de presiones $\mathbf{P} = (P_1, \ldots, P_n)$.

Los controles $\mathbf{w} = (w_1, \ldots, w_n)$ se obtienen de los datos minimizando la función de costo

$$C(\mathbf{w}, R) = \frac{1}{2}\sum_{k=1}^{N}(\varphi_t(\mathbf{P}^k \cdot \mathbf{w} - R) - V_k)^2$$

y para esos controles se realiza el pronóstico usando la fórmula (1.1.1).

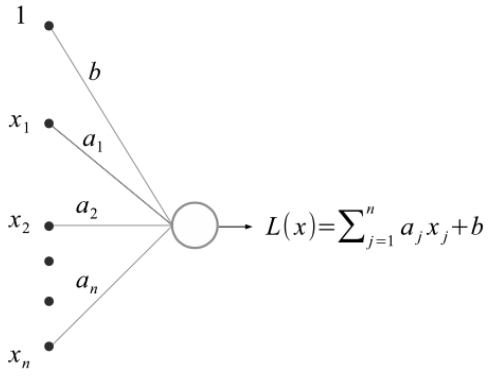



FIGURA 1.2. Una neurona de aproximación lineal

## 1.2. Regresión lineal

Las redes neuronales no son un concepto reciente. Aparecieron por primera vez bajo el nombre de regresión lineal a principios del s. XIX en trabajos de Gauss y Legendre. Veamos como un modelo de regresión lineal puede ser naturalmente interpretado como una red neuronal.

Consideremos una función continua a valores reales $f\colon K \subset \mathbb{R}^n \to \mathbb{R}$ donde $K$ es un conjunto compacto. Queremos construir una función lineal $L = L(\mathbf{x})$ que aproxime $f$ en el sentido de $L^2$ usando una neurona con una activación lineal. La entrada es la variable $\mathbf{x} = (x_1, \ldots, x_n) \in K$ y la salida es la función lineal $L(\mathbf{x}) = \sum_{j=1}^{n} a_j x_j + b$ donde $a_j$ son los pesos y $b$ es el sesgo de la neurona, ver la Figura 1.2.

Para simplificar el análisis, asumimos que la aproximación se realiza cerca del origen, i.e. $L(0) = f(0)$, lo que implica que el valor del sesgo es $b = f(0)$. Usando una traslación vertical, podemos siempre asumir que $f(0) = 0$, con lo que podemos tomar el sesgo $b = 0$. Luego, la función lineal es de la forma $L(\mathbf{x}) = L_{\mathbf{a}}(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x}$ donde $\mathbf{a} = (a_1, \ldots, a_n)$.

Consideramos como función de costo a la distancia $L^2$ entre la salida de la neurona y la función objetivo $f$,

$$C(\mathbf{a}) = \frac{1}{2} \int_K (L_{\mathbf{a}}(\mathbf{x}) - f(\mathbf{x}))^2 \, d\mathbf{x} = \frac{1}{2} \int_K (\mathbf{a} \cdot \mathbf{x} - f(\mathbf{x}))^2 \, d\mathbf{x}.$$

Calculemos el gradiente de la función $C$.

$$\begin{aligned} \frac{\partial C}{\partial a_k}(\mathbf{a}) &= \int_K x_k (\mathbf{a} \cdot \mathbf{x} - f(\mathbf{x})) \, d\mathbf{x} \\ &= \sum_{j=1}^{n} a_j \int_K x_j x_k \, d\mathbf{x} - \int_K x_k f(\mathbf{x}) \, d\mathbf{x} \\ &= \sum_{j=1}^{n} a_j r_{jk} - m_k \end{aligned}$$

Luego,

$$\nabla C(\mathbf{a}) = R\mathbf{a}^t - \mathbf{m}^t,$$

donde $R = (r_{jk})_{1 \le j,k \le n}$ y $\mathbf{m} = (m_1, \ldots, m_n)$. Observemos que $R = R^t$.

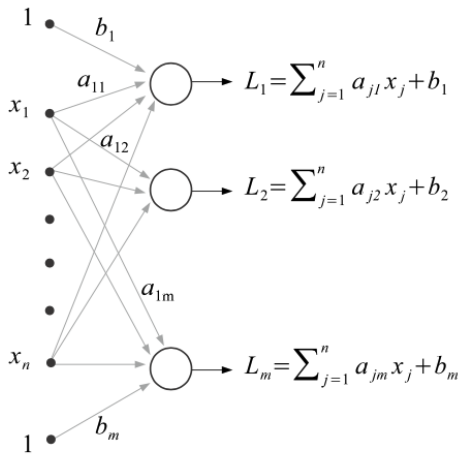



FIGURA 1.3. Una red neuronal de aproximación lineal

El valor óptimo de los pesos $\mathbf{a} = (a_1, \ldots, a_n)$ se alcanzan entonces cuando $\nabla C(\mathbf{a}) = 0$. Si la matriz $R$ resulta inversible, entonce $\mathbf{a}^t = R^{-1}\mathbf{m}^t$.

En general el cálculo de $R^{-1}$ (sobre todo cuando la dimensión $n$ es grande) suele ser costoso computacionalmente y aún cuando la dimensión es pequeña, la matriz puede ser mal condicionada y su cálculo introducir errores significativos. Es mejor para eso utilizar otros métodos de resolución como por ejemplo el método de *descenso de gradiente*. Más adelante veremos con detalle este método.

Se toma un vector de pesos inicial $\mathbf{a}_0$ y a partir de la misma se general la recursión

$$\mathbf{a}_{i+1} = \mathbf{a}_i - \lambda \nabla C(\mathbf{a}_i),$$

donde el parámetro $\lambda > 0$ se lo llama la *tasa de aprendizaje*. Se puede ver que esta sucesión $\{\mathbf{a}_i\}_{i\geq 1}$ aproxima a la solución buscada.

Cuando se busca aproximar una función a valores vectoriales $F\colon K \subset \mathbb{R}^n \to \mathbb{R}^m$ con $m > 1$, $F = (f_1, \ldots, f_m)$, queremos hallar una función lineal del tipo $L(\mathbf{x}) = A\mathbf{x}^t + \mathbf{b}^t$ donde ahora $A = (a_{jk}) \in \mathbb{R}^{m\times n}$ es una matriz y $\mathbf{b} = (b_1, \ldots, b_m) \in \mathbb{R}^m$ es un vector de sesgo.

Nuevamente, si queremos aplicar la aproximación en $x = 0$, sigue que el vector de sesgo $\mathbf{b} = F(0)$ está fijo. La salida de la $k-$ésima neurona estará dada por la $k-$ésima componente de $L$, $L_k(x) = \sum_{j=1}^{n} a_{jk}x_j + b_k$, ver la Figura 1.3.

La función de costo, será entonces

$$C(A) = \frac{1}{2}\int_K \|L(\mathbf{x}) - F(\mathbf{x})\|^2 \, d\mathbf{x} = \sum_{k=1}^{m} C_k(\mathbf{a}^k),$$

donde $\mathbf{a}^k = (a_{1k}, \ldots, a_{nk})$ es la $k-$ésima fila de la matriz $A$ y

$$C_k(\mathbf{a}^k) = \frac{1}{2}\int_K (L_k(\mathbf{x}) - f_k(\mathbf{x}))^2 \, d\mathbf{x} = \frac{1}{2}\int_K (\mathbf{a}^k \cdot \mathbf{x} + b_k - f_k(\mathbf{x}))^2 \, d\mathbf{x}.$$

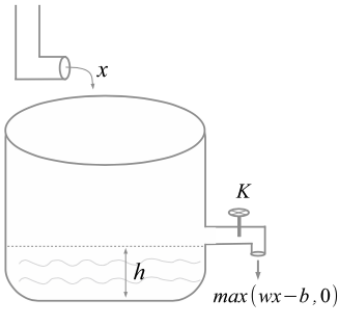



FIGURA 1.4. Un tanque de agua con una canilla y un flujo saliente $\phi(x) = \text{máx}\{wx - b, 0\}$

La minimización del costo $C(A)$ es equivalente a la minimización simultánea de los costos $C_k(\mathbf{a}^k)$, $1 \leq k \leq m$. Cada uno de esas funciones de costo es relativa a una neurona y hemos mostrado su procedimiento de optimización en el caso escalar.

## 1.3. La red de "fábrica de cócteles"

En este ejemplo introduciremos la idea de la *red de aprendizaje profundo* que será el tópico central de este curso.

Asumamos que tenemos un flujo entrante de agua a una tasa $r$ en un tanque que tiene una canilla $K$ que controla el flujo saliente del tanque. Ver la Figura 1.4.

La canilla está situada a una distancia $h$ del fondo del tanque. Luego de un cierto tiempo $t$, la cantidad de agua que fluyó dentro del tanque es $x = rt$. Notemos por $w$ al parámetro que controla el flujo saliente del tanque. Si $A$ es el área de la base del tanque, entonces $x - Ah$ representa el flujo sobrante y la cantidad $w(x - Ah) = wx - b$ sera eliminado del tanque por la canilla, donde $b = wAh$. El flujo saliente del tanque se modela como

$$\phi(x) = \text{máx}\{wx - b, 0\} = \begin{cases} 0, & \text{si } x \leq b/w \\ wx - b, & \text{caso contrario.} \end{cases}$$

Vamos ahora a organizar nuestros tanques en una red de seis tanques que mezclan tres ingredientes en un cóctel (ver la Figura 1.5). Azúcar, agua y vino se vierten en los tanques $A$, $B$ y $C$. Una mezcla de los tanques $A$ y $B$, se vierte al tanque $D$ como agua dulce y una mezcla de los tanques $B$ y $C$ se vierte al tanque $E$ como vino diluido. La mezcla producida por los contenidos de los tanques $D$ y $E$ se vierte al taque $F$. La canilla del tanque $F$ controla la producción final del cóctel.

La producción final del cóctel es le resultado de una combinación compleja de mezclas de los ingredientes, azúcar, agua y vino, vertidos en cantidades $x_1, x_2$ y $x_3$. Las canillas controlan la proporción de las soluciones que pasan al siguiente contenedor y son usadas para ajustar la calidad de la producción de cóctel. Luego de probar el resultado final, un experto en cóctel ajusta las canillas de manera tal que el cóctel mejore su calidad lo más posible. El proceso de ajuste de las canillas corresponde al aprendizaje de cómo hacer el cóctel de los ingredientes.

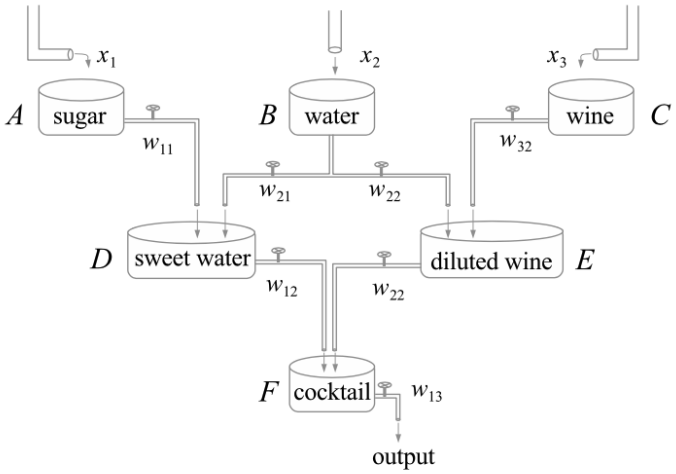



FIGURA 1.5. Una fábrica de cócteles como una red neuronal con dos capas escondidas

## 1.4. Resumen

Veamos ahora como las características comunes de los ejemplos nos van a llevar a desarrollar nuevos conceptos abstractos como el perceptron y la red neuronal.

En todos los ejemplos se empieza con información entrante proporcionada por las variables $x_1, \ldots, x_n$. En los ejemplos que vimos, estas son deterministas, pero podrían bien ser aleatorias (i.e. las mismas dependen de la observación realizada pero están distribuidas según una ley de probabilidad).

Otra característica en común es que los ingresos $x_i$ están multiplicados por un peso $w_i$ y luego esos productos están sumados formando expresiones del tipo $\sum_{i=1}^{n} w_i x_i$ que es el producto interno entre el vector de entradas $\mathbf{x} = (x_1, \ldots, x_n)$ y el vector de pesos $\mathbf{w} = (w_1, \ldots, w_n)$. En los ejemplos que vimos aparece restando un sesgo $b$. El mismo se puede eliminar introduciendo un nuevo peso $w_0 = b$ y un nuevo ingreso $x_0 = -1$ lo que lleva a $\sum_{i=0}^{n} w_i x_i = \sum_{i=1}^{n} w_i x_i - b$ que también es el producto interno de dos vectores.

Otro ingrediente en común es la función de activación, denotada por $\varphi(x)$, Vimos algunos ejemplos de funciones de activación: lineales, lineales a trozos. Veremos de más tipos.

La función de salida $y$ se obtiene aplicando la función de activación $\varphi$ a el producto interno mediante la expresión $y = \varphi(\mathbf{w} \cdot \mathbf{x})$. Si las variables de entrada $x_i$ son deterministas, la salida $y$ también lo será, pero si las entradas son variables aleatorias, la salida también será aleatoria.

El objetivo de estos ejemplos es el de ajustar el sistema de pesos $\mathbf{w}$ de manera tal que la función de salida $y$ aproxime una cierta función de manera más precisa posible. En algunos casos, esto puede hacerse de manera exacta, pero en la mayoría será sólo una aproximación. Este objetivo se alcanza eligiendo los pesos $\mathbf{w}$ de manera tal que cierta función de error sea minimizada. La función de error, es típicamente una función tipo distancia entre la salida realizada y la deseada.

Resumiendo, en los ejemplos se tienen los siguientes ingredientes: entradas $\{x_i\}$, pesos $\{w_i\}$, sesgo $b$, una función de activación $\varphi(x)$ y una función de error. Estas partes son usadas para construir una función aproximante, mediante el proceso de ajustar los pesos que se lo denomina *aprendizaje* de los datos observados. Más adelante veremos un concepto abstracto que reúne todas estas propiedades y contiene todas estas características.

## 1.5. Ejercicios

EJERCICIO 1.5.1. Una fábrica tiene $n$ proveedores donde cada uno produce diferentes cantidades por día, $x_1, \ldots, x_n$. La fábrica está conectada con sus proveedores por un sistema de rutas, que pueden ser utilizadas con capacidades variables $c_1, \ldots, c_n$, de manera tal que la fábrica queda provista por una cantidad diaria $x = c_1x_1 + \cdots + c_nx_n$.

(a) Dado que la producción de la fábrica empieza cuando el suministro alcanza una cantidad crítica diaria $b$, escribir la fórmula para los ingresos diarios de la fábrica $y$.
(b) Formular el problema como un problema de aprendizaje.

EJERCICIO 1.5.2. Un número $n$ de instituciones financieras, cada una teniendo una riqueza de $x_i$, deposita una cantidad de dinero en un fondo a una tasa ajustable para el depósito de $w_i$. Luego, el dinero en el fondo viene dado por $x = w_1x_1 + \cdots + w_nx_n$. El fondo está diseñado para funcionar de la siguiente forma: mientras el fondo tenga menos que una cierta reserva $M$, el manager del fondo no realiza inversiones. Sólo se invierte el dinero excedente a la reserva $M$. Sea $k = e^{rt}$, donde $r$ y $t$ denotan la tasa de retorno y el tiempo de inversión respectivamente.

(a) Hallar la fórmula para la inversión.
(b) Enunciar el problema de aprendizaje asociado.

EJERCICIO 1.5.3. (a) Dada una función continua $f\colon [0,1] \to \mathbb{R}$ hallar la función lineal $L(x) = ax + b$ tal que $L(0) = f(0)$ y que minimize el error cuadrático $\frac{1}{2}\int_0^1 (L(x) - f(x))^2\, dx$.
(b) Dada una función continua $f\colon [0,1] \times [0,1] \to \mathbb{R}$ hallar la función lineal $L(x) = ax + by + c$ tal que $L(0,0) = f(0,0)$ y que minimize el error cuadrático

$$\frac{1}{2}\int_0^1 \int_0^1 (L(x,y) - f(x,y))^2\, dxdy.$$

EJERCICIO 1.5.4. Dado un compacto $K \subset \mathbb{R}^n$, asociamos la matriz simétrica

$$r_{ij} = \int_K x_i x_j\, dx.$$

La inversibilidad de la matriz $R = (r_{ij})$ en general depende de $K$ y de la dimensión $n$.

(a) Probar que si $n = 2$ la matriz $R$ es inversible para cualquier compacto $K \in \mathbb{R}^2$.
(b) Probar que si $K = [0,1]^n$, entonces $R$ es inversible para todo $n \geq 1$.

Capítulo 2

# Funciones de activación

Para poder aprender una función no lineal objetivo, una red neuronal usa funciones de activación que son no lineales. La elección de una función de activación específica define tipos diferentes de redes neuronales. Dentro de la enorme variedad de funciones de activación que existen en la literatura, los tres tipos más comúnmente usados son: funciones lineales, funciones escalón, funciones sigmoidales y funciones *palo de hockey*.

Cada una de estas funciones tiene ventajas y desventajas (continuidad, diferenciabilidad, acotación, etc.) que aparecen a la hora de realizar diferentes cálculos.

## 2.1. Algunos ejemplos de funciones de activación

**Funciones lineales.** La pendiente de la función lineal puede ser usada para modelar la tasa de activación de una neurona. Se suele usar fundamentalmente en una red de multicapa para la capa de salida. Una neurona con una función de activación lineal es equivalente a una regresión lineal: $f(x) = kx$.

**Funciones escalón.** Este tipo de función de activación está inspirado en neuronas biológicas que exhiben saltos que simplificadamente modelan la activación de una neurona del sistema nervioso. Los ejemplos más comunes son:

*Función de Heaviside.* Está dada por

$$H(x) = \begin{cases} 0, & \text{si } x < 0 \\ 1, & \text{si } x \geq 0. \end{cases}$$

Si bien esta función no es diferenciable en el origen, si tiene derivada en sentido de funciones generalizadas o en *sentido distribucional*, $H'(x) = \delta(x)$ donde $\delta$ es la función *delta de Dirac*. La función generalizada $\delta(x)$ se caracteriza por las propiedades

$$\delta(x) = \begin{cases} 0, & \text{si } x \neq 0 \\ +\infty, & \text{si } x = 0, \end{cases} \qquad \text{y} \qquad \int_{\mathbb{R}} \delta(x)\, dx = 1.$$

*Función bipolar.* También se la conoce como la función signo y se define como

$$S(x) = \begin{cases} -1, & \text{si } x < 0 \\ 1, & \text{si } x \geq 0. \end{cases}$$

Se ve fácilmente que se tiene la relación $S(x) = 2H(x) - 1$ y en consecuencia $S'(x) = 2\delta(x)$.

**Funciones palo-de-hockey.** Estas son funciones cuyos gráficos son similares a los de un palo de hockey. Algunos ejemplos:

*Unidad lineal rectificada (ReLU).* En este caso, la activación es lineal sólo si $x \geq 0$.

$$\mathrm{ReLU}(x) = xH(x) = \text{máx}\{0, x\} = \begin{cases} 0, & \text{si } x < 0 \\ x, & \text{caso contrario.} \end{cases}$$

Observemos que $\mathrm{ReLU}'(x) = H(x)$.

*Unidad lineal rectificada paramétrica (PReLU).* En este caso la activación es lineal a trozos con distinta pendiente si $x < 0$ o $x \geq 0$.

$$\mathrm{PReLU}(\alpha, x) = \begin{cases} \alpha x, & \text{si } x < 0 \\ x, & \text{caso contrario.} \end{cases}$$

*Unidad lineal exponencial (ELU).* Esta función es lineal y positiva para $x > 0$ y negativa exponencial para $x < 0$.

$$\mathrm{ELU}(\alpha, x) = \begin{cases} x, & \text{si } x > 0 \\ \alpha(e^x - 1), & \text{si } x \leq 0 \end{cases}$$

Esta función es diferenciable en $x = 0$ si y sólo si $\alpha = 1$.

*Unidad lineal sigmoidal (SLU).* Buscamos una función que tenga un comportamiento similar a $\mathrm{ELU}(x)$, pero que sea $C^\infty$. Eso podemos obtenerlo de la forma

$$\mathrm{SLU}(x) = \phi(x) = x\sigma(x) = \frac{x}{1 + e^{-x}}.$$

Observemos que esta función es aproximadamente lineal para $x \gg 1$ y aproximadamente exponencial para $-x \gg 1$. Sin embargo no es monótona y cambia de signo.

**Funciones sigmoidales.** Este tipo de funciones de activación tienen la ventaja de ser suaves y que pueden aproximar a las funciones escalón con la precisión deseada. Cuando se normalizan para que sus valores estén en $[0, 1]$ su valor puede interpretarse como una probabilidad. Algunos ejemplos:

*Función logística.*

$$\sigma_c(x) = \sigma(c, x) = \frac{1}{1 + e^{-cx}},$$

donde e parámetro $c > 0$ controla la tasa de encendido de la neurona en el sentido de que valores grandes de $c$ corresponden a cambios rápidos de 0 a 1.

Observemos que cuando $c \to \infty$, tenemos que $\sigma(c, x) \to H(x)$ para $x \neq 0$.

*Tangente hiperbólica.* También se la llama función sigmoidal bipolar.

$$\mathbf{t}(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

**Funciones sombrero.** Estas funciones de activación se utilizan cuando la neurona es activada al máximo para un cierto valor del potencial de acción.

*Gaussiana.*

$$g(x) = e^{-x^2}$$

*Doble exponencial.*

$$f(x) = e^{-\lambda|x|}, \quad \lambda > 0.$$

## 2.2. Resumen

Cada unidad en una red neuronal usa una función que procesa la señal entrante, llamada la función de activación. La importancia de estas funciones de activación es la de introducir no-linealidades en la red. Activaciones lineales sólo pueden producir resultados lineales y sólo pueden ser utilizadas para una clasificación lineal (o problemas de regresión lineal), lio que resulta muy restrictivo. Las funciones de activación no lineales resuelven este problema.

Existen muchos tipos diferentes de funciones de activación que han sido propuestos en la literatura cada una de ellas para un propósito diferente. Algunas de las mismas las hemos visto en este capítulo.

## 2.3. Ejercicios

EJERCICIO 2.3.1. (a) Mostrar que la función logística $\sigma(x)$ verifica la desigualdad $0 < \sigma'(x) \leq \frac{1}{4}$ para $x \in \mathbb{R}$.

(b) ¿Cómo se modifica esa desigualdad para la función $\sigma_c(x)$?

EJERCICIO 2.3.2. Sean $S(x)$ y $H(x)$ las funciones signo y Heaviside respectivamente. Mostrar que

(a) $S(x) = 2H(x) - 1$,

(b) $\mathrm{ReLU}(x) = \frac{1}{2}x(S(x) + 1)$.

Capítulo 3

# Funciones de costo

En el proceso de aprendizaje, los parámetros de una red neuronal se deben ajustar para minimizar ciertas funciones objetivo que representan la medida de la proximidad entre lo que predice la red y el objetivo buscado. Esto es conocido con los nombres equivalentes de *funciones de costo*, *funciones de pérdida* o *funciones de error*. En este capítulo vamos a describir las funciones de costo más usuales en redes neuronales.

## 3.1. Entrada, salida y objetivo

La entrada de una red neuronal es obtenida de datos conocidos o de sensores que perciben el entorno. La entrada es la variable con la que se alimenta la red. Puede ser una variable unidimensional $x \in \mathbb{R}$, un vector $\mathbf{x} \in \mathbb{R}^n$, una matriz, o una variable aleatoria $X$.

La red en sí actúa como una función, es decir, modifica la variable de entrada de una cierta manera y produce una salida que puede ser, nuevamente, unidimensional $y \in \mathbb{R}$, un vector $\mathbf{y} \in \mathbb{R}^m$, una variable aleatoria $Y$, etc. La ley por la cual la entrada es modificada en la salida se realiza mediante un *mapeo ingreso-egreso*, $f_{w,b}$. El índice $(w, b)$ sugiere que los parámetros internos de la red mientras se realiza esta asignación están fijados en $(w, b)$. Siguiendo las notaciones previas, tendríamos $y = f_{w,b}(x)$, o $\mathbf{y} = f_{w,b}(\mathbf{x})$, o $Y = f_{w,b}(X)$.

La *función objetivo* es la relación deseada que la red trata de aproximar. Esta función es independiente de los parámetros $(w, b)$ y será denotada, según corresponda, por $z = \phi(x)$, o $\mathbf{z} = \phi(\mathbf{x})$, o $Z = \phi(X)$, según los datos sean unidimensionales, vectoriales o variables aleatorias. Podría eventualmente, haber alguna mezcla, por ejemplo $z = \phi(\mathbf{x})$ cuando la entrada sea vectorial, pero la salida unidimensional.

La red neuronal, ajusta los parámetros $(w, b)$ hasta que la variable de salida $y$ se encuentre dentro de un entorno de la variable objetivo $z$. Esta proximidad se puede medir de diferentes maneras, involucrando diferentes funciones de costo, $C(w, b) = \text{dist}(y, z)$, con una distancia a ser determinada. Los valores optimos de los parámetros de la red serán dados por $(w^*, b^*) = \arg\min_{(w,b)} C(w, b)$.

El proceso por el cual los parámetros $(w, b)$ se van ajustando a $(w^*, b^*)$ se lo llama *aprendizaje*. Equivalentemente, decimos que la red aprende la función objetivo $\phi(x)$. Este proceso de aprendizaje involucra la minimización de la función costo. En este capítulo vamos a investigar algunos tipos de función costo.

## 3.2. Ejemplos de función costo

Vamos a ver algunos ejemplos, pueden consultar otros adicionales en el libro **[5]**.

**3.2.1. Función de error de supremo.** Asumamos que la red neuronal, que toma valores de entrada $x \in [0,1]$, debe aprender una función continua $\phi\colon [0,1] \to \mathbb{R}$. Si $f_{w,b}$ es el mapa entrada-salida de la red, entonces la función costo supremo asociada es

$$C(w,b) = \sup_{x\in[0,1]} |f_{w,b}(x) - \phi(x)|.$$

A todo uso práctico, la función objetivo es conocida en finitos puntos

$$z_i = \phi(x_i), \qquad 1 \leq i \leq n,$$

luego la función costo usada es

$$C(w,b) = \max_{1\leq i\leq n} |f_{w,b}(x_i) - z_i|.$$

**3.2.2. Función de error $L^2$.** Como en el caso anterior, supongamos que la variable de entrada es $x \in [0,1]$ y la función objetivo es $\phi\colon [0,1] \to \mathbb{R}$. Si $f_{w,b}$ es el mapa entrada-salida de la red, la función costo que mide la distancia $L^2$ entre la salida y el objetivo es

$$C(w,b) = \int_0^1 (f_{w,b}(x) - \phi(x))^2\, dx.$$

Nuevamente, si se conoce al a función objetivo sólo en $n$ puntos,

$$z_i = \phi(x_i), \qquad 1 \leq i \leq n,$$

entonces la función costo es la distancia Euclideana en $\mathbb{R}^n$

$$C(w,b) = \sum_{i=1}^{n} (f_{w,b}(x_i) - z_i)^2 = \|f_{w,b}(\mathbf{x}) - \mathbf{z}\|^2,$$

donde $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{z} = (z_1, \ldots, z_n)$ y $f_{w,b}(\mathbf{x}) = (f_{w,b}(x_1), \ldots, f_{w,b}(x_n))$.

Observemos que para $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n$ fijos, $C(w,b)$ es una función suave y la misma puede ser minimizada, por ejemplo, por el método de descenso de gradiente que estudiaremos en el siguiente capítulo.

**3.2.3. Función de error cuadrático medio.** Consideremos ahora una red neuronal cuya entrada es una variable aleatoria $X$, y su salida es la variable aleatoria $Y = f_{w,b}(X)$, donde $f_{w,b}$ es el mapa de entrada-salida de la red. Asumamos que la red es usada para aproximar a la variable aleatoria objetivo $Z$.

La función de error en este caso, mide la proximidad de la salida $Y$ con el objetivo $Z$. Una forma habitual de medir esa proximidad es mediante la esperanza de la diferencia al cuadrado

$$C(w,b) = \mathbb{E}[(Y - Z)^2] = \mathbb{E}[(f_{w,b}(X) - Z)^2].$$

Para hallar el mínimo $(w^*, b^*) = \arg\min C(w,b)$ se utiliza alguno de los algoritmos de minimización que veremos en el capítulo siguiente.

Veamos algunas de las razones de por qué esta función de error (y la vista en la sección 3.2.2) son muy utilizadas.

1. Las funciones de cuadrado integrable sobre un espacio de probabilidad, forma un *espacio de Hilbert* con producto interno dado por $\langle X, Y\rangle = \mathbb{E}[XY]$. Esto define una norna $\|X\|^2 = \mathbb{E}[X^2]$ la cual induce una distancia $\operatorname{dist}(X, Y) = \|X - Y\|$.

   Luego, la función costo no es otra cosa que el cuadrado de la distancia $C(w, b) = \operatorname{dist}(Y, Z)^2$.
2. Otro motivo es la relación que existe entre la función de costo y la esperanza condicional. La red neuronal transforma a la variable aleatoria de entrada en otra de salida $Y = f_{w,b}(X)$ que es parametrizada por $w$ y $b$. La información generada por la variable aleatoria $f_{w,b}(X)$ es la $\sigma-$álgebra $\mathcal{E}_{w,b} = \sigma(f_{w,b}(X))$ (es decir, la menor $\sigma-$álgebra que hace medible a $f_{w,b}(X)$). Todas estas $\sigma-$álgebras generan toda la información de salida $\mathcal{E} = \bigvee_{w,b} \mathcal{E}_{w,b}$ que es la $\sigma-$álgebra generada por la unión $\bigcup_{w,b} \mathcal{E}_{w,b}$. (La notación $\mathcal{E}$ viene de *exit*, salida en inglés).

   En general, la variable aleatoria objetivo $Z$ no está determinada por la información $\mathcal{E}$. El problema entonces puede formularse como: *dada la información de salida $\mathcal{E}$, hallar la mejor aproximación de $Z$ basada en la información $\mathcal{E}$.* Esta es una variable aleatoria que se denota por $Y = \mathbb{E}[Z|\mathcal{E}]$, que se la llama la *esperanza condicional* de $Z$ dado $\mathcal{E}$. El mejor predictor, $Y$, queda determinado entonces por $\mathcal{E}$ (es decir, es $\mathcal{E}-$medible) y es el que minimiza la distancia a $Z$ entre todas las variables aleatorias $U$ que son $\mathcal{E}-$medibles. En otras palabras, $Y$ verifica
   $$\operatorname{dist}(Y, Z) = \min_{U,\ \mathcal{E}-\text{medible}} \operatorname{dist}(U, Z).$$
   Notamos que $Y$ es la proyección ortogonal de $Z$ sobre el espacio de las funciones $\mathcal{E}-$medibles (con respecto al producto interno $\langle X, Y\rangle = \mathbb{E}[XY]$).
3. La definición de función de costo puede extenderse fácilmente al caso en el que las variables aleatorias son conocidas por mediciones. Consideremos $n$ mediciones de las variables aleatorias $(X, Z)$ dadas por $(x_1, z_1), \ldots, (x_n, z_n)$. Esto forma lo que se denomina como el *conjunto de entrenamiento* de la red neuronal. Entonces, en este caso la función de costo se define como
   $$\tilde{C}(w, b) = \frac{1}{n} \sum_{i=1}^{n} (f_{w,b}(x_i) - z_i)^2,$$
   que es la media empírica del cuadrado de la diferencia entre $Y$ y $Z$.

Observemos que en el caso trivial en el que las variables $X$ y $Z$ son independientes, la red neuronal es poco lo que puede hacer. En efecto, si $X$ y $Z$ son independientes, entonces también lo serán $f_{w,b}(X)$ y $Z$ con lo que la información de salida $\mathcal{E}$ va a ser independiente de $Z$. En consecuencia, nuestro mejor estimador será
$$Y = \mathbb{E}[Z|\mathcal{E}] = \mathbb{E}[Z].$$
Es decir que el mejor estimador es simplemente un número que es el promedio de mis mediciones.

**3.2.4. Entropía cruzada.** Vamos a presentar la definición y propiedades de la entropía cruzada para el caso unidimensional. Más adelante lo usaremos en el caso de dimensión 2.

Sean $p$ y $q$ dos densidades en $\mathbb{R}$. Se define la *función de verosimilitud negativa* a

$$-\ell_q(x) = -\ln q(x).$$

Esta función mide la información proporcionada por $q(x)$. Observemos que si $q(x)$ es grande, quiere decir que estamos en regiones de alta probabilidad con lo que no tenemos mayor detalle de lo que ocurre y en ese caso $-\ell_q(x) \leq 0$. Mientras que si $q(x) \sim 0$, estamos en regiones de baja probabilidad, con lo que ese dato nos da información mucho más precisa y $-\ell_q(x) \gg 1$.

La función de entropía cruzada de $p$ con respecto a $q$ se define como la esperanza con respecto a $p$ de la función de verosimilitud negativa de $q$, es decir

$$S(p,q) = \mathbb{E}^p[-\ell_q] = -\int_{\mathbb{R}} p(x) \ln q(x)\, dx.$$

Esta función mide la información proporcionada por $q$ desde el punto de vista de la distribución $p$ que se supone conocida. Para densidades discretas, tenemos que $q \in (0,1)$ entonces $S(p,q) \geq 0$. Sin embargo tenemos un resultado más fino:

Proposición 3.2.1. *Se tiene la desigualdad $S(p,q) \geq H(p)$, donde $H(p)$ es la* entropía de Shannon *definida como*

$$H(p) = S(p,p) = -\mathbb{E}^p[\ell_p] = -\int_{\mathbb{R}} p(x) \ln p(x)\, dx.$$

Demostración. Evaluemos la diferencia de ambos términos:

$$\begin{aligned} S(p,q) - H(p) &= -\int_{\mathbb{R}} p(x) \ln q(x)\, dx + \int_{\mathbb{R}} p(x) \ln p(x)\, dx \\ &= -\int_{\mathbb{R}} p(x) \ln \frac{q(x)}{p(x)}\, dx. \end{aligned}$$

Ahora usamos la desigualdad elemental $\ln t \leq t - 1$ para $t > 0$ y obtenemos

$$\begin{aligned} S(p,q) - H(p) &\geq \int_{\mathbb{R}} p(x) \left( \frac{q(x)}{p(x)} - 1 \right) dx \\ &= \int_{\mathbb{R}} q(x)\, dx - \int_{\mathbb{R}} p(x)\, dx = 1 - 1 = 0. \end{aligned}$$

Luego $S(p,q) - H(p) \geq 0$ y, además, $S(p,q) = H(p)$ si y sólo si $p = q$. □

Este resultado nos dice que dada una densidad $p$, el mínimo de la entropía cruzada $S(p,q)$ se alcanza cuando $q = p$ y ese mínimo coincide con la entropía de Shannon de la densidad $p$.

Observemos que en el caso de densidades continuas, la entropía $H(p)$ puede ser finita (positiva o negativa) o bien puede ser infinita, pero si $p$ es discreta, entonces la entropía siempre es finita y positiva.

Para densidades continuas con esperanza y varianza finita, la entropía de Shannon es siempre finita.

Corolario 3.2.2. *Sea $X$ una variable aleatoria sobre $\mathbb{R}$ con densidad $p$. Asumimos que $X$ tiene esperanza y varianza finita. Entonces*

$$H(p) \leq \frac{1}{2} \ln(2\pi e \mathbb{V}(X)),$$

*donde* $\mathbb{V}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$ *es la varianza.*

DEMOSTRACIÓN. Llamemos $\mu = \mathbb{E}[X]$ y $\sigma^2 = \mathbb{V}[X]$. Definamos ahora la densidad normal $q$ con misma esperanza y varianza que $p$, es decir, $q(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. Observemos entonces que por la Proposición 3.2.1 se tiene que

$$H(p) \leq S(p, q).$$

Ahora, la entropía cruzada de $p$ con respecto a $q$ se puede calcular explícitamente:

$$\begin{aligned} S(p,q) &= -\int_R p(x) \ln q(x)\, dx = \frac{1}{2}\ln(2\pi\sigma^2) + \frac{1}{2\sigma^2}\int_{\mathbb{R}} p(x)(x-\mu)^2\, dx \\ &= \frac{1}{2}\ln(2\pi\sigma^2) + \frac{1}{2} = \frac{1}{2}\ln(2\pi e\sigma^2). \end{aligned}$$

Esto finaliza la demostración. Observemos que la igualdad se alcanza para variables aleatorias normalmente distribuidas. □

OBSERVACIÓN 3.2.3. La entropía cruzada se utiliza por lo general en problemas de clasificación, mientras que el error cuadrático medio es usado en problemas de regresión.

## 3.3. Funciones de costo y regularización

Para evitar los problemas de sobreajuste de los datos de entrenamiento (overfitting), es conveniente agregar términos adicionales a las funciones de costo. Se puede observar que el modelo suele sobreajustar los datos de entrenamiento si se permite que los parámetros tomen valores arbitrariamente grandes. Sin embargo, si se restringen los valores de los parámetros a permanecer acotados, esto impide que el modelo pase por todos los puntos de entrenamiento y por ende previene el sobreajuste.

Para lograr esta restricción, se suelen usar términos de regularización a las funciones de costo. Los términos más usuales son tipo $L^1$ o $L^2$.

*Regulariación* $L^2$. Consideremos los parámetros $\mathbf{w} \in \mathbb{R}^n$. Su norma $L^2$ se define como $\|\mathbf{w}\|_2^2 = \sum_{i=1}^n w_i^2$. La función de costo con la regularización $L^2$ se obtiene sumando un múltiplo de la norma $L^2$ a la función de costo original

$$L_2(\mathbf{w}) = C(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2,$$

donde $\lambda > 0$ es un multiplicador de Lagrange que regula el equilibrio entre el tamaño de los pesos y el mínimo de $C(\mathbf{w})$. Un valor grande de $\lambda$ significa un valor pequeño de los pesos a costa de un valor grande para $C(\mathbf{w})$. Análogamente, un valor pequeño de $\lambda$ permite valores grandes de los pesos y un valor más pequeño del costo, con el riesgo de que el modelo sobreajuste. El valor del *hiperparámetro* $\lambda$ debe ser seleccionado para conseguir minimizar el sobreajuste sin aumentar demasiado el costo.

*Regularización* $L^1$. La norma $L^1$ de $\mathbf{w} \in \mathbb{R}^n$ se define como $\|\mathbf{w}\|_1 = \sum_{i=1}^n |w_i|$. Luego, la función costo con regularización $L^1$ se define como

$$L_1(\mathbf{w}) = C(\mathbf{w}) + \lambda\|\mathbf{w}\|_1.$$

El rol del hiperparámetro $\lambda$ es similar al caso $L^2$. Observemos que la norma $L^1$ no es diferenciable con lo que algunos métodos de minimización pueden no funcionar adecuadamente.

*Regularización potencial.* Esta es una generalización de las dos regularizaciones previas. Se considera una función $U\colon \mathbb{R}^n \to \mathbb{R}_{>0}$ que satisface

1. $U(\mathbf{x}) = 0$ si y sólo si $\mathbf{x} = 0$
2. $U$ tiene un mínimo global en $x = 0$.

En el caso en que $U$ sea una función de clase $C^2$, la condición 2 se verifica si $\nabla U(0) = 0$ y $D^2U(0) \geq 0$. La función $U$, llamada *función potencial*, es una generalización de las normas $L^1$ y $L^2$.

La función de costo regularizada se define entonces como

$$G(\mathbf{w}) = C(\mathbf{w}) + \lambda U(\mathbf{w}), \qquad \lambda > 0.$$

La elección del potencial óptimo que posea las mejores propiedades de regularización para una función de costo dada, se realiza verificando su performance en el test del error (ver la sección siguiente). El test del error debe decrecer significativamente cuando el término $U(\mathbf{w})$ es agregado a la función de costo inicial $C(\mathbf{w})$.

## 3.4. Entrenamiento y test de errores

Una de las principales características que hace diferente el aprendizaje automático (machine learning) a un problema regular de optimización, que minimiza sólo un error, es el problema de doble optimización con dos tipos diferentes de errores que serán discutidos en esta sección. Es la diferencia entre estos dos errores lo que determinará qué tan bien funciona el algoritmo de aprendizaje automático.

En un enfoque basado en aprendizaje automático el conjunto de datos $\{(x_i, z_i)\}_{i\in I}$ es dividido en tres partes: *conjunto de entrenamiento*, *conjunto de testeo* y *conjunto de validación*. Se asume que todos estos conjuntos poseen idéntica distribución, siendo generados por una distribución de probabilidad en común. Otra hipótesis es que cada uno de estos conjuntos de datos son independientes entre sí. En cuando a los tamaños, el más grande de estos conjuntos de datos es el de entrenamiento $\mathcal{T}$ que suele tener aproximadamente el 70 % de los datos. El segundo es el de testeo $T$ que tiene aproximadamente el 20 % de los datos. Finalmente el de validación $\mathcal{V}$ con aproximadamente el 10 %.

La función de costo evaluada en los datos del conjunto de entrenamiento se llama el *error de entrenamiento*. Similarmente, la función de costo evaluada en los datos del conjunto de testeo se llama *error de testeo* o *error de generalización*. Por ejemplo,

$$C_{\mathcal{T}}(w, b) = \frac{1}{m}\sum_{j=1}^{m}(f_{w,b}(x_j) - z_j)^2, \qquad (x_j, z_j) \in \mathcal{T}$$

$$C_T(w, b) = \frac{1}{k}\sum_{i=1}^{k}(f_{w,b}(x_i) - z_i)^2, \qquad (x_i, z_i) \in T,$$

con $m = \#\mathcal{T}$ y $k = \#T$, son los errores de entrenamiento y de testeo respectivamente asociados al promedio de la suma de los cuadrados.

En una primera etapa, un proceso de optimización (como el método de descenso de gradiente) es usado para minimizar el error de entrenamiento $C_{\mathcal{T}}(w, b)$ ajustando

los parámetros $(w, b)$. Notemos por $(w^*, b^*)$ los valores óptimos de los parámetros. Este procedimiento se llama *entrenamiento*.

En una segunda etapa evaluamos el error de testeo en los valores optimales de los parámetros que hayamos, obteniendo un error de testeo $C_T(w^*, b^*)$. En general, se espera obtener la desigualdad

$$C_{\mathcal{T}}(w^*, b^*) \leq C_T(w^*, b^*).$$

Las siguientes variantes son posibles:

1. Ambos errores, $C_{\mathcal{T}}(w^*, b^*)$ y $C_T(w^*, b^*)$, son pequeños. Esto significa que la red generaliza bien, es decir que no sólo responde bien en el conjunto de entrenamiento, sino que también en datos no usados. Este es el escenario deseado en un algoritmo de aprendizaje automático.
2. El error de entrenamiento $C_{\mathcal{T}}(w^*, b^*)$ es pequeño, pero el error de testeo $C_T(w^*, b^*)$ es aún grande. En este caso la red no generaliza bien. Lo que sucede es que la red sobreajusta al conjunto de entrenamiento. El caso extremo es que el error de entrenamiento sea cero, en ese caso la red *memoriza* los datos de entrenamiento en el sistema de pesos y sesgos. En estos casos se precisa usar una técnica de regularización. Si luego de esto el error de testeo no disminuye, probablemente haya que modificar la arquitectura de la red. Una forma es disminuyendo el número de parámetros. Típicamente, sobre-parametrizar la red lleva a un sobreajuste.
3. Ambos errores, $C_{\mathcal{T}}(w^*, b^*)$ y $C_T(w^*, b^*)$, son grandes. En este caso, la red neuronal subajusta los datos de entrenamiento. Para solucionar este problema, necesitamos aumentar la capacidad de la red cambiando su arquitectura a una red con más parámetros.

El conjunto de validación, $\mathcal{V}$, es usado para ajustar los hiperparámetros (tasa de aprendizaje, profundidad de la red, etc.) de manera tal que el error de validación sea el menor posible. Según **[5]**, "Finding the optimal hyperparameters is more like an art rather than science, depending on the scientist's experience, and we shall not deal with it here".

Otro tema importante a discutir es el *sobre entrenamiento* y el *tiempo óptimo de parada* del entrenamiento. Entrenando durante un tiempo suficientemente largo (involucrando un número suficientemente grande de ciclos) el error de entrenamiento se puede hacer, en general, muy pequeño. Al principio del período de entrenamiento, el error de testeo también disminuye, hasta que se llega a un punto luego del cual comienza a crecer lentamente. El tiempo de parada óptimo del entrenamiento es cuando el error de testeo llega a su mínimo. Luego de ese instante, la diferencia entre el error de testeo y el de entrenamiento comienza a crecer, un hecho que lleva al sobreajuste. Es decir, la red funciona muy bien con los datos de entrenamiento, pero no con los datos de prueba. Ver Figura 3.4.

## 3.5. Significado geométrico

Si $C(w, b)$ y $U(w)$ son suaves, entonces la función de costo regularizada $G(w, b) = C(w, b) + \lambda U(w)$ también resulta suave y su mínimo se realiza para

$$(w^*, b^*) = \arg\min(C(w, b) + \lambda U(w)).$$

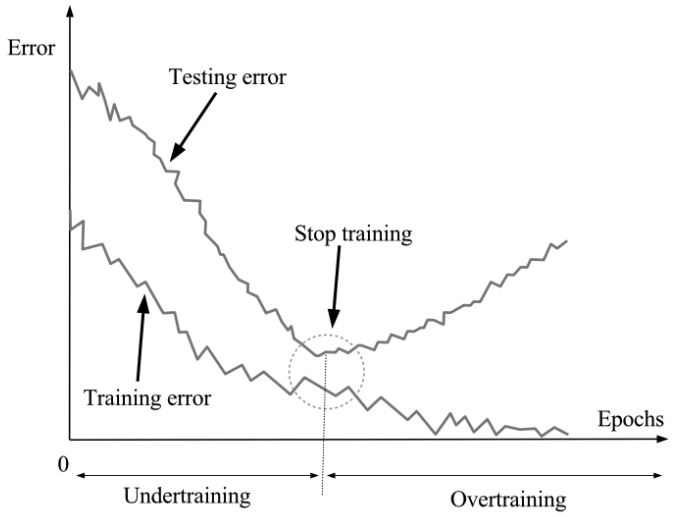



FIGURA 3.1. Regiones de sobre y sub entrenamiento para una red neuronal.

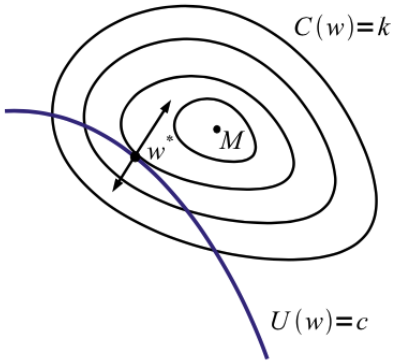



FIGURA 3.2. La función de costo $C(w, b)$ tiene un mínimo en $M$; sus superficies de nivel vienen dadas por $S_k = \{C(w, b) = k\}$. Una de esas superficies de nivel es tangente a la superficie $\{U(w) = c\}$ en un punto en donde los gradientes $\nabla C$ y $\nabla U$ son colineales.

En este punto mínimo, el gradiente se tiene que anular. En particular $\nabla_w C(w^*, b^*) = -\lambda \nabla_w U(w^*)$. Esto significa que los vectores normales a las superficies de nivel de $C(w, b)$ y $U(w)$ son colineales. Esto ocurre cuando las superficies de nivel son tangentes. Ver Figura 3.5. Como los vectores normales a las superficies de nivel son colineales y tienen sentido opuesto, resulta que $\lambda > 0$.

**La importancia de $\lambda$.** Para poder entender mejor el rol del multiplicador $\lambda$, asumamos por simplicidad que $U(w) = \|w\|_2^2$. Si $w^*$ es un punto de contacto entre $\{C(w, b) = k\}$ y $\{U(w) = c\}$ (ver Figura 3.5), entonces la ecuación $\nabla_w C(w^*, b^*) = -\lambda \nabla_w U(w^*)$ se convierte en $\nabla_w C(w^*, b^*) = -2\lambda w^*$. Esto implica que

$$\|\nabla_w C(w^*, b^*)\| = 2\lambda\sqrt{c},$$

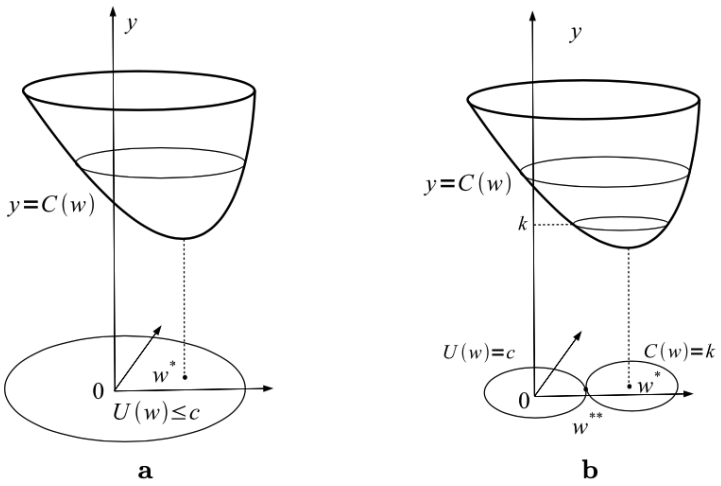



FIGURA 3.3. La función costo $C(w, b)$ en dos posibilidades distintas: **a.** $w^* \in \{U(w) \leq c\}$ **b.** $w^* \notin \{U(w) \leq c\}$

es decir que la magnitud del vector normal a la superficie $\{C(w, b) = k\}$ en $w^*$ depende tanto de $\lambda$ como de $c$. Veamos las siguientes observaciones:

1. Asumamos que $\nabla_w C(w^*, b^*) \neq 0$. Entonces valores pequeños (grandes) de $c$ se corresponden con valores grandes (pequeños) de $\lambda$. Equivalentemente, valores pequeños de $\lambda$ se corresponden a valores grandes de los pesos $w$ y valores grandes de $\lambda$ se corresponden con valores pequeños de $w$.
2. Asumamos ahora que $\nabla_w C(w^*, b^*) = 0$. Entonces $\lambda = 0$ o $c = 0$. La condición $c = 0$ es equivalente a $w = 0$, lo que implica que $w^* = 0$, es decir que la función de costo $C(w, b)$ tiene un mínimo global en $w = 0$. Dado que la mayoría de las funciones de costo no verifican esto, podemos concluir que $\lambda = 0$ en este caso.

**El rol de la restricción.** Sea $w^* = \arg\min C(w, b)$ y asumamos que $w^* \neq 0$.

Se tienen dos casos posibles:

1. $w^* \in \{w \in \mathbb{R}^n : U(w) \leq c\}$. Este caso corresponde al caso en que $c$ es suficientemente grande tal que el mínimo de $C(w, b)$ está contenido en el interior de la superficie $\{U(w) = c\}$. Ver Figura 3.5 **a**. En este caso, elegimos $\lambda = 0$.
2. $w^* \notin \{w \in \mathbb{R}^n : U(w) \leq c\}$. Esta situación describe el caso en que $c$ es suficientemente pequeño para que el mínimo de $C(w, b)$ esté afuera del dominio $D_c = \{U(w) \leq c\}$. En este caso el mínimo de $C(w, b)$ sobre el dominio $D_c$ va a localizarse en el borde, sobre la superficie $\{U(w) = c\}$, en un punto $w^{**}$ por el que pase una superficie de nivel de $C(w, b)$ tangente a la superficie $\{U(w)) = c\}$, ver Figura 3.5 **b**. Las incógnitas $w^{**}$ y $\lambda$ satisfacen las ecuaciones

$$\partial_{w_k} C(w^{**}) = -\lambda \partial_{w_k} U(w^{**}), \quad U(w^{**}) = c.$$

Para obtener una aproximación de la solución de esta la ecuación, minimizamos $G(w, b) = C(w, b) - \lambda U(w)$ usando un método de descenso de gradiente. El hiperparámetro $\lambda$ es ajustado para que el mínimo de $G$ sea lo menor posible.

**Eliminación del sobreajuste.** Hemos visto que en el caso en que el error de entrenamiento $C_{\mathcal{T}}(w,b)$ es pequeño, mientras que el error de testeo $C_T(w,b)$ es grande, tenemos un problema de sorbeajuste.

Una forma de intentar solucionar este problema es requerir que el vector de pesos $w$ sea pequeño. En este caso tanto el error de testeo como el error de entrenamiento. En este caso ambos errores van a estar próximos. Una forma de ver esto es la siguiente. Si asumimos que $w \to 0$, entonces

$$Y = f_{w,b}(X) \simeq f_{0,b}(X) = \phi(b^{(L)}),$$

donde $b^{(L)}$ es el sesgo de las neuronas en la última capa. Notemos que $\phi(b^{(L)})$ es independiente de la entrada $X$, así que será el mismo valor ya sea si es parte del testeo o del entrenamiento.

## 3.6. Resumen

Durante el proceso de aprendizaje, las redes neuronales tratan de igualar sus salidas a los objetivos dados. Este proceso de aproximación involucra el uso de funciones de costo, que miden la diferencia entre lo que es predicho (la salida) y lo que es deseado (el objetivo). Estas proximidades son de varios tipos, dependiendo de lo que la red está tratando de aprender: una función, una variable aleatoria, una densidad de probabilidad, etc. Algunas de estas funciones de costo son funciones distancia, mientras que otras no. Sin embargo todas miden la separación de la salida respecto del objetivo.

Algunas veces, para eliminar el problema de sobreajuste, la función de costo es aumentada con un término de regularización. Estos términos dependen de los pesos y tienen la tarea de minimizar el costo sujeto a pesos de magnitudes pequeñas. El coeficiente del término de regularización es un hiperparámetro que controla el compromiso entre el tamaño de los pesos y el mínimo de la función de costo. El valor de este hiperparámetro es ajustado usando el conjunto de validación.

El conjunto de datos disponibles es dividido en tres partes que son usados para los siguientes propósitos: entrenamiento, testeo y validación. Cuando la función de costo es calculada usando los datos del conjunto de entrenamiento y se optimiza, se obtiene el error de entrenamiento. Cuando se usa el conjunto de datos de testeo, se obtiene el error de testeo. Un pequeño error de entrenamiento junto a un error de testeo grande es una señal de sobreajuste. Un error de entrenamiento grande es una señal de subajuste. El propósito de la regularización es obtener un menor error de testeo.

## 3.7. Ejercicios

Ejercicio 3.7.1. Sean $p, p_i, q, q_i$ funciones de densidad en $\mathbb{R}$ y $\alpha \in \mathbb{R}$. Mostrar que la entropía cruzada verifica las siguientes propiedades:

(a) $S(p_1 + p_2, q) = S(p_1, q) + S(p_2, q)$
(b) $S(\alpha p, q) = \alpha S(p, q) = S(p, q^\alpha)$
(c) $S(p, q_1 q_2) = S(p, q_1) + S(p, q_2)$.

Ejercicio 3.7.2. Verificar que la entropía cruzada verifica

$$S(p,q) \geq 1 - \int_{\mathbb{R}} p(x)q(x)\,dx.$$

EJERCICIO 3.7.3. Sea $X$ una variable aleatoria discreta. Mostrar que $H(X) \geq 0$.

EJERCICIO 3.7.4. Asumamos que la variable objetivo $Z$ es $\mathcal{E}-$medible. ¿Cuál es la función de error cuadrático medio en este caso?

EJERCICIO 3.7.5. Supongamos que la red tiene una función de entrada-salida $f_{w,b}$ lineal en $w$ y en $b$. Probar que la función de costo

$$C(w,b) = \sum_{i=1}^{n} (f_{w,b}(x_i) - z_i)^2,$$

alcanza su mínimo para un par de parámetros $(w^*, b^*)$ que pueden ser calculado explícitamente.

Capítulo 4

# Algoritmos de minimización

El proceso de aprendizaje en aprendizaje supervisado consiste en ajustar los parámetros de la red (pesos y sesgos) hasta que una cierta función de costo es minimizada. Dado que el número de parámetros es muy grande, es necesario contar con un algoritmo de minimización que sea robusto. En este capítulo veremos algunos algoritmos de minimización y discutiremos ventajas y desventajas de cada uno de estos algoritmos.

## 4.1. Propiedades generales de los mínimos

En esta sección revisaremos algunos conceptos básicos sobre los mínimos de funciones de una o varias variables. Estas técnicas analíticas teóricas sólo son eficientes si el número de variables no es muy grande. Sin embargo, en aprendizaje automático, el número de variables es del orden de miles, o más. Luego, estos métodos analíticos clásicos no son eficientes para aplicaciones concretas. Los discutiremos por completitud y porque son la base de construcción de métodos más sofisticados y robustos.

**4.1.1. Funciones de una variable real.** Recordemos que si tenemos una función continua $f\colon [a,b] \to \mathbb{R}$, definida en un intervalo compacto, esta es acotada y alcanza sus cotas dentro del intervalo $[a,b]$. Es decir, existe (por lo menos) un valor $c \in [a,b]$ tal que

$$f(c) = \min_{x\in[a,b]} f(x).$$

Este es el *mínimo global* para $f(x)$. Sin embargo, la función puede también tener mínimos locales. Más aún, si el valor mínimo (ya sea local o global) se encuentra en el interior del intervalo, i.e. $c \in (a,b)$, y si la función es diferenciable, entonces tenemos que $f'(c) = 0$. Geométricamente, esto significa que la recta tangente al gráfico de $y = f(x)$ en el punto $(c, f(c))$ es horizontal, Recordemos que esta condición es necesaria, pero no suficiente. Si, adicionalmente, la función es convexa (es decir, verifica $f''(x) \geq 0$), entonces esta condición resulta suficiente.

**4.1.2. Funciones de varias variables reales.** Sea $K$ un conjunto compacto en $\mathbb{R}^n$, es decir es un conjunto cerrado y acotado. Por ejemplo, $K = [a_1, b_1] \times \cdots \times [a_n, b_n]$ con $a_i, b_i \in \mathbb{R}$, o $K = \{\mathbf{x} \in \mathbb{R}^n\colon \|\mathbf{x}\|_2 \leq R\}$, con $R > 0$. Si $f\colon K \to \mathbb{R}$ es continua, entonces existe un punto $\mathbf{c} \in K$ tal que

$$\frac{\partial f}{\partial x_i}(\mathbf{c}) = 0, \quad i = 1, \ldots, n.$$

En notación vectorial, este sistema de ecuaciones se puede escribir como $\nabla f(\mathbf{c}) = 0$.

Esta condición geométricamente es equivalente a que el plano tangente al gráfico $z = f(\mathbf{x})$ en el punto $(\mathbf{c}, f(\mathbf{c}))$ sea horizontal (i.e. paralelo al hiperplano $z = 0$ en $\mathbb{R}^{n+1} = \{(\mathbf{x}, z)\colon \mathbf{x} \in \mathbb{R}^n, z \in \mathbb{R}\}$).

El desarrollo de Taylor de orden 2 de $f(\mathbf{x})$ en un entorno de $\mathbf{x} = \mathbf{c}$ es

$$\begin{aligned} f(\mathbf{x}) =& f(\mathbf{c}) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\mathbf{c})(x_i - c_i) \\ &+ \frac{1}{2} \sum_{j,k=1}^{n} \frac{\partial^2 f}{\partial x_j \partial x_k}(\mathbf{c})(x_j - c_j)(x_k - c_k) + o(\|\mathbf{x} - \mathbf{c}\|^2) \end{aligned}$$

Si llamamos $Hf$ o $D^2 f$ a la matriz Hessiana o matriz de derivadas segundas, dada por

$$Hf(\mathbf{x}) = D^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_n \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{pmatrix},$$

el desarrollo de Taylor se puede reescribir como

$$f(\mathbf{x}) = f(\mathbf{c}) + \nabla f(\mathbf{c}) \cdot (\mathbf{x} - \mathbf{c}) + \frac{1}{2}(\mathbf{x} - \mathbf{c}) Hf(\mathbf{c}) (\mathbf{x} - \mathbf{c})^t + o(\|\mathbf{x} - \mathbf{c}\|^2).$$

Observemos que la matriz Hessiana resulta simétrica si $f \in C^2$, es decir $Hf(\mathbf{x}) = Hf(\mathbf{x})^t$.

Si se verifica que $\nabla f(\mathbf{c}) = 0$, entonces el desarrollo de Taylor queda

$$f(\mathbf{x}) = f(\mathbf{c}) + \frac{1}{2}(\mathbf{x} - \mathbf{c}) Hf(\mathbf{c}) (\mathbf{x} - \mathbf{c})^t + o(\|\mathbf{x} - \mathbf{c}\|^2).$$

Asumamos ahora que la matriz Hessiana es definida positiva, i.e.

$$\mathbf{v} Hf(\mathbf{c}) \mathbf{v}^t > 0, \quad \forall v \in \mathbb{R}^n \setminus \{0\}.$$

Si llamamos $\Phi(\mathbf{x}) = \mathbf{x} Hf(\mathbf{c}) \mathbf{x}^t$ la forma cuadrática asociada a la matriz Hessiana, esta es una función continua y si llamamos $K = \{\mathbf{x} \in \mathbb{R}^n \colon \|\mathbf{x}\| = 1\}$, este conjunto resulta compacto y por ende existe $\mathbf{y} \in K$ tal que

$$0 < m = \Phi(\mathbf{y}) = \min_{\mathbf{x} \in K} \Phi(\mathbf{x}).$$

Por otro lado, observemos que la forma cuadrática es *homogénea de grado 2*, es decir:

$$\Phi(\lambda \mathbf{x}) = (\lambda \mathbf{x}) Hf(\mathbf{c}) (\lambda \mathbf{x})^t = \lambda^2 (\mathbf{x} Hf(\mathbf{c}) \mathbf{x}^t) = \lambda^2 \Phi(\mathbf{x}).$$

Luego, si $\mathbf{x} \neq 0$, tenemos que $\mathbf{x}/\|\mathbf{x}\| \in K$, luego

$$0 < m \leq \Phi\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right) = \frac{1}{\|\mathbf{x}\|^2} \Phi(\mathbf{x}),$$

Luego, obtenemos la siguiente desigualdad para la matriz Hessiana:

$$\mathbf{x} Hf(\mathbf{c}) \mathbf{x}^t \geq m \|\mathbf{x}\|^2, \qquad (m > 0). \tag{4.1.1}$$

Con la ayuda de (4.1.1) podemos demostrar el siguiente criterio

PROPOSICIÓN 4.1.1. *Sea $f$ una función de clase $C^2$ y $\mathbf{c} \in \mathbb{R}^n$ una solución de $\nabla f(\mathbf{c}) = 0$. Si la matriz Hessiana de $f$ en $\mathbf{c}$. $Hf(\mathbf{c})$ es definida positiva, entonces $\mathbf{c}$ es un mínimo local de $f$.*

DEMOSTRACIÓN. Por lo desarrollado anteriormente, tenemos que $f$ admite el desarrollo

$$f(\mathbf{x})-f(\mathbf{c})=\frac{1}{2}(\mathbf{x}-\mathbf{c})Hf(\mathbf{c})(\mathbf{x}-\mathbf{c})^t+o(\|\mathbf{x}-\mathbf{c}\|^2)\geq\frac{m}{2}\|\mathbf{x}-\mathbf{c}\|^2+o(\|\mathbf{x}-\mathbf{c}\|^2), \tag{4.1.2}$$

donde hemos usado (4.1.1) en la desigualdad. El término de error, verifica

$$\lim_{\mathbf{x}\to\mathbf{c}}\frac{o(\|\mathbf{x}-\mathbf{c}\|^2)}{\|\mathbf{x}-\mathbf{c}\|^2}=0,$$

luego, dado $\varepsilon>0$ existe $\delta>0$ tal que

$$\frac{|o(\|\mathbf{x}-\mathbf{c}\|^2)|}{\|\mathbf{x}-\mathbf{c}\|^2}<\varepsilon,\quad \text{si } \|\mathbf{x}-\mathbf{c}\|<\delta.$$

En particular, tomando $\varepsilon=m/4$, tenemos que existe $\delta>0$ tal que

$$-\frac{m}{4}\|\mathbf{x}-\mathbf{c}\|^2\leq o(\|\mathbf{x}-\mathbf{c}\|^2)\quad \text{si } \|\mathbf{x}-\mathbf{c}\|<\delta. \tag{4.1.3}$$

Combinando (4.1.2) y (4.1.3) obtenemos

$$f(\mathbf{x})-f(\mathbf{c})\geq\frac{m}{4}\|\mathbf{x}-\mathbf{c}\|^2\geq 0,$$

de donde obtenemos el resultado. □

EJEMPLO 4.1.2 (Matriz Hessiana en dimensión 2). Si $f$ es de clase $C^2$ en $\mathbb{R}^2$, entonces si matriz Hessiana tiene la forma

$$Hf=\begin{pmatrix}f_{xx} & f_{xy}\\ f_{xy} & f_{yy}\end{pmatrix}.$$

Si $\mathbf{v}=(a,b)\in\mathbb{R}^2$, entonces la forma cuadrática asociada es de la forma

$$\mathbf{v}Hf\mathbf{v}^t=f_{xx}a^2+2f_{xy}ab+f_{yy}b^2.$$

Realizando cálculos elementales, esta expresión puede escribirse como

$$\mathbf{v}Hf\mathbf{v}^t=f_{xx}\left(a+\frac{f_{xy}}{f_{xx}}b\right)^2+\frac{b^2}{f_{xx}}\underbrace{(f_{xx}f_{yy}-f_{xy}^2)}_{\det Hf}.$$

De esta expresión se deduce fácilmente lo siguiente:

1. Si $f_{xx}>0$ y $\det Hf>0$, entonces $Hf$ es definida positiva.
2. Si $f_{xx}<0$ y $\det Hf>0$, entonces $Hf$ es definida negativa.
3. Si $\det Hf<0$, entonces $Hf$ es indefinida.

EJEMPLO 4.1.3 (Funciones cuadráticas). Sea $A\in\mathbb{R}^{n\times n}$ una matriz simétrica y definida positiva y sean $\mathbf{b}\in\mathbb{R}^n$ y $d\in\mathbb{R}$. Consideremos la siguiente función

$$f(\mathbf{x})=\frac{1}{2}\mathbf{x}A\mathbf{x}^t-\mathbf{b}\cdot\mathbf{x}+d,\quad \mathbf{x}\in\mathbb{R}^n.$$

En coordenadas,

$$f(x_1,\ldots,x_n)=\frac{1}{2}\sum_{i,j=1}^{n}a_{ij}x_ix_j-\sum_{k=1}^{n}b_kx_k+d.$$

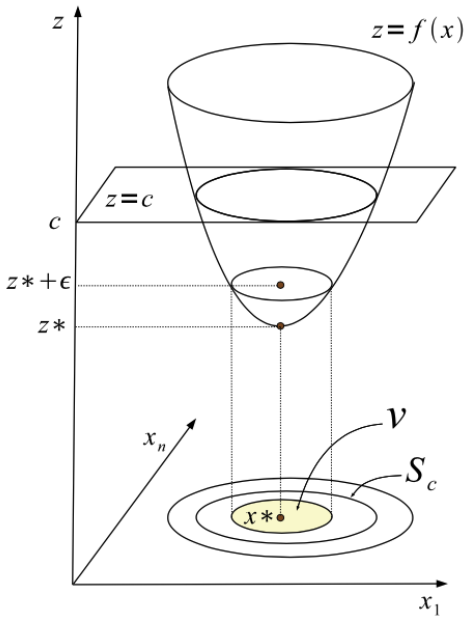



FIGURA 4.1. Conjuntos de nivel para la función $z = f(\mathbf{x})$.

La función $f\colon \mathbb{R}^n \to \mathbb{R}$ resulta suave con $(\nabla f(\mathbf{x}))^t = A\mathbf{x}^t - \mathbf{b}^t$, $Hf(\mathbf{x}) = A$. Al ser $A$ definida positiva, $f$ resulta convexa, tiene un único punto crítico dado por $\mathbf{c}^t = A^{-1}\mathbf{b}^t$ que resulta ser un mínimo global.

Si a la función cuadrática $f$ la restringimos a un conjunto compacto $K \subset \mathbb{R}^n$, puede ser que el punto crítico $\mathbf{c}$ pertenezca o no a $K$. Luego, en el caso en que $\mathbf{c} \notin K$ no podemos hallar los mínimos como puntos críticos de $f$. Más aún, esos puntos críticos van a pertenecer a la frontera de $K$, $\partial K$ y para hallarlos habrá que utilizar algún método alternativo (p.ej. multiplicadores de Lagrange).

## 4.2. Algoritmos de descenso de gradiente

El algoritmo de descenso de gradiente es un procedimiento para hallar el mínimo de una función navegando a través de las curvas de nivel asociadas siguiendo la dirección de máximo decrecimiento.

**4.2.1. Conjuntos de nivel.** Consideremos a la función $z = f(\mathbf{x})$, $\mathbf{x} \in D \subset \mathbb{R}^n$ con $n \geq 2$ y definamos el conjunto de nivel $c \in \mathbb{R}$ como

$$\mathcal{S}_c = f^{-1}(\{c\}) = \{\mathbf{x} \in D\colon f(\mathbf{x}) = c\}.$$

El *Teorema de la función implícita* nos permite asegurar que si $f \in C^1$ y $\nabla f(\mathbf{x}) \neq 0$ para $\mathbf{x} \in \mathcal{S}_c$, el conjunto de nivel $\mathcal{S}_c$ resulta ser una hipersuperficie (es decir, una superficie de dimensión $n-1$). A la familia $\{\mathcal{S}_c\}_{c\in\mathbb{R}}$ se la llama la familia de *hipersuperficies de nivel* de la función $f$. Para $n = 2$ se las llama las *curvas de nivel*. Geométricamente, las hipersuperficies de nivel se obtienen intersectando el gráfico $z = f(\mathbf{x})$ con el plano horizontal $z = c$. Ver Figura 4.2.1.

PROPOSICIÓN 4.2.1. *El gradiente $\nabla f$ es perpendicular a $\mathcal{S}_c$.*

DEMOSTRACIÓN. Sea $\mathbf{x}_0 \in \mathcal{S}_c$ y consideremos una curva $\mathbf{x}(t)$, $t \in (-\delta, \delta)$ tal que $\mathbf{x}(0) = \mathbf{x}_0$ y $\mathbf{x}(t) \in \mathcal{S}_c$ para todo $t \in (-\delta, \delta)$. Luego, el vector tangente a la curva en $t = 0$, $\mathbf{v} = \mathbf{x}'(0)$ es un vector tangente a $\mathcal{S}_c$ en el punto $\mathbf{x}_0$.

Por otro lado, como $\mathbf{x}(t) \in \mathcal{S}_c$ para todo $t \in (-\delta, \delta)$, se tiene que $f(\mathbf{x}(t)) = c$ para todo $t \in (-\delta, \delta)$. Luego si llamamos $\phi(t) = f(\mathbf{x}(t))$, tenemos que $\phi\colon (-\delta, \delta) \to \mathbb{R}$ es constante, por ende $\phi'(t) = 0$. Pero

$$0 = \phi'(t) = \frac{d}{dt} f(\mathbf{x}(t)) = \nabla f(\mathbf{x}(t)) \cdot \mathbf{x}'(t).$$

Evaluando en $t = 0$ obtenemos

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{v} = 0.$$

Como $\mathbf{v}$ es un vector tangente arbitrario, concluimos lo deseado. □

Asumamos ahora que la función $z = f(\mathbf{x})$ tiene un mínimo local estricto en $\mathbf{x}^* \in D$, es decir, existe un entorno $\mathcal{V}$ de $\mathbf{x}^*$ tal que $f(\mathbf{x}^*) < f(\mathbf{x})$ para todo $\mathbf{x} \in \mathcal{V} \backslash \{\mathbf{x}^*\}$. Notemos por $z^* = f(\mathbf{x}^*)$ el valor mínimo de $f$ en $\mathcal{V}$. Adicionalmente, asumamos que $\nabla f(\mathbf{x}) \neq 0$ para $\mathbf{x} \in \mathcal{V} \setminus \{\mathbf{x}^*\}$. Luego tenemos que $\mathcal{S}_{z^*} = \{\mathbf{x}^*\}$, que $\mathcal{S}_c \subset \mathcal{V}$ para $c \in [z^*, z^* + \varepsilon)$ con $\varepsilon > 0$ pequeño. Más aún, la familia de hipersuperficies $\{\mathcal{S}_c\}_{c \in (z^*, z^* + \varepsilon)}$ está *anidada*, es decir, si $z^* < c_1 < c_2 < z^* + \varepsilon$, entonces $\mathcal{S}_{c_1} \subset \mathrm{Int}(\mathcal{S}_{c_2})$. Ver Figura 4.2.1.

Ojo ahora, creo que la demostración de [**5**, Lemma 4.2.2] está mal, y en consecuencia están mal los Teoremas 4.2.3 y 4.2.4. En su reemplazo vamos a dar una demostración alternativa de un resultado análogo al enunciado en [**5**, Theorem 4.2.3].

TEOREMA 4.2.2. *Asumamos que $f\colon D \subset \mathbb{R}^n \to \mathbb{R}$ es de clase $C^2$, que $\mathbf{x}^* \in D$ es un mínimo local estricto de $f$ y que existe un entorno $\mathcal{V} \subset D$ de $\mathbf{x}^*$ tal que $\nabla f(\mathbf{x}) \neq 0$ para $\mathbf{x} \in \mathcal{V} \setminus \{\mathbf{x}^*\}$. Entonces, dado $\mathbf{x}_0 \in \mathcal{V}$, existe una curva $\gamma\colon [0, 1] \to \mathbb{R}^n$ tal que*

1. $\gamma(0) = \mathbf{x}_0$;
2. $\gamma(1) = \mathbf{x}^*$;
3. $\gamma'(t)$ *es perpendicular a* $\mathcal{S}_{f(\gamma(t))}$ *para* $t \in [0, 1)$.

Para la demostración de este teorema, vamos a usar el siguiente resultado de estabilidad de Lyapunov que puede encontrarse en [**3**].

TEOREMA 4.2.3 (Teorema de estabilidad de Lyapunov). *Sea $F\colon D \subset \mathbb{R}^n \to \mathbb{R}^n$ un campo vectorial de clase $C^1$ y sea $\mathbf{x}^* \in D$ tal que $F(\mathbf{x}^*) = 0$. Sea $U\colon \mathcal{V} \subset D \to \mathbb{R}$, donde $\mathcal{V}$ es un entorno de $\mathbf{x}^*$, una función de clase $C^1$ tal que*

1. $U(\mathbf{x}^*) = 0$ *y* $U(\mathbf{x}) > 0$ *para todo* $\mathbf{x} \in \mathcal{V} \setminus \{\mathbf{x}^*\}$.
2. $\nabla U(\mathbf{x}) \cdot F(\mathbf{x}) < 0$ *para* $\mathbf{x} \in \mathcal{V} \setminus \{\mathbf{x}^*\}$.

*(Una tal función $U$ que verifique 1 y 2 se la llama* función de Lyapunov*)*

*Entonces $\mathbf{x}^*$ es un punto de equilibrio del sistema de ecuaciones diferenciales ordinarias*

$$\mathbf{x}'(t) = F(\mathbf{x}(t)) \tag{4.2.1}$$

*que es* asintóticamente estable. *Es decir, que dado $\varepsilon > 0$, existe $\delta > 0$ tal que si $\|\mathbf{x}_0 - \mathbf{x}^*\| < \delta$, entonces $\|\mathbf{x}(t) - \mathbf{x}^*\| < \varepsilon$ para $t > 0$, donde $\mathbf{x}(t)$ es la solución de (4.2.1) que verifica $\mathbf{x}(0) = \mathbf{x}_0$ y $\lim_{t \to \infty} \mathbf{x}(t) = \mathbf{x}^*$.*

Con la ayuda del Teorema 4.2.3 demostremos el Teorema 4.2.2

DEMOSTRACIÓN DEL TEOREMA 4.2.2. Observemos que si llamamos $U(\mathbf{x}) = f(\mathbf{x})$, entonces $U$ es una función de Lyapunov para la ecuación

$$\mathbf{x}' = -\nabla f(\mathbf{x}). \tag{4.2.2}$$

En efecto, es muy simple verificar que se cumplen las condiciones 1 y 2 del Teorema 4.2.3.

Luego, si llamamos $\alpha(t)$ a la solución de (4.2.2) tal que $\alpha(0) = \mathbf{x}_0$, por el Teorema 4.2.3 tenemos que $\alpha(t) \to \mathbf{x}^*$ cuando $t \to \infty$.

Observemos que $\alpha\colon [0,\infty) \to \mathbb{R}^n$ verifica

1. $\alpha(0) = \mathbf{x}_0$;
2. $\alpha(t) \to \mathbf{x}^*$ cuando $t \to \infty$;
3. $\alpha'(t) = -\nabla f(\alpha(t))$, entonces $\alpha'(t)$ es perpendicular a $\mathcal{S}_{f(\alpha(t))}$ para todo $t > 0$.

Esto casi es lo que pide el teorema. Para obtener la curva $\gamma$ pedida, la construimos a partir de $\alpha$ de la forma:

$$\gamma(t) = \begin{cases} \mathbf{x}^* & \text{si } t = 1 \\ \alpha(1 - \frac{1}{1-t}) & \text{si } 0 \le t < 1. \end{cases}$$

Entonces $\gamma$ resulta continua, $\gamma(0) = \alpha(0) = \mathbf{x}_0$ y

$$\gamma'(t) = -\frac{1}{(1-t)^2}\alpha'(1 - \tfrac{1}{1-t}) = \frac{1}{(1-t)^2}\nabla f(\alpha(1 - \tfrac{1}{1-t})) = \frac{1}{(1-t)^2}\nabla f(\gamma(t)),$$

por lo que $\gamma'(t)$ resulta perpendicular a $\mathcal{S}_{f(\gamma(t))}$ para $t \in [0,1)$ por la Proposición 4.2.1. □

La construcción de la curva del Teorema 4.2.2 no es de utilidad a efectos computacionales. Para poder implementar la construcción de la curva, necesitamos poder aproximarla por una poligonal $\mathcal{B}_m = [\mathbf{x}_0\mathbf{x}_1\cdots\mathbf{x}_m]$ con las siguientes propiedades:

1. $\mathbf{x}_k \in \mathcal{S}_{c_k}$;
2. $c_{k+1} < c_k$, para $k = 0, 1, \ldots, m-1$;
3. el segmento $[\mathbf{x}_k\mathbf{x}_{k+1}]$ es perpendicular a $\mathcal{S}_{c_k}$.

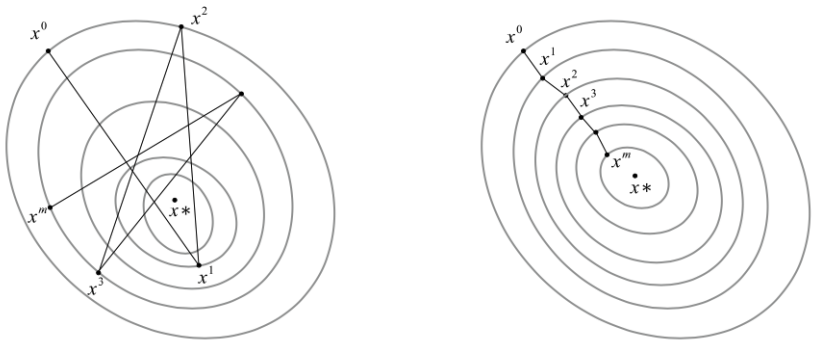
El algoritmo de construcción es el siguiente: empezamos con un punto $\mathbf{x}_0$ en un entorno del mínimo $\mathbf{x}^*$. Elegimos un parámetro $\eta > 0$ pequeño y nos movemos una distancia $\eta$ de $\mathbf{x}_0$ siguiendo la dirección normal a $S_{c_0}$ hacia el interior de la región, es decir en la dirección de $-\nabla f(\mathbf{x}_0)$:

$$\mathbf{x}_1 = \mathbf{x}_0 - \eta\frac{\nabla f(\mathbf{x}_0)}{\|\nabla f(\mathbf{x}_0)\|}.$$

Repetimos este procedimiento y así obtenemos la sucesión de puntos

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta\frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|}.$$

Este procedimiento se continúa y luego de $m$ pasos se llega al punto $\mathbf{x}_m$ que esperamos sea una buena aproximación del punto buscado $\mathbf{x}^*$.




**a** **b**

FIGURA 4.2. La poligonal $\mathcal{B}_m = [\mathbf{x}_0 \cdots \mathbf{x}_m]$ en dos casos: **a**. El paso $\eta$ es grande. **b**. El paso $\eta$ es pequeño.

¿Cómo se puede elegir el número de pasos $m$? En otras palabras, ¿cómo sabemos cuándo detener el algoritmo? El mejor criterio para detener el algoritmo es continuar siempre que se verifique que $c_{k+1} < c_k$. Para un tamaño de paso $\eta > 0$ fijo, existe el primer $m$ tal que se verifican las propiedades 1, 2 y 3 para $k < m$ y $c_{m+1} \geq c_m$. Mientras más pequeño sea el paso $\eta$, más grande será el orden de parada $m$ y más próximos estaremos de $\mathbf{x}^*$. Cuando $\eta \to 0$, la línea poligonal $\mathcal{B}_m$ aproxima a la curva $\gamma$ dada por el Teorema 4.2.2.

Si el algoritmo se detiene luego de $m$ pasos, se tienen las siguientes cotas

$$\|\mathbf{x}_0 - \mathbf{x}^*\| - m\eta \leq \|\mathbf{x}_m - \mathbf{x}^*\| \leq \operatorname{diam}(\mathcal{S}_{c_m}), \tag{4.2.3}$$

donde $\operatorname{diam}(\mathcal{S}_c)$ es el diámetro del conjunto $\mathcal{S}_c$. Esta estimación, entre otras cosas, indica que una cota razonable para el número de pasos $m$ viene dado por

$$m < \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|}{\eta}.$$

Para demostrar las cotas de (4.2.3), se razona como sigue: Usando la desigualdad triangular

$$\begin{aligned}\|\mathbf{x}_0 - \mathbf{x}^*\| &\leq \|\mathbf{x}_0 - \mathbf{x}_1\| + \|\mathbf{x}_1 - \mathbf{x}_2\| + \cdots + \|\mathbf{x}_{m-1} - \mathbf{x}_m\| + \|\mathbf{x}_m - \mathbf{x}^*\| \\ &= m\eta + \|\mathbf{x}_m - \mathbf{x}^*\|.\end{aligned}$$

Para la segunda desigualdad de (4.2.3), se tiene

$$\|\mathbf{x}^* - \mathbf{x}_m\| \leq \max_{\mathbf{y} \in \mathcal{S}_{c_m}} \|\mathbf{x}^* - \mathbf{y}\| \leq \sup_{\mathbf{x},\mathbf{y} \in \mathcal{S}_{c_m}} \|\mathbf{x} - \mathbf{y}\| = \operatorname{diam}(\mathcal{S}_{c_m}).$$

La elección del paso $\eta$ en la construcción de la poligonal es muy importante. Se pueden tener los casos:

1. Si $\eta$ es grande, el algoritmo para muy tempranamente y no se obtiene una buena aproximación de $\mathbf{x}^*$. Ver Figura 4.2.1 **a**;
2. Si $\eta$ es demasiado pequeño, la cantidad de pasos que se tienen es muy grande y puede ser muy costoso computacionalmente. Ver Figura 4.2.1 **b**.

En general, el tamaño del paso $\eta$ resulta de un compromiso entre el margen de error y el tiempo de corrida de la aplicación. Más adelante formalizaremos esta idea.

**4.2.2. Derivadas direccionales.** Otro concepto que usaremos es el de *derivada direccional*, que mide la tasa de variación instantánea de una función en un punto al moverse en una dirección dada. Más precisamente, si $\mathbf{u}$ es un vector unitario en $\mathbb{R}^n$, $f\colon U \subset \mathbb{R}^n \to \mathbb{R}$ y $\mathbf{x}_0 \in U$, la derivada direccional de $f$ en $\mathbf{x}_0$ en la dirección $\mathbf{u}$ se define como

$$\frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_0) = \lim_{h\to 0} \frac{f(\mathbf{x}_0 + t\mathbf{u}) - f(\mathbf{x}_0)}{h},$$

cuando ese límite existe.

Si la función $f$ es diferenciable en $\mathbf{x}_0$ entonces la derivada direccional en $\mathbf{x}_0$ existe en cualquier dirección $\mathbf{u}$ y la misma se calcula como

$$\frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) \cdot \mathbf{u}.$$

**4.2.3. El método de descenso más rápido.** El *método de descenso más rápido* (*steepest descent method* en inglés), también llamado *método de descenso de gradiente*, es un método numérico basado en un algoritmo voraz (greedy algorithm) por el cual el mínimo de una función es buscado dirigiendo cada paso en la dirección en donde la función decrece más rápidamente.

Para aplicar este método, estamos interesados en encontrar una dirección unitaria $\mathbf{u}$ en la cual $f$ decrezca tanto como sea posible en un paso de tamaño $\eta$. Para eso, usamos la aproximación lineal

$$f(\mathbf{x}_0 + \eta\mathbf{u}) - f(\mathbf{x}_0) = \eta \nabla f(\mathbf{x}_0) \cdot \mathbf{u} + o(\eta^2).$$

Como asumimos que $0 < \eta \ll 1$, podemos despreciar el efecto del término de error cuadrático $o(\eta^2)$. Luego, para obtener la dirección $\mathbf{u}$ que haga la variación lo más negativo posible, usamos la desigualdad de Cauchy-Schwartz para el producto escalar

$$-\|\nabla f(\mathbf{x}_0)\|\|\mathbf{u}\| \leq \nabla f(\mathbf{x}_0) \cdot \mathbf{u} \leq \|\nabla f(\mathbf{x}_0)\|\|\mathbf{u}\|.$$

Como $\|\mathbf{u}\| = 1$ es fácil ver que la igualdad inferior es alcanzada sí y sólo si

$$\mathbf{u} = -\frac{\nabla f(\mathbf{x}_0)}{\|\nabla f(\mathbf{x}_0)\|}.$$

Luego, el cambio más grande en la función es aproximadamente igual a

$$f(\mathbf{x}_0 + \eta\mathbf{u}) - f(\mathbf{x}_0) \simeq \eta \nabla f(\mathbf{x}_0) \cdot \mathbf{u} = -\eta\|\nabla f(\mathbf{x}_0)\|.$$

La constante $\eta$ es llamada la *tasa de aprendizaje*. De la relación previa, el cambio en la función en cada paso es proporcional a la magnitud del gradiente y a la tasa de aprendizaje.

El algoritmo consiste en la siguiente iteración que construye la siguiente sucesión $\{\mathbf{x}_k\}_{k\geq 0}$:

1. Elegir un punto inicial $\mathbf{x}_0$ en la base de atracción del mínimo global de $f$, $\mathbf{x}^*$.
2. Construir la sucesión $\{\mathbf{x}_k\}_{k\geq 0}$ usando la iteración

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|}. \tag{4.2.4}$$

Esta construcción garantiza un cambio negativo en la función objetivo que es dado por $f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \simeq -\eta\|\nabla f(\mathbf{x}_k)\| < 0$.

Notemos que el segmento $[\mathbf{x}_k\mathbf{x}_{k+1}]$ es perpendicular a la hipersuperficie $\mathcal{S}_{f(\mathbf{x}_k)}$. Luego, obtenemos la poligonal $\mathcal{B}_m = [\mathbf{x}_0\mathbf{x}_1\cdots\mathbf{x}_m]$ de la Sección 4.2.1, que es una aproximación de la curva $\gamma$ dada en el Teorema 4.2.2.

Sin embargo esta construcción tiene un inconveniente, que arreglaremos en breve. Como $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| = \eta > 0$, la sucesión aproximante $\{\mathbf{x}_k\}_{k\geq 0}$ no converge, luego puede fácilmente perder el punto mínimo $\mathbf{x}^*$. Para superar este problema, vamos a utilizar una tasa de aprendizaje $\eta$ variable en el sentido que se vuelva más pequeña cuando la función cambie más lentamente (cuando el gradiente sea pequeño). Asumiremos que la tasa de aprendizaje es proporcional al tamaño del gradiente de $f$ con una constante de proporcionalidad $\delta > 0$, es decir $\eta_k = \delta\|\nabla f(\mathbf{x}_k)\|$. Entonces la iteración (4.2.4) resulta

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \delta\nabla f(\mathbf{x}_k). \tag{4.2.5}$$

PROPOSICIÓN 4.2.4. *La sucesión $\{\mathbf{x}_k\}_{k\geq 0}$ definida en* (4.2.5) *resulta convergente si y sólo si la sucesión de gradientes tiende a 0, $\nabla f(\mathbf{x}_k) \to 0$ cuando $k \to \infty$.*

DEMOSTRACIÓN. Asumamos primero que $\{\mathbf{x}_k\}_{k\geq 0}$ es convergente. De (4.2.5) tenemos que

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\| = \delta\|\nabla f(\mathbf{x}_k)\|.$$

Como la sucesión es convergente, el lado izquierdo tiende a 0, lo que prueba lo deseado.

Asumamos ahora que $\nabla f(\mathbf{x}_k) \to 0$. Para ver que $\{\mathbf{x}_k\}_{k\geq 0}$ es convergente debemos ver que es de Cauchy. Para eso tenemos que demostrar que para todo $p > 0$,

$$\|\mathbf{x}_{k+p} - \mathbf{x}_k\| \to 0, \quad \text{si } k \to \infty.$$

Pero

$$\|\mathbf{x}_{k+p} - \mathbf{x}_k\| \leq \sum_{j=0}^{p-1} \|\mathbf{x}_{k+j+1} - \mathbf{x}_{k+j}\| = \delta \sum_{j=0}^{p-1} \|\nabla f(\mathbf{x}_{k+j})\| \to 0, \quad \text{si } k \to \infty.$$

Esto concluye la demostración. □

Observemos que si $f$ es de clase $C^1$ y estamos en las hipótesis de la Proposición 4.2.4, entonces $\mathbf{x}_k \to \mathbf{x}^*$ y $\nabla f(\mathbf{x}_k) \to \nabla f(\mathbf{x}^*) = 0$.

EJEMPLO 4.2.5. Consideremos la función $f\colon (0,1)\times(-2,2) \to \mathbb{R}$ dada por $f(x,y) = \frac{1}{2}y^2 - x$. Su gráfico se ve en la figura 4.2.3 y tiene un mínimo en el punto $(1,0)$. Sea $(x_0, y_0)$ un punto del dominio de la función. El gradiente de $f$ viene dado por $\nabla f(x,y) = (-1, y)$, y la ecuación 4.2.5 se traduce en

$$\begin{cases} x_{k+1} = x_k + \delta, \\ y_{k+1} = (1-\delta)y_k. \end{cases}$$

Esta iteración se puede resolver explícitamente como

$$\begin{cases} x_k = k\delta + x_0, \\ y_k = (1-\delta)^k y_0. \end{cases}$$

La sucesión $\{y_k\}_{k\geq 0}$ converge si $|1-\delta| < 1$, es decir si $0 < \delta < 2$. Hay dos casos que llevan a comportamientos bien distintos en la iteración:

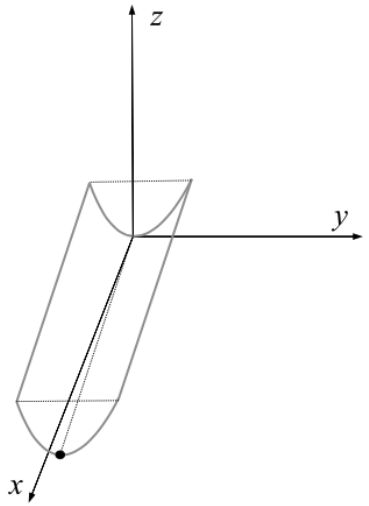
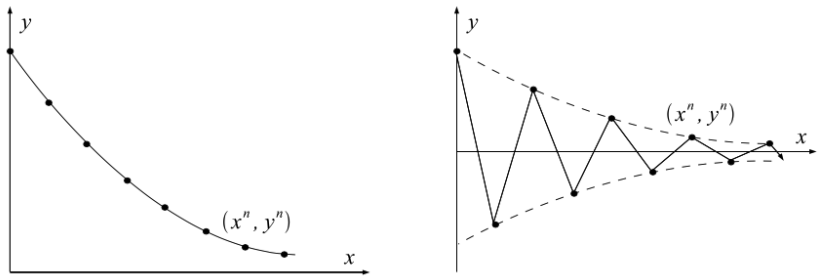



FIGURA 4.3. El gráfico de $z = \frac{1}{2}y^2 - x$.




**a** **b**

FIGURA 4.4. Iteraciones: **a**. Caso $0 < \delta < 1$. **b**. Caso $1 < \delta < 2$.

1. Si $0 < \delta < 1$, la sucesión $y_k$ converge a 0 de manera monótona con signo constante (el signo de $y_0$. La sucesión $x_k$ es una progresión aritmética con paso igual a la tasa de aprendizaje $\delta$. La iteración se detiene cuando $x_k$ sale del dominio (i.e. $x_k > 1$). El valor de salida es $m = \lfloor \frac{1-x_0}{\delta} \rfloor$. Ver Figura 4.2.3 **a.**
2. Si $1 < \delta < 2$, la sucesión $y_k$ converge a 0 de forma oscilatoria. Esto corresponde a la situación en la que la iteración asciende las paredes del cañón sobrepasando el fondo del cañón. Ver Figura 4.2.3 **b**.

**4.2.4. Método del Hessiano.** El método de descenso de gradiente no toma en consideración la curvatura de la superficie $z = f(\mathbf{x})$. que está descripta por la matriz Hessiana $Hf(\mathbf{x})$. Necesitaremos algunas propiedades de esta matriz.

PROPOSICIÓN 4.2.6. *Sea $f\colon D \subset \mathbb{R}^n \to \mathbb{R}$ una función de clase $C^2$. Valen las siguientes propiedades:*

(a) *La matriz Hessiana es simétrica;*
(b) *Para todo $\mathbf{x} \in D$, los autovalores de $Hf(\mathbf{x})$ son reales;*

(c) *Para todo* $\mathbf{x} \in D$*, la matriz Hessiana* $Hf(\mathbf{x})$ *es diagonalizable y sus autovalores* $\{\mathbf{u}_i\}_{1\leq i\leq n}$ *forman una base ortonormal de* $\mathbb{R}^n$*.*

(d) *Dado* $\mathbf{v} \in \mathbb{R}^n$*, si definimos* $v_i = v \cdot \mathbf{u}_i$*, se tiene que*

$$\mathbf{v}Hf(\mathbf{x})\mathbf{v}^t = \sum_{i=1}^{n} v_i\lambda_i,$$

*donde* $\{\lambda_i\}_{1\leq i\leq n}$ *son los autovalores de* $Hf(\mathbf{x})$ *contados según su multiplicidad.*

(e) *Si llamamos* $\lambda_{min}$ *y* $\lambda_{max}$ *al menor y mayor autovalor de* $Hf(\mathbf{x})$ *respectivamente, entonces*

$$\lambda_{min}\|\mathbf{v}\|^2 \leq \mathbf{v}Hf(\mathbf{x})\mathbf{v}^t \leq \lambda_{max}\|\mathbf{v}\|^2.$$

Demostración. Para ver (a), sólo hay que observar que como $f \in C^2$, entonces

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) = \frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}).$$

Las propiedades (b)–(e) son propiedades conocidas de matrices simétricas. □

La fórmula (4.2.5) provee la recurrencia para el método de descenso de gradiente. Observamos que la aproximación lineal

$$\begin{aligned} f(\mathbf{x}_{k+1}) &\simeq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k) \cdot (\mathbf{x}_{k+1} - \mathbf{x}_k) \\ &= f(\mathbf{x}_k) - \delta\|\nabla f(\mathbf{x}_k)\|^2 < f(\mathbf{x}_k), \end{aligned}$$

produce un valor de $f$ menor que el valor previo. Pero esto es sólo una aproximación que se verifica al eliminar la corrección cuadrática y funciona para valores pequeños de $\delta$. Si tomamos en consideración los términos cuadráticos, entonces vemos que la desigualdad puede no ser más cierta. Para estudiar este caso, hacemos una aproximación de segundo orden

$$\begin{aligned} f(\mathbf{x}_{k+1}) &\simeq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k) \cdot (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x}_{k+1} - \mathbf{x}_k)Hf(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k)^t \\ &= f(\mathbf{x}_k) - \delta\|\nabla f(\mathbf{x}_k)\|^2 + \frac{\delta^2}{2}\nabla f(\mathbf{x}_k)Hf(\mathbf{x}_k)\nabla f(\mathbf{x}_k)^t. \end{aligned}$$

El último término es el término de corrección debido a la curvatura de $f$. Hay dos casos de importancia que vale la pena resaltar:

1. El Hessiano $Hf$ es definido negativo. En este caso el término de corrección es negativo, luego la aproximación de Taylor implica

   $$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k) - \delta\|\nabla f(\mathbf{x}_k)\|^2 < f(\mathbf{x}_k),$$

   es decir que para cualquier tasa de aprendizaje $\delta > 0$ cada iteración provee un valor de $f$ más pequeño.
2. El Hessiano $Hf$ es definido positivo. En este caso el término de corrección es positivo. La Proposición 4.2.6, parte (e), nos da una cota de error para el término de corrección:

   $$0 \leq \frac{\delta^2}{2}\lambda_{\min}\|\nabla f(\mathbf{x}_k)\|^2 \leq \frac{\delta^2}{2}\nabla f(\mathbf{x}_k)Hf(\mathbf{x}_k)\nabla f(\mathbf{x}_k)^t \leq \frac{\delta^2}{2}\lambda_{\max}\|\nabla f(\mathbf{x}_k)\|^2.$$

Luego al comparar los valores de $f$ en iteraciones sucesivas, obtenemos

$$f(\mathbf{x}_{k+1}) \simeq f(\mathbf{x}_k) - \delta\|\nabla f(\mathbf{x}_k)\|^2 + \frac{\delta^2}{2}\nabla f(\mathbf{x}_k) H f(\mathbf{x}_k) \nabla f(\mathbf{x}_k)^t$$
$$\leq f(\mathbf{x}_k) - \delta\|\nabla f(\mathbf{x}_k)\|^2 + \frac{\delta^2}{2}\lambda_{\max}\|\nabla f(\mathbf{x}_k)\|^2,$$

y este último término es menor que $f(\mathbf{x}_k)$ si y sólo si

$$-\delta + \frac{\delta^2}{2}\lambda_{\max} < 0 \iff \delta < \frac{2}{\lambda_{\max}}. \tag{4.2.6}$$

Luego, si la tasa de aprendizaje verifica (4.2.6), entonces la aproximación cuadrática nos da un valor para $f(\mathbf{x}_{k+1})$ menor que $f(\mathbf{x}_k)$.

## 4.3. Búsqueda estocástica

El método de descenso de gradiente no provee resultados satisfactorios en todos los casos. Esto puede ser corregido con ciertas variantes del método. Pero, algunas veces es necesario modificar el método para evitar caer en regiones *plateau* (donde el gradiente es muy pequeño sin que en esa región se encuentre el mínimo buscado) y el método de descenso se vuelve muy ineficiente. En esta sección presentamos un método que tiene una variante estocástica que soluciona este inconveniente. El mínimo es buscado a lo largo de un procesos de difusión aleatoria en lugar de seguir un camino determinístico.

**4.3.1. Modelo determinístico.** Para entender la idea que vamos a utilizar, analicemos primero el modelo determinístico de descenso de gradiente desde otra perspectiva.

Supongamos que elegimos un punto de partida $\mathbf{x}_0$ y queremos encontrar el camino más rápido para llegar a $\mathbf{x}^*$, el mínimo de nuestra función $f$. La idea entonces es buscar ese camino $\mathbf{x}(t)$ como una curva integral de un cierto campo vectorial $\mathbf{b}(\mathbf{x})$, i.e.

$$\mathbf{x}'(t) = \mathbf{b}(\mathbf{x}(t)), \qquad \mathbf{x}(0) = \mathbf{x}_0.$$

El objetivo entonces es el de encontrar al campo $\mathbf{b}(\mathbf{x})$ de manera tal que la función $\varphi(t) := f(\mathbf{x}(t))$ decaiga lo más rápidamente posible. Luego, calculamos la derivada de $\varphi(t)$:

$$\varphi'(t) = \frac{d}{dt} f(\mathbf{x}(t)) = \nabla f(\mathbf{x}(t)) \cdot \mathbf{x}'(t) = \nabla f(\mathbf{x}(t)) \cdot \mathbf{b}(\mathbf{x}(t)).$$

Esta derivada se hace mínima si $\mathbf{b}(\mathbf{x}) = -\lambda \nabla f(\mathbf{x})$ para $\lambda > 0$. Es decir si usamos el método de descenso de gradiente.

**4.3.2. Variante estocástica.** Observemos que en el modelo determinista (descenso de gradiente) si se ingresa en una región en donde el gradiente de $f$ es pequeño, la tasa de cambio de la trayectoria $\mathbf{x}(t)$ es pequeña, entonces el tiempo que tarda en salir de esa región (asumiendo que el mínimo aún se encuentra lejos) puede ser muy grande. Una manera de solucionar este problema es introducir *ruido* o un *componente aleatorio*. La idea va a ser que este ruido sea chico en regiones donde el gradiente de $f$ sea grande pero que el ruido aumente si se ingresa a una región de plateau.

Para poder avanzar en la introducción del ruido, veamos primero unos conceptos probabilísticos.

*4.3.2.1. Movimiento Browniano y ruido blanco.* Vamos a dar una muy breve definición e introducción a este tema. Recomendamos una lectura más detallada del mismo y pueden hacerlo, por ejemplo, en **[9, 10]** o en las notas **[12]**.

DEFINICIÓN 4.3.1. Un movimiento Browniano, o proceso de Wiener, es un proceso estocástico $W_t$, $t \geq 0$, tal que

1. $W_0 = 0$ casi seguramente;
2. si $0 \leq u < s < t$, entonces $W_t - W_s$ y $W_s - W_u$ son independientes (el proceso tiene incrementos independientes);
3. $t \mapsto W_t$ es continuo casi seguramente;
4. los incrementos están normalmente distribuidos, con $W_t - W_s \sim N(0, |t-s|)$.

Observemos que por 1 y 4 tenemos que $\mathbb{E}[W_t] = 0$ y $\mathbb{E}[W_t^2] = t$.

Un movimiento Browniano $m-$dimensional es un vector $W_t = (W_t^1, \ldots, W_t^m)$ en donde cada componente $W_t^k$ es un movimiento Browniano dado por la Definición 4.3.1 y son independientes.

Sea $\mathbf{b}\colon \mathbb{R}^n \to \mathbb{R}^n$ un campo vectorial de clase $C^1$ y sea $\sigma\colon \mathbb{R}^n \to \mathbb{R}^{n\times m}$ una matriz de clase $C^1$.

Se define entonces el proceso estocástico $X_t \in \mathbb{R}^n$ como la solución de la ecuación diferencial estocástica

$$dX_t = \mathbf{b}(X_t)\, dt + \sigma(X_t)\, dW_t. \tag{4.3.1}$$

Al campo $\mathbf{b}(\mathbf{x})$ se lo llama la *deriva* (en inglés *drift*) y a la matriz $\sigma(\mathbf{x})$ se la llama *difusión* (en inglés *diffusion*).

Para entender lo que significa una solución de (4.3.1), es conveniente mirar la ecuación a tiempo discreto. Fijemos un $\delta > 0$ y tomemos $t_k = k\delta$. Notemos además $X_k = X_{t_k}$. Entonces (4.3.1) se transforma en

$$X_{k+1} - X_k = \mathbf{b}(X_k)\delta + \sigma(X_k)(W_{t_{k+1}} - W_{t_k}).$$

Si usamos la condición 4 de la Definición 4.3.1, entonces $B_k := W_{t_{k+1}} - W_{t_k} \sim N(0, \delta)$, luego llegamos a

$$X_{k+1} = X_k + \mathbf{b}(X_k)\delta + \sigma(X_k)B_k. \tag{4.3.2}$$

(Este esquema se llama el *método de Euler-Maruyama* para la aproximación de ecuaciones diferenciales estocásticas)

Comparemos (4.3.2) con el método de descenso de gradiente (4.2.5). Observemos que si tomamos $\mathbf{b}(\mathbf{x}) = -\nabla f(\mathbf{x})$ lo que tenemos es una perturbación aleatoria del descenso de gradiente.

Razonemos ahora como en la sección previa. Lo que queremos es buscar entonces el campo $\mathbf{b}(\mathbf{x})$ y la matriz de difusión $\sigma(\mathbf{x})$ de manera tal que $f(X_t)$ decaiga lo más rápidamente posible. Al ser un objeto aleatorio, lo que se busca es que decaiga *en promedio* lo más rápidamente posible. Luego, la cantidad a analizar es

$$\varphi(t) = \mathbb{E}[f(X_t)|X_0 = \mathbf{x}_0].$$

Para analizar la variación de $\varphi(t)$, se usa la *fórmula de Dynkin* (ver **[9]**)

$$\varphi(t+\delta) = \varphi(t) + \delta\mathbb{E}[\mathcal{A}f(X_t)|X_0 = \mathbf{x}_0] + o(\delta^2), \tag{4.3.3}$$

donde

$$\mathcal{A}f = \sum_{i,j=1}^{n} a_{ij}\frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{k=1}^{n} b_k \frac{\partial f}{\partial x_k} = \operatorname{tr}(AHf) + \mathbf{b}\cdot\nabla f.$$

con

$$A = (a_{ij})_{1\le i,j\le n},\ A = \frac{1}{2}\sigma\sigma^t \qquad \text{y} \qquad \mathbf{b} = (b_1, \dots, b_n).$$

Ahora, cuando la curva se encuentra en el plateau, la función objetivo es plana, luego tiene un Hessiano pequeño. Para poder salir del plateau entonces ahí vamos a definir una matriz de difusión grande. Consecuentemente, elegimos la matriz de difusión $\sigma$ de manera tal que $\sigma\sigma^t$ sea proporcional a la inversa del Hessiano, i.e.

$$\sigma\sigma^t = \lambda(Hf)^{-1}. \tag{4.3.4}$$

Con esta elección, obtenemos

$$\mathcal{A}f(\mathbf{x}) = \frac{n}{2}\lambda + \nabla f(\mathbf{x})\cdot\mathbf{b}(\mathbf{x}),$$

y ahora, para minimizar $\mathcal{A}f(\mathbf{x})$ elegimos $\mathbf{b}(\mathbf{x}) = -\eta\nabla f(\mathbf{x})$, de donde obtenemos

$$\mathcal{A}f(\mathbf{x}) = \frac{n}{2}\lambda - \eta\|\nabla f(\mathbf{x})\|^2.$$

Si queremos que $\mathcal{A}f(\mathbf{x}) < 0$ necesitamos que las tasas de aprendizaje $\lambda$ y $\eta$ verifiquen la relación

$$\frac{\lambda}{\eta} < \frac{2}{n}\|\nabla f(\mathbf{x})\|^2.$$

Para poder entender mejor este método, analicemos un ejemplo sencillo,

Ejemplo 4.3.2. Consideremos el caso unidimensional dado por la función objetivo $f(x) = \frac{1}{2}x^2$ para $x \in (a, b)$ con $a > 0$. Luego la ecuación estocástica queda $dX_t = b(X_t)\,dt + \sigma(X_t)\,dW_t$, $X_0 = x_0 \in (a, b)$.

El Hessiano en este caso coincide con la derivada segunda $Hf(x) = f''(x) = 1$, luego debemos elegir

$$\sigma\sigma^t = \sigma^2 = \lambda\frac{1}{f''(x)} = \lambda \quad \Rightarrow \quad \sigma = \sqrt{\lambda}.$$

Por otro lado,

$$b(x) = -\eta f'(x) = -\eta x.$$

Ahora las tasas de aprendizaje $\lambda$ y $\eta$ deben verificar

$$\frac{\lambda}{\eta} < 2|f'(x)|^2 = 2x^2$$

como $x \in (a, b)$ podemos elegir $\lambda = 2a^2\eta$. La ecuación estocástica queda

$$dX_t = -\eta X_t\,dt + \sqrt{\lambda}\,dW_t, \qquad X_0 = x_0.$$

Esta ecuación se llama *ecuación de Langevin*. La solución a esta ecuación viene dada por el *proceso de Orstein-Uhlenbeck*

$$X_t = x_0 e^{-\eta t} + \sqrt{\lambda}\int_0^t e^{-\eta(t-s)}\,dW_s,$$

que es la suma de una función determinística y una integral estocástica. Ahora observemos que

$$\mathbb{E}\left[\int_0^t h(s)\,dW_s\right] = 0 \qquad \text{y} \qquad \mathbb{E}\left[\left(\int_0^t h(s)\,dW_s\right)^2\right] = \int_0^t h^2(s)\,ds. \tag{4.3.5}$$

Estas igualdades se ven fácilmente recordando que

$$\int_0^t h(s)\,dW_s \simeq \sum_{k=1}^{N} h(s_k)\,\underbrace{(W_{s_k+\delta} - W_{s_k})}_{B_k},$$

donde $N\delta = t$ y observando que $\mathbb{E}[B_k] = 0$, $\mathbb{E}[B_k^2] = \delta$ y $B_k, B_j$ son independientes si $k \neq j$. Ver las propiedades de la Definición 4.3.1.

Usando (4.3.5), se llega a

$$\mathbb{E}[X_t] = x_0 e^{-\eta t} \qquad \text{y} \qquad \mathbb{V}[X_t] = \frac{\lambda}{2\eta}(1 - e^{-2\eta t}).$$

La media representa la dirección esperada del movimiento, mientras que la integral es la parte del ruido blanco.

La altura esperada $\varphi(t)$ se puede calcular explícitamente en este caso

$$\begin{aligned}
\varphi(t) &= \mathbb{E}[f(X_t)|X_0 = x_0] = \frac{1}{2}\mathbb{E}[(X_t)^2|X_0 = x_0] \\
&= \frac{1}{2}\mathbb{E}\left[x_0^2 e^{-2\eta t} + 2x_0\sqrt{\lambda}e^{-\eta t}\int_0^t e^{-\eta(t-s)}\,dW_s + \lambda\left(\int_0^t e^{-\eta(t-s)}\,dW_s\right)^2\right] \\
&= \frac{1}{2}\left(x_0^2 e^{-2\eta t} + \frac{\lambda}{2\eta}(1 - e^{-2\eta t})\right) = \frac{1}{2}\left(\frac{\lambda}{2\eta} + \left(x_0^2 - \frac{\lambda}{2\eta}\right)e^{-2\eta t}\right).
\end{aligned}$$

Como $\frac{\lambda}{2\eta} = a^2 < x_0^2$, sigue que $\varphi(t)$ es decreciente, luego el mínimo será

$$\lim_{t\to\infty} \varphi(t) = \frac{1}{2}\frac{\lambda}{2\eta} = \frac{1}{2}a^2.$$

Observemos que este es el resultado esperado que, para este caso particular, lo podíamos obtener con una simple inspección de la función.

## 4.4. Resumen

Encontrar el mínimo global de una función de costo dada, es un ingrediente importante en cualquier algoritmo de aprendizaje automático. El proceso de aprendizaje es asociado con el proceso de ajustar las variable de la función de costo en sus valores optimales. Algunos métodos son de primer orden (es decir que involucran sólo las derivadas primeras), como por ejemplo el método de descenso de gradiente lo que los hace eficientes computacionalmente. Algunos otros métodos son de segundo orden e involucran las derivadas segundas, como por ejemplo el método del Hessiano.

El más conocido y utilizado de los métodos para minimización es el de descenso de gradiente. Este método es fácil de implementar, pero presenta dificultades cuando la función de costo no es convexa. Este método se puede mejorar de varias maneras para evitar caer en mínimos locales o en zonas de plateau. En estas notas sólo discutimos

unas pocas variantes, pero pueden consultar en **[5]** por más variantes como AdaGrad, Adam, AdaMax, RMSProp, por ejemplo.

La búsqueda estocástica propone un algoritmo para salir de las zonas de plateau. Esto involucra la búsqueda de mínimos a lo largo de procesos estocásticos en lugar de curvas determinísticas. Este método, sin embargo, resulta computacionalmente costoso dado que involucra el cálculo de la inversa de la matriz Hessiana de la función de costo.

## 4.5. Ejercicios

EJERCICIO 4.5.1. Sea $f(x, y) = e^x \sin y$, con $(x, y) \in (0, 1) \times (0, \frac{\pi}{2})$.

(a) Mostrar que $f$ es armónica (i.e. $f_{xx} + f_{yy} = 0$);
(b) Hallar $\|\nabla f\|$;
(c) Mostrar que la ecuación $\nabla f = 0$ no posee soluciones;
(d) Hallar los máximos y mínimos de $f$.

EJERCICIO 4.5.2. Considerar la forma cuadrática $Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}A\mathbf{x}^t - \mathbf{b} \cdot \mathbf{x}$, donde $A \in \mathbb{R}^{n\times n}$ es una matriz no singular.

(a) Hallar el gradiente $\nabla Q$;
(b) Escribir la iteración de descenso de gradiente;
(c) Hallar el Hessiano $HQ$;
(d) Escribir la iteración del método del Hessiano.

Capítulo 5

# Neuronas abstractas

El concepto de *neuronas abstractas* es el componente fundamental de cualquier red neuronal. Es una unidad que imita el comportamiento de una neurona biológica y consiste de una entrada (señal de entrada), pesos (pesos sinápticos) y una función de activación (modelo de disparo neuronal). Este capítulo introduce los tipos más comunes de neuronas (perceptrones, neuronas sigmoides, etc.) e investiga sus propiedades.

## 5.1. Definición y propiedades

En vista de los ejemplos presentados en el Capítulo 1, consideremos la siguiente definición que formaliza la intuición desarrollada:

DEFINICIÓN 5.1.1. Una neurona abstracta es un cuádruple $(\mathbf{x}, \mathbf{w}, \varphi, y)$, donde $\mathbf{x} = (x_0, \ldots, x_n)$ es el vector de entrada, $\mathbf{w} = (w_0, \ldots, w_n)$ es el vector de pesos, con $x_0 = -1$ y $w_0 = b$, el sesgo, y $\varphi$ es la función de activación que define la función de salida $y = \varphi(\mathbf{w} \cdot \mathbf{x}) = \varphi(\sum_{i=0}^{n} w_i x_i)$.

La forma en que una neurona abstracta aprende un objetivo deseado $\mathbf{z}$ es ajustando los pesos $\mathbf{w}$ de manera tal que una cierta función que mide el error entre las variables deseadas $\mathbf{z}$ y las salidas $y$ sea minimizada. Algunas posibles funciones de error las hemos visto en el Capítulo 3 y los algoritmos de minimización los tratamos en el Capítulo 4.

La neurona abstracta está representada en la Figura 5.1. La unidad de cómputo está dividida en dos partes: la primera contiene el símbolo de sumación, $\sum$, que indica la suma de las entradas con los pesos respectivos, y la segunda contiene a la función de activación $\varphi$ usada para definir la salida $y$.

Notemos que el sesgo $b$ está incluido en el sistema de pesos $w_0 = b$ con entrada constante $x_0 = -1$. Equivalentemente se puede considerar un sesgo $w_0 = -b$ con

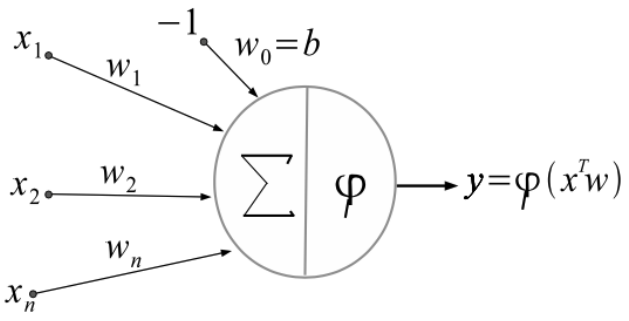



FIGURA 5.1. Una neurona abstracta con sesgo $b$ y función de activación $\varphi$.

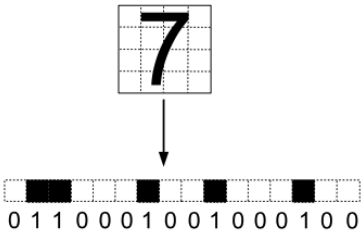



FIGURA 5.2. La transformación de $4 \times 4$ píxeles en una sucesión de 0s y 1s.

entrada correspondiente $x_0 = 1$. En cualquiera de estos casos, la expresión del producto interno es dado por

$$\mathbf{w} \cdot \mathbf{x} = w_1 x_1 + \cdots w_n x_n - b.$$

A los vectores $\mathbf{w} = (w_0, \ldots, w_n)$ y $\mathbf{x} = (x_0, \ldots, x_n)$ se los llaman los vectores de *pesos extendidos* y de *entradas extendidas* respectivamente. La función de salida $y = \varphi(\mathbf{w} \cdot \mathbf{x})$ se la llama también la *función primitiva* de la unidad de cómputo.

Las entradas $\{x_i\}_{i=1,\ldots,n}$ son comunicadas a través de ejes a la unidad de cómputo luego de ser previamente multiplicadas por los pesos, ver Figura 5.1. Los datos de entrada pueden ser de diversos tipos, como por ejemplo:

- *binario* si $x_i \in \{0, 1\}$ para $i = 1, \ldots, n$;
- *signo* si $x_i \in \{-1, 1\}$ para $i = 1, \ldots, n$;
- *dígito* si $x_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ para $i = 1, \ldots, n$;
- *números reales arbitrarios* si $x_i \in \mathbb{R}$ para $i = 1, \ldots, n$;
- *números reales acotados* si $x_i \in [a, b]$ para $i = 1, \ldots, n$.

Por ejemplo, todo dígito escrito a mano puede ser transformado en un dato digital, ver Figura 5.1. La matriz de $4 \times 4$ se lee línea a línea y se ubica como una sucesión de 0s y 1s de longitud 16. Se pone un valor de 1 si el pixel asignado es activado en más del un 50 %. De otra forma se asigna el valor 0. Codificar caracteres de esta manera es muy simplista y se pierde información sobre formas locales. Sin embargo hay mejores maneras de codificar figuras usando *convoluciones*. En general, en el caso de imágenes en escala en grises, cada píxel tiene una activación que es un número entre 0 y 1 (1 para negro y 0 para blanco).

**Eficiencia de la entrada.** Asumamos que uno quiere implementar un circuito de neuronas. Una pregunta que debe ser hecha es: ¿qué tipo de datos es más eficiente? Equivalentemente, ¿cuántos cambios de estado optimizan la información transmitida desde el punto de vista del costo de circuito de implementación?

Para responder a esta pregunta, sea $\beta$ el número de estados de la señal de entrada, por ejemplo $\beta = 2$ para señales binarias. El número de canales o entradas viene dado por $n$. El costo de implementación, $F$, se asume proporcional tanto con $n$ como con $\beta$, i.e. $F = cn\beta$ donde $c > 0$ es la constante de proporcionalidad. Usando $n$ canales con $\beta$ estados, nos da un total de $\beta^n$ entradas posibles. Asumamos ahora que el costo $F$ es fijo y tratemos de optimizar la cantidad de números transmitidos como función de $\beta$. Consideremos la constante $k = \frac{F}{c}$, entonces $n = \frac{F}{c\beta} = \frac{k}{\beta}$. Entonces el número de entradas totales representadas es $f(\beta) = \beta^n = \beta^{\frac{k}{\beta}}$. Para calcular el máximo de esta

función, tomamos el logaritmo (que es una función monótona) $g(\beta) = \ln f(\beta) = \frac{k}{\beta} \ln \beta$ y su derivada es $g'(\beta) = \frac{k}{\beta^2}(1 - \ln \beta)$. En consecuencia, el máximo se alcanza para $\beta = e \simeq 2{,}718$. El entero más próximo a este número es 3, por ende $\beta = 3$ es el número óptimo de estados para las señales de entrada desde el punto de vista del costo.

Sin embargo, este no es el único punto de vista y nos centraremos en una teoría de modelos de neuronas con cualquier número de estados para las señales de entrada.

En lo que sigue presentaremos algunos tipos clásicos de neuronas especificando sus tipos de entradas y funciones de activación.

## 5.2. El modelo del perceptrón

Un *perceptrón* es una neurona cuya entrada es cero o uno, i.e. $x_i \in \{0, 1\}$ , y cuya función de activación viene dada por la función de Heaviside

$$\varphi(x) = \begin{cases} 1, & \text{si } x < 0 \\ 0, & \text{si } x \geq 0. \end{cases}$$

La salida del perceptrón es una *puerta umbral*

$$y = \varphi(\mathbf{w} \cdot \mathbf{x}) = \begin{cases} 1, & \text{si } \sum_{i=1}^{n} w_i x_i < b \\ 0, & \text{si } \sum_{i=1}^{n} w_i x_i \geq b. \end{cases}$$

De esta manera, un perceptrón es una regla que toma decisiones pesando la evidencia dada por las entradas $\{x_i\}_{i=1,\ldots,n}$. El umbral $b$ es una medida de qué tan fácil es para el perceptron decidir que la salida es 1. Para todo propósito general, esto es visto como otro peso, denotado por $w_0$ llamado el sesgo.

Veamos la interpretación geométrica del perceptrón. Consideremos un hiperplano $(n-1)-$dimensional en $\mathbb{R}^n$ definido por

$$\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^n \colon \mathbf{w} \cdot \mathbf{x} = b\},$$

donde $\mathbf{x} = (x_1, \ldots, x_n)$ y $\mathbf{w} = (w_1, \ldots, w_n)$. El vector normal $\mathbf{n}$ es el vector de pesos $\mathbf{n} = \mathbf{w}$. El hiperplano para por un punto $\mathbf{p}$ si se verifica la relación $b = \mathbf{w} \cdot \mathbf{p}$. Luego el resultado $y$ del perceptrón es asociar el valor 0 a uno de los dos semiespacios determinados por el hiperplano $\mathcal{H}$ y 1 al resto. Ver Figura 5.2. En resumen, dado un borde lineal entre dos regiones, un perceptrón puede determinar a cual región pertenece un punto dado.

El perceptrón puede implementar las puertas lógicas AND ("$\wedge$") y OR ("$\vee$"). Esta propiedad hace importancia al perceptrón para cálculos lógicos.

**Implementando AND.** Consideremos la operación Booleana definida por la tabla

| $x_1$ | $x_2$ | $y = x_1 \wedge x_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

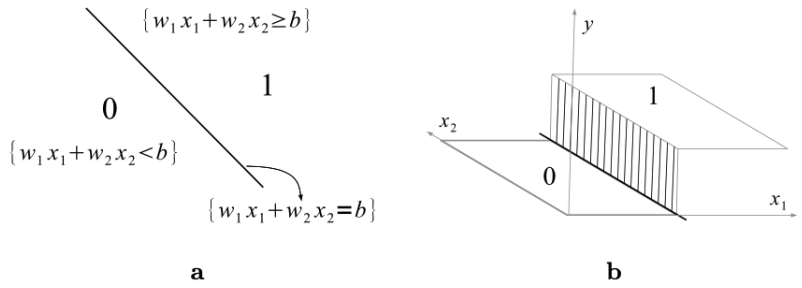



FIGURA 5.3. **a.** La separación del plano en semiespacios. **b.** El gráfico de la función de activación para un perceptrón con dos entradas, $x_1$ y $x_2$.

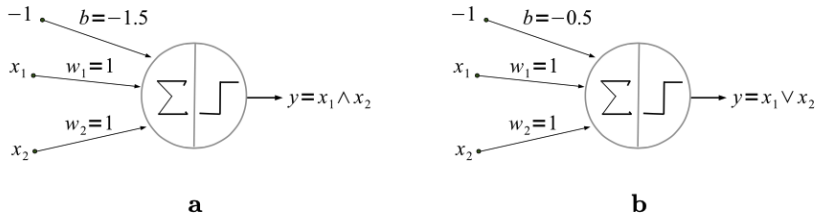



FIGURA 5.4. Implementación de una función Booleana usando un perceptrón: **a.** La función AND. **b.** La función OR.

La misma función de salida $y = x_1 \wedge x_2$ puede ser generada por un perceptrón con dos entradas $x_1, x_2 \in \{0, 1\}$, pesos $w_1, w_2 = 1$, sesgo $b = 1{,}5$ y $x_0 = -1$, ver Figura 5.2 **a**. La salida se escribe como

$$y = \varphi(x_1 + x_2 - 1{,}5) = \begin{cases} 0, & \text{si } x_1 + x_2 < 1{,}5 \\ 1, & \text{si } x_1 + x_2 \geq 1{,}5 \end{cases}$$

La condición $\{x_1 + x_2 \geq 1{,}5\}$ sólo se satisface si $x_1 = x_2 = 1$, en todos los otros casos se verifica $\{x_1 + x_2 < 1{,}5\}$.

En vista de la interpretación geométrica del perceptrón, la linea $x_1 + x_2 = 1{,}5$ divide los datos de entrada $\{(0, 0); (0, 1); (1, 0); (1, 1)\}$ en dos clases asociando 1 al semiespacio $\{x_1 + x_2 \geq 1{,}5\}$ y 0 al semiespacio $\{x_1 + x_2 < 1{,}5\}$. Ver Figura 5.2 **a**.

**Implementando OR.** Esta función Booleana está definida por la tabla:

| $x_1$ | $x_2$ | $y = x_1 \vee x_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

La función de salida $y = x_1 \vee x_2$ es generada por un perceptrón con dos entradas $x_1, x_2 \in \{0, 1\}$, pesos $w_1, w_2 = 1$, sesgo $b = -0{,}5$ y $x_0 = -1$, ver Figura 5.2 **b**. La salida

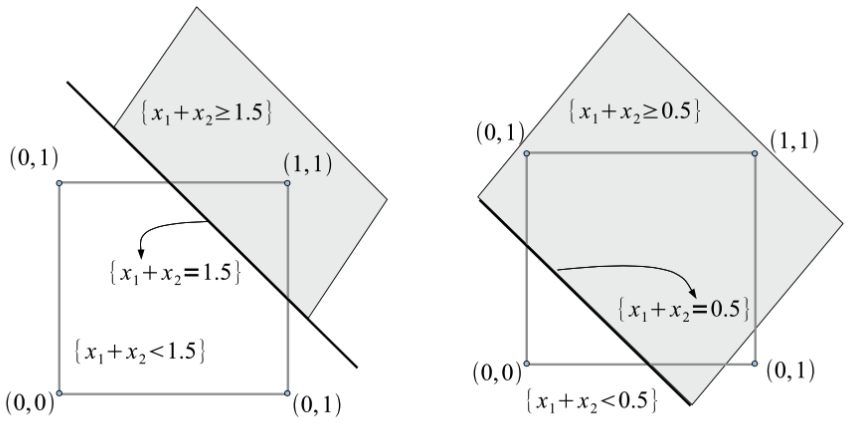



FIGURA 5.5. Partición usando un perceptrón; el semiespacio sombreado es asociado con el valor de salida 1.: **a.** La función AND. **b.** La función OR.

se escribe como

$$y = \varphi(x_1 + x_2 - 1{,}5) = \begin{cases} 0, & \text{si } x_1 + x_2 < 0{,}5 \\ 1, & \text{si } x_1 + x_2 \geq 0{,}5. \end{cases}$$

La conclusión sigue de la observación de que la condición $\{x_1 + x_2 < 0{,}5\}$ se satisface si y sólo si $x_1 = x_2 = 0$. Ver la Figura 5.2 **b** para la interpretación geométrica de este hecho.

Sin embargo, un perceptrón no puede implementar todas las funciones lógicas. Por ejemplo, no puede implementar la función XOR (el OR exclusivo), que viene dado por la tabla

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Este hecho fue observado por primera vez en [**15**]. Veremos luego que la función XOR puede ser aprendida por una red neuronal de dos perceptrones.

La imposibilidad de implementar XOR por un perceptrón sigue del hehco que no hay una línea que separe los símbolos 0 y 1 en la Figura 5.2 **a**. Esto puede demostrarse de varias maneras:

1. Una demostración sigue como consecuencia de las propiedades de separación del plano. Si uno asume por contradicción la existencia de una línea de separación, entonces el plano es dividido por la línea en dos semiplanos, $H_1$ y $H_2$ que son conjuntos convexos (ver Figura 5.2 **b**). Dado que ambos símbolos 1 pertenecen a $H_1$, entonces por convexidad el segmento que los une queda contenido en $H_1$. Análogamente el segmento que une los símbolos 0 es contenido en $H_2$. Como

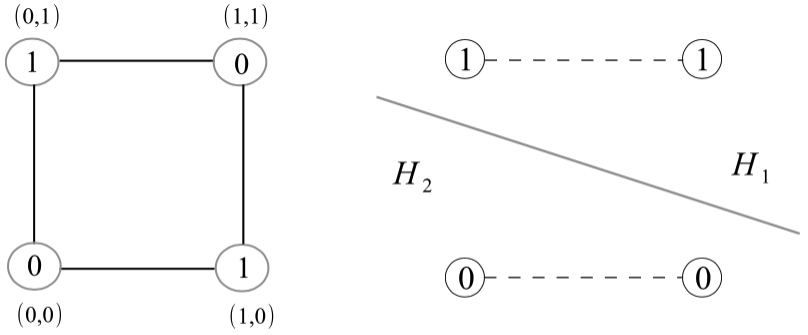



FIGURA 5.6. Clasificación de clústeres usando un perceptrón: **a.** La función XOR. **b.** Una línea divide el plano en dos conjuntos convexos $H_1$ y $H_2$.

$H_1 \cap H_2 = \emptyset$, los segmentos que unen los símbolos 1 y 0 deben ser disjuntos, pero esto es una contradicción dado que se intersectan (ver Figura 5.2 **a**).

2. La segunda prueba se basa en motivos algebraicos. Asumamos que existe una línea de separación de la forma $w_1x_1+w_2x_2 = b$ que separa los puntos $\{(0,0),(1,1)\}$ de los puntos $\{(0,1),(1,0)\}$. Considerando alguna elección, podemos suponer testeando con el primer grupo que $0 < b$ y $w_1 + w_2 < b$ y testeando con el segundo grupo que $w_1 > b$ y $w_2 > b$. La suma de las últimas dos desigualdades, junto con la segunda desigualdad implca que $2b < b$ que contradice que $b > 0$.

**División de clústeres.** Motivados por la aplicación previa de división de datos de entrada en clases, lo extendemos al problema más general de clasificación en dos clústeres. Permitimos ahora que los datos de entrada sean números tomando valores en el intervalo $[0,1]$ y asumimos por simplicidad que $n = 2$. Entonces cada dato es un par de números reales que se representan como un punto en el cuadrado unitario $[0,1] \times [0,1]$. Supongamos ahora que asociamos una etiqueta o característica con dos posibles valores a cada punto, dada por 'grupo 1' (o, color rojo, forma de estrella, etc.) y 'grupo 2' (o, color azul, forma de disco, etc.). La pregunta entonces es: ¿puede un perceptrón decidir si un punto dado perteneces a un grupo o al otro? La respuesta va a depender de la distribución de los datos. Si los datos pueden ser separados por una línea de manera tal que un grupo quede contenido en un semiplano y el otro en el otro semiplano, entonces el perceptrón podrá eventualmente ser capaz de decidir qué grupo está asociado con qué etiqueta luego de un adecuado ajuste de los pesos. Para poder hacer esto, introducimos la función

$$z(x_1,x_2) = \begin{cases} 0, & \text{si } (x_1,x_2) \in \mathcal{G}_1 \\ 1, & \text{si } (x_1,x_2) \in \mathcal{G}_2, \end{cases}$$

donde $\mathcal{G}_1$ y $\mathcal{G}_2$ representan a "grupo 1" y "grupo 2" respectivamente. El perceptrón decide entre estos dos grupos si somos capaces de encontrar dos pesos $w_1, w_2$ y un sesgo $b$ tales que la línea $w_1x_1 + w_2x_2 - b = 0$ separa a los grupos $\mathcal{G}_1$ y $\mathcal{G}_2$. Si esto es

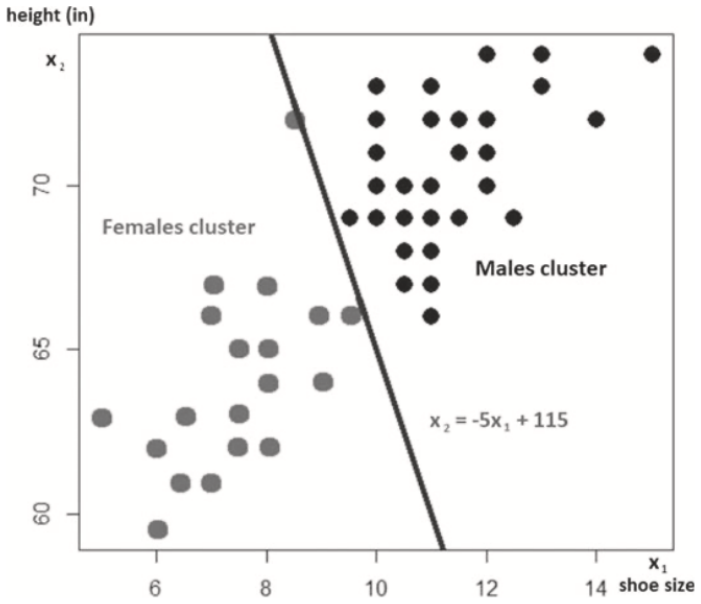



FIGURA 5.7. Clasificación de un perceptrón entre varones y mujeres. La línea $x_2 = -x_1 + 115$ es un borde de decisión en el espacio $\{x_1, x_2\}$ entre dos clústeres separables linealmente.

posible, la función de salida

$$y = \varphi(w_1 x_1 + w_2 x_2 - b)$$

tendrá la misma expresión que la dada para la función $z$. Los detalles para los algoritmos de aprendizaje para el perceptrón los veremos más adelante.

EJEMPLO 5.2.1. Consideremos la entrada $(x_1, x_2)$, donde $x_1$ denota el talle del calzado y $x_2$ la altura (en pulgadas) de algunas personas en la población. Asociamos una etiqueda para cada individuo considerando el mapa $(x_1, x_2) \mapsto \ell \in \{\text{varón}, \text{mujer}\}$. Dado un par $(x_1, x_2)$, un perceptrón es capaz de distinguir si corresponde a un varón o a una mujer si hay una línea de separación $w_1 x_1 + w_2 x_2 = b$ entre los dos grupos.

Se obtienen datos de una población de estudiantes de la Universidad de Michigan graficados en la Figura 5.2.1. Notamos que los grupos de varones y mujeres son separables linealmente con la línea dada por $x_2 = -5x_1 + 115$. Luego, dado un tamaño de calzado $s$ y una altura $h$, el perceptrón lo clasificará como "varón" si $h > -5s + 115$ y como "mujer" en caso contrario.

Este problema se puede formular en términos de neuronas de varias maneras

1. Primero, asociamos un valor numérico a los datos. Etiquetamos los puntos correspondientes a varones como $z = 1$ y a los correspondientes a mujeres como $z = -1$. Consideramos la neurona con entrada $(x_1, x_2)$, pesos $w_1, w_2$ y sesgo $b$ teniendo la función de activación

$$S(u) = \begin{cases} 1, & \text{si } u \geq 0 \\ -1, & \text{si } u < 0. \end{cases}$$

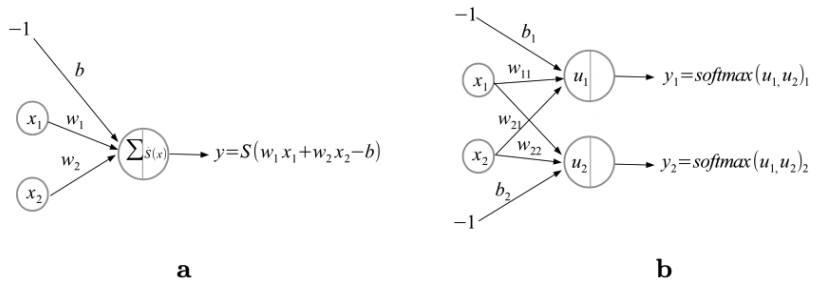



FIGURA 5.8. **a**. La neurona con etiquetas $z \in \{-1, 1\}$. **b**. La red neuronal en el caso de etiquetas en vectores *one-hot*.

La salida de la neurona será la función $y = S(w_1x_1 + w_2x_2 - b)$, ver la Figura 1 **a**. Si $(x_1, x_2)$ corresponde a un varón, entonces $w_1x_1 + w_2x_2 - b > 0$, y entonces la salida es $y = 1$. Si $(x_1, x_2)$ corresponde a una mujer, entonces $w_1x_1 + w_2x_2 - b < 0$, entonces la salida es $y = -1$. Los parámetros de la línea de separación se obtienen minimizando la función distancia entre la salida y los vectores etiquedatos como

$$(w_1^*, w_2^*, b^*) = \arg\min_{(w_1,w_2,b)} \|\mathbf{y} - \mathbf{z}\|^2,$$

con

$$\|\mathbf{y} - \mathbf{z}\|^2 = \sum_{\mathbf{x}\sim\text{datos}} (y(\mathbf{x}) - z(\mathbf{x}))^2,$$

donde $z(\mathbf{x})$ e $y(\mathbf{x})$ son, respectivamente, las etiquetas y las salidas de los datos $\mathbf{x} = (x_1, x_2)$. Esta distancia puede ser minimizada, por ejemplo, usando el método de descenso de gradiente.

2. En este caso usaremos datos etiquetados usando vectores *one-hot*. Asociamos el vector $(1, 0)$ con varones y $(0, 1)$ con mujeres. Consideramos una red neuronal con entrada $\mathbf{x} = (x_1, x_2)$ y dos neuronas con función de activación softmax:

$$(y_1, y_2) = \text{softmax}(u_1, u_2) = \left( \frac{e^{u_1}}{e^{u_1} + e^{u_2}}, \frac{e^{u_2}}{e^{u_1} + e^{u_2}} \right),$$

ver Figura 1 **b**. Acá, $u_1, u_2$ son las señales tomadas de las entradas como

$$u_1 = w_{11}x_1 + w_{21}x_2 - b_1$$
$$u_2 = w_{21}x_1 + w_{22}x_2 - b_2.$$

La función costo a ser minimizada en este caso es

$$\sum_{\mathbf{x}\sim\text{datos}} \left( \|y_1(\mathbf{x}) - z_1(\mathbf{x})\|^2 + \|y_2(\mathbf{x}) - z_2(\mathbf{x})\|^2 \right).$$

## 5.3. La neurona sigmoide

Hemos visto que un perceptrón funciona como una herramienta que provee de dos estados de decisión, 0 y 1, sin valores intermedios. La situación puede ser mejorada si permitimos que la neurona tome valores intermedios en el intervalo $(0, 1)$; de esta forma, las salidas cercanas a 1 corresponden a decisiones que son más proclives de ocurrir, mientras que salidas más cercanas a 0 representan decisiones con menor chance de

suceder. En este caso, la salida de la neurona ocurre como una medida de verosimilitud para tomar decisiones.

Una *neurona sigmoide* es una unidad de cómputo con entrada $\mathbf{x} = (x_1, \ldots, x_n)$ y función de activación dada por la función logística

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Notamos por $\mathbf{w} = (w_1, \ldots, w_n)$ al vector de pesos y por $b$ al sesgo. La salida de la neurona es un número entre 0 y 1 dado por

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} - b) = \frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{x}+b}}.$$

Si consideramos como función de activación a la función logística escalada

$$\sigma_c(z) = \frac{1}{1 + e^{-cz}}, \qquad c > 0,$$

entonces la salida de la neurona sigmoide tiene a la salida del perceptrón cuando $c \to \infty$.

La ventaja de este tipo de neuronas es doble: Por un lado aproxima perceptrones y por otro la función de salida es continua con respecto a los pesos. Esto último no sucede con los perceptrones. Esta propiedad de continuidad es de extrema importancia en varios algoritmos que veremos más adelante.

Vale la pena notar la forma en que los pesos $w_i$ se relacionan con las entradas $x_i$ y la salida $y$. Para eso usamos la fórmula de la inversa de la función logística, $\sigma^{-1} = \ln \frac{x}{1-x}$. Luego la relación $y = \sigma(\mathbf{w} \cdot \mathbf{x} - b)$ se convierte en $\ln \frac{y}{1-y} = \mathbf{w} \cdot \mathbf{x} - b$. Evaluemos ahora esta expresión en dos valores diferentes de los ingresos $x_i = x_i^*$ y $x_i = x_i^* + 1$:

$$\ln \frac{y}{1-y}\Big|_{x_i=x_i^*} = w_1x_1 + \cdots + w_ix_i^* + \cdots + w_nx_n - b,$$
$$\ln \frac{y}{1-y}\Big|_{x_i=x_i^*+1} = w_1x_1 + \cdots + w_i(x_i^* + 1) + \cdots + w_nx_n - b.$$

Restando ambas expresiones y usando las propiedades del logaritmo llegamos a la expresión

$$w_i = \ln \left( \frac{\frac{y}{1-y}\big|_{x_i=x_i^*+1}}{\frac{y}{1-y}\big|_{x_i=x_i^*}} \right),$$

que es una expresión independiente de $x_i^*$. Esta expresión permite obtener el valor de los pesos $w_i$ de la neurona en términos de las salidas, si las entradas se modifican en una unidad.

Una neurona sigmoide puede también tener otras funciones de activación que tengan un aspecto similar, como la tangente hiperbólica $\mathbf{t}(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$.

Si bien diferentes elecciones de función sigmoide pueden tener distintas velocidades de aprendizaje, desde el punto de vista matemático tienen un tratamiento muy similar. Por ejemplo, el cambio infinitesimal en la salida de una neurona sigmoide $y = \varphi(\mathbf{w} \cdot \mathbf{x} - b)$, con función de activación $\varphi$ en términos de los pesos y sesgo es dada por la

expresión diferencial

$$\begin{aligned} dy &= \sum_{i=1}^{n} \frac{\partial y}{\partial w_i} dw_i + \frac{\partial y}{\partial b} db \\ &= \sum_{i=1}^{n} \varphi'(\mathbf{w}\cdot\mathbf{x} - b) x_i dw_i + \varphi'(\mathbf{w}\cdot\mathbf{x} - b)(-1) db \\ &= \varphi'(\mathbf{w}\cdot\mathbf{x} - b) \left( \sum_{i=1}^{n} x_i dw_i - db \right). \end{aligned}$$

Notemos que la variación $dy$ es proporcional a $\varphi'$. Esa constante de proporcionalidad depende de la función sigmoide elegida y en algunos casos es fácilmente calculable. Por ejemplo

$$\begin{aligned} &\text{si } \varphi(z) = \sigma(z) \quad \text{entonces} \quad \varphi'(z) = \sigma'(z) = \sigma(z)(1 - \sigma(z)), \\ &\text{si } \varphi(z) = \mathbf{t}(z) \quad \text{entonces} \quad \varphi'(z) = \mathbf{t}'(z) = 1 - \mathbf{t}^2(z). \end{aligned}$$

La expresión diferencial, en términos de los gradientes se expresa como

$$\nabla y = (\nabla_{\mathbf{w}} y, \partial_b y) = \varphi'(\mathbf{w}\cdot\mathbf{x} - b)(\mathbf{x}, -1).$$

Es decir, el gradiente $\nabla y$ es proporcional al vector extendido de entradas $(\mathbf{x}, -1)$ con constante de proporcionalidad dada por la derivada de la función de activación $\varphi$. Usaremos este hecho en el *algoritmo de retropropagación* (back-propagation algorithm) más adelante.

## 5.4. Regresión logística

In esta sección veremos dos aplicaciones de la neurona sigmoide que aprende usando una *regresión logística*. El primer ejemplo trata con pronosticar la probabilidad de default de una compañía y el segundo con una clasificación binaria en clústeres.

**5.4.1. Probabilidad de default de una compañía.** Estamos interesados en predecir la probabilidad de default de una compañía durante un cierto período de tiempo $[0, T]$ usando una neurona sigmoide. Asumamos que la entrada $\mathbf{x}$ de la neurona es un vector que contiene cierta información de la misma como, por ejemplo: Reservas de efectivo, ingresos, costos, gastos de mano de obra, etc. El conjunto de entrenamiento consiste en $n$ pares $\{(\mathbf{x}_i, z_i)\}_{1\leq i\leq n}$, con $\mathbf{x}_i$ como antes y $z_i \in \{-1, 1\}$. Un valor $z_i = 1$ significa que la $i-$ésima compañía defaulteó en el período $[0, T]$ y un valor $z_i = -1$ significa que no defaulteó. Las mediciones $\{(\mathbf{x}_i, z_i)\}_{1\leq i\leq n}$ representan empíricamente un par de variables aleatorias $(X, Z)$ donde $Z$ toma valores $\pm 1$ y $X$ es un vector aleatorio $m-$dimensional. La probabilidad condicional $\mathbb{P}(Z|X)$ viene dada por la tabla

| $z$ | $-1$ | $1$ |
|---|---|---|
| $\mathbb{P}(z\|\mathbf{x})$ | $1 - h(\mathbf{x})$ | $h(\mathbf{x})$ |

para una cierta función $h\colon \mathbb{R}^m \to [0, 1]$, donde $m$ es la dimensión de la entrada $X$. Una forma conveniente de elegir $h(\mathbf{x})$ es usando una función sigmoide

$$h(\mathbf{x}) = \sigma(\mathbf{w}\cdot\mathbf{x}).$$

El producto interno $\mathbf{w} \cdot \mathbf{x}$ se interpreta como el *puntaje por defecto* de la compañía con la entrada $\mathbf{x}$. Un puntaje alto implica que el valor de $\sigma(\mathbf{w} \cdot \mathbf{x})$ es cercano a 1, lo que se interpreta como una alta probabilidad de default. Por otro lado, un valor bajo (negativo) del puntaje implica un valor de $\sigma(\mathbf{w} \cdot \mathbf{x})$ cercano a 0, que se interpreta como una baja probabilidad de default.

Observemos que la función sigmoide $\sigma$ verifica $\sigma(-x) = 1 - \sigma(x)$, por lo que si $z = -1$,

$$\sigma(z\mathbf{w} \cdot \mathbf{x}) = \sigma(-\mathbf{w} \cdot \mathbf{x}) = 1 - \sigma(\mathbf{w} \cdot \mathbf{x}) = 1 - h(\mathbf{x}),$$

y si $z = 1$,

$$\sigma(z\mathbf{w} \cdot \mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}) = h(\mathbf{x}),$$

Luego, la tabla previa se puede escribir como

| $z$ | $-1$ | $1$ |
|---|---|---|
| $\mathbb{P}(z\|\mathbf{x})$ | $\sigma(-\mathbf{w} \cdot \mathbf{x})$ | $\sigma(\mathbf{w} \cdot \mathbf{x})$ |

La distribución de probabilidad depende entonces de los parámetros $\mathbf{w}$ y lo consideramos como un modelo de distribución producido por la neurona. El siguiente paso es encontrar los pesos $\mathbf{w}$ para los cuales el modelo de distribución aproxima de la mejor manera posible a los datos de entrenamiento $\{(\mathbf{x}_i, z_i)\}_{1 \leq i \leq n}$. Esto lo podemos conseguir eligiendo $\mathbf{w}$ de manera tal que la verosimilitud de los datos $z_i$ al aproximarse por $y_i$ sea máxima. Asumiendo independencia de los datos, tenemos

$$\begin{aligned}
\mathbf{w}^* &= \arg\max_{\mathbf{w}} \mathbb{P}(z_1, \ldots, z_n | \mathbf{x}_1, \ldots, \mathbf{x}_n) \\
&= \arg\max_{\mathbf{w}} \prod_{i=1}^{n} \mathbb{P}(z_i|\mathbf{x}_i) = \arg\max_{\mathbf{w}} \ln\left(\prod_{i=1}^{n} \mathbb{P}(z_i|\mathbf{x}_i)\right) \\
&= \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \ln \mathbb{P}(z_i|\mathbf{x}_i) = \arg\min_{\mathbf{w}} \left(-\sum_{i=1}^{n} \ln \mathbb{P}(z_i|\mathbf{x}_i)\right) \\
&= \arg\min_{\mathbf{w}} \left(-\frac{1}{n}\sum_{i=1}^{n} \ln \mathbb{P}(z_i|\mathbf{x}_i)\right).
\end{aligned}$$

Ahora observemos que

$$\begin{aligned}
-\frac{1}{n}\sum_{i=1}^{n} \ln \mathbb{P}(z_i|\mathbf{x}_i) &= -\frac{1}{n}\sum_{i=1}^{n} \ln \sigma(z_i \mathbf{w} \cdot \mathbf{x}_i) = -\frac{1}{n}\sum_{i=1}^{n} \ln \frac{1}{1 + e^{-z_i \mathbf{w} \cdot \mathbf{x}_i}} \\
&= \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-z_i \mathbf{w} \cdot \mathbf{x}_i}).
\end{aligned}$$

Luego el peso óptimo está dado por

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-z_i \mathbf{w} \cdot \mathbf{x}_i}).$$

**Descenso de gradiente.** No se tiene una fórmula exacta para el valor del peso óptimo $\mathbf{w}^*$, así que lo buscamos aproximar con una sucesión $\mathbf{w}_j$ usando el método de descenso de gradiente.

Llamemos

$$F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-z_i \mathbf{w}\cdot\mathbf{x}_i}),$$

su gradiente viene dado por

$$\begin{aligned}\nabla F(\mathbf{w}) &= -\frac{1}{n}\sum_{i=1}^{n} \frac{z_i \mathbf{x}_i e^{-z_i \mathbf{w}\cdot\mathbf{x}_i}}{1 + e^{-z_i \mathbf{w}\cdot\mathbf{x}_i}} = -\frac{1}{n}\sum_{i=1}^{n} \frac{z_i \mathbf{x}_i}{1 + e^{z_i \mathbf{w}\cdot\mathbf{x}_i}} \\ &= \frac{1}{n}\sum_{i=1}^{n} z_i \mathbf{x}_i \sigma(-z_i \mathbf{w}\cdot\mathbf{x}_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} z_i \mathbf{x}_i \sigma(z_i \mathbf{w}\cdot\mathbf{x}_i) - \frac{1}{n}\sum_{i=1}^{n} z_i \mathbf{x}_i.\end{aligned}$$

Luego, dado un peso inicial $\mathbf{w}_0$, la sucesión aproximante viene dada por

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \eta \nabla F(\mathbf{w}_j) = \mathbf{w}_j - \frac{\eta}{n}\sum_{i=1}^{n} z_i \mathbf{x}_i \sigma(-z_i \mathbf{w}_j \cdot \mathbf{x}_i),$$

donde $\eta > 0$ es la tasa de aprendizaje.

Para aplicar este método, asumiendo que tenemos el conjunto de entrenamiento $\{(\mathbf{x}_i, z_i)\}_{1\le i\le n}$ y si tenemos una nueva entrada $\mathbf{x}$ describiendo los parámetros de otra compañía queremos entonces evaluar la probabilidad de default de la misma. Entonces la probabilidad deseada se calcula como $h(\mathbf{x}) = \sigma(\mathbf{w}^* \cdot \mathbf{x})$, donde $\mathbf{w}^* = \lim_{j\to\infty} \mathbf{w}_j$.

**5.4.2. Clasificación binaria.** Esta sección trata sobre el uso de una neurona sigmoide como un *clasificador binario*. Esto significa que una neurona sigmoide es entrenada para distinguir entre dos clústeres distintos en el plano en donde se usa nuevamente la regresión logística como algoritmo de aprendizaje.

Asumamos que tenemos dos grupos de puntos en el plano: blancos y negros. Queremos poder dividir los puntos en dos grupos, el clúster de puntos negros denotado por $\mathcal{G}_1$ y el clúster de puntos blancos denotado por $\mathcal{G}_2$. Si $\mathbf{x} = (x_1, x_2)$ representa las coordenadas de un punto en el plano, consideramos la función objetivo a la función

$$z(\mathbf{x}) = \begin{cases} 1, & \text{si } \mathbf{x} \in \mathcal{G}_1 \\ -1, & \text{si } \mathbf{x} \in \mathcal{G}_2. \end{cases}$$

Asumamos que existe una línea de decisión en el plano dada por $w_1 x_1 + w_2 x_2 = b$, que intenta dividir los puntos de los clústeres $\mathcal{G}_1$ y $\mathcal{G}_2$. Ver Figura 5.4.2. El objetivo es ajustar los parámetros $w_1, w_2$ y $b$ para que la línea represente la mejor partición de los datos en los distintos clústeres.

Consideremos la partición del plano en líneas paralelas a la línea de decisión

$$\{w_1 x_1 + w_2 x_2 - b = c \colon c \in \mathbb{R}\},$$

ver Figura 5.4.2. La líneas con valores de $c$ positivo corresponden a la región negra $\mathcal{G}_1$, mientras que las que tienen un valor negativo de $c$ corresponden a la región $\mathcal{G}_2$. Podemos pensar al plano como una escala de grises desde el blanco, cuando $c = -\infty$ al negro, cuando $c = +\infty$.

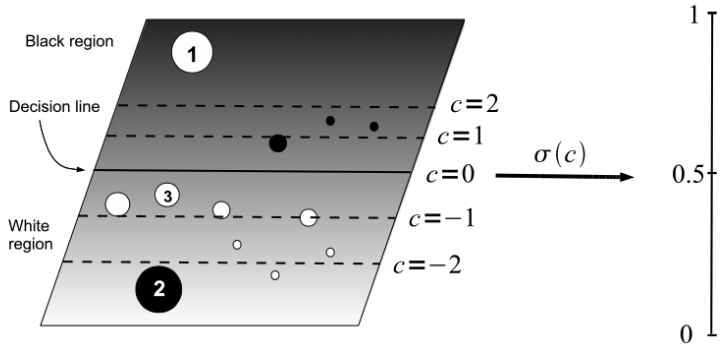



FIGURA 5.9. La familia de líneas $w_1x_1 + w_2x_2 - b = c$, $c \in \mathbb{R}$. La función sigmodea $\sigma$ mapea cada línea a un punto en $(0, 1)$ visto como una probabilidad.

Cada punto es aproximadamente del mismo color que el color del fondo que lo rodea. Si un punto tiene un color que es muy diferente al fondo en donde está, ese elemento "sorpresa" es grande y corresponde a una gran cantidad de información. Por otro lado, si el color del punto no difiere del color de su entorno, entonces la información que provee ese punto es pequeña. Vamos a construir una medida de la información basada en el efecto de esta diferencia de color, que luego será minimizada. La información asociada a un punto $\mathbf{x} = (x_1, x_2)$ se define como

$$H(\mathbf{x}) = -\ln \sigma(z(\mathbf{x})(\mathbf{w} \cdot \mathbf{x} - b)),$$

donde $\mathbf{w} = (w_1, w_2)$. Tratemos de entender esta construcción para los puntos no clasificados y para los que están correctamente clasificados.

*El caso de puntos no clasificados.* Elijamos un punto no clasificado con coordenadas $\mathbf{x}$. Es decir un punto que tiene un color diferente al de su entorno. Por ejemplo tomemos el punto blanco con etiqueta "1" en la Figura 5.4.2. Este es un punto blanco en una región muy negra, por lo que la información asociada con este evento es grande. Representamos este hecho en la figura considerando un radio grande, de manera tal que la información está asociada al área del disco blanco. Tenemos entonces que existe un $c > 0$ tal que $\mathbf{x}$ verifica $\mathbf{w} \cdot \mathbf{x} - b = c$. La función sigmoide $\sigma$ asocia a $c$ con un valor entre 0 y 1 que puede ser considerado como una probabilidad $\mathbb{P}(\mathbf{x}) = \sigma(c)$. La información asociada con el punto $\mathbf{x}$ viene dada con la *función de verosimilitud negativa logarítmica*, $-\ln \mathbb{P}(\mathbf{x}) = -\ln \sigma(c)$. Si $c$ es grande, entonces $\sigma(c)$ es próxima a 1, luego la información $-\ln \mathbb{P}(\mathbf{x})$ es cercana a 0. Esto no tiene sentido, dado que la información debería ser grande. Para solucionar este problema, definimos la información levemente diferente, usando la probabilidad complementaria. Dado que el punto es blanco, $\mathbf{x} \in \mathcal{G}_2$, entonces $z(\mathbf{x}) = -1$. Definimos entonces la probabilidad como $\mathbb{P}(\mathbf{x}) = \sigma(z(\mathbf{x})c) = \sigma(-c) = 1 - \sigma(c)$. Luego, cuando $c$ es grande, la probabilidad es cercana a 0, por ende la información $-\ln \mathbb{P}(\mathbf{x}) = -\ln(z(\mathbf{x})c) = -\ln \sigma(-c) = -\ln(1-\sigma(c))$ es también grande.

De manera análoga, para el punto negro etiquetado como "2" dentro de la región blanca de la Figura 5.4.2 va a poseer una información alta.

*El caso de los puntos correctamente clasificados.* Elegimos ahora un punto correctamente clasificado, por ejemplo el punto blanco con etiqueta "3" de la figura 5.4.2

situado en el área blanca. Notamos por $\mathbf{x}$ a sus coordenadas y $c < 0$ a la constante tal que $\mathbf{w} \cdot \mathbf{x} - b = c$. Observemos que si $c$ es grande (en valor absoluto) y asociamos a $\mathbf{x}$ la probabilidad $\mathbb{P}(\mathbf{x}) = \sigma(z(\mathbf{x})c)$, tenemos en este caso que $z(\mathbf{x}) = -1$ pues $\mathbf{x} \in \mathcal{G}_2$, luego $\sigma(z(\mathbf{x})c)$ estará cerca de 1, luego su logaritmo estará cerca de 0.

Ahora estamos en la posición de buscar la mejor línea usando la información como función de error. Este método resulta equivalente al método de máxima verosimilitud. Consideramos $n$ puntos del plano con coordenadas $\mathbf{x}_i$ y tipo de color $z_i = z(\mathbf{x}_i)$, $1 \leq i \leq n$. La información total se define como la suma de las informaciones puntuales, es decir

$$E(\mathbf{w}, b) = \sum_{i=1}^{n} H(\mathbf{x}_i) = -\sum_{i=1}^{n} \ln \sigma(z_i(\mathbf{w} \cdot \mathbf{x}_i - b)).$$

La mejor línea de decisión es la que corresponde a los valores $\mathbf{w}$ y $b$ que minimicen a la información total $E$. Estos valores óptimos están dados por

$$\begin{aligned}(\mathbf{w}^*, b^*) &= \arg\min_{(\mathbf{w},b)} E(\mathbf{w}, b) = \arg\max_{(\mathbf{w},b)} \sum_{i=1}^{n} \ln \sigma(z_i(\mathbf{w} \cdot \mathbf{x}_i - b)) \\ &= \arg\max_{(\mathbf{w},b)} \ln \left( \prod_{i=1}^{n} \sigma(z_i(\mathbf{w} \cdot \mathbf{x}_i - b)) \right) \\ &= \arg\max_{(\mathbf{w},b)} \ln \left( \prod_{i=1}^{n} \mathbb{P}_{\mathbf{w},b}(z_i|\mathbf{x}_i) \right) \\ &= \arg\max_{(\mathbf{w},b)} \mathbb{P}_{\mathbf{w},b}(z_1, \ldots, z_n|\mathbf{x}_1, \ldots, \mathbf{x}_n),\end{aligned}$$

donde $\mathbb{P}_{\mathbf{w},b}(z|\mathbf{x})$ representa la probabilidad de que el color sea de tipo $z$ dado que las coordenadas del punto son $\mathbf{x}$. En la última igualdad usamos que el logaritmo es creciente y asumimos que los datos son independientes.

**Descenso de gradiente.** Como no tenemos una solución explícita para $\mathbf{w}^*$ y $b^*$ empleamos el método de descenso de gradiente para obtener una aproximación. El gradiente se escribe como

$$\nabla E = (\nabla_{\mathbf{w}} E, \partial_b E).$$

Observemos que el error se escribe como

$$E(\mathbf{w}, b) = -\sum_{i=1}^{n} \ln \sigma(z_i(\mathbf{w} \cdot \mathbf{x}_i - b)) = \sum_{i=1}^{n} \ln(1 + e^{-(z_i(\mathbf{w} \cdot \mathbf{x}_i - b))}),$$

derivando obtenemos

$$\begin{aligned}\nabla_{\mathbf{w}} E &= -\sum_{i=1}^{n} \frac{z_i \mathbf{x}_i}{1 + e^{-(z_i(\mathbf{w} \cdot \mathbf{x}_i - b))}} \\ \partial_b E &= \sum_{i=1}^{n} \frac{z_i}{1 + e^{-(z_i(\mathbf{w} \cdot \mathbf{x}_i - b))}}.\end{aligned}$$

Luego, la sucesión aproximante se construye como

$$\begin{aligned}\mathbf{w}_{j+1} &= \mathbf{w}_j - \eta \nabla_{\mathbf{w}} E(\mathbf{w}_j, b_j) \\ b_{j+1} &= b_j - \eta \partial_b E(\mathbf{w}_j, b_j),\end{aligned}$$

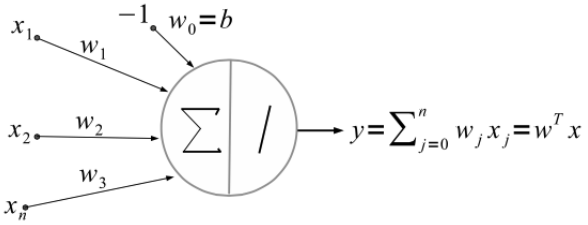



FIGURA 5.10. Neurona lineal con sesgo $b$ y función de activación lineal

donde $\eta > 0$ es la tasa de aprendizaje.

## 5.5. Neuronas lineales

Una *neurona lineal* es una neurona con función de activación lineal $\varphi(x) = x$ y $n$ entradas aleatorias. Su algoritmo de aprendizaje usa la función de costo de cuadrados mínimos. Estas neuronas se pueden usar para el reconocimiento de patrones, filtración de datos y, por supuesto, para aproximar funciones lineales.

La neurona tiene $n$ entradas, que son variables aleatorias, $X_1, \ldots, X_n$. Los pesos para las entradas son números notados por $w_1, \ldots, w_n$. El sesgo es considerado como el peso $w_0 = b$ (ver Figura 5.5). Consideremos $X_0 = -1$ contante y usamos la notación

$$\mathbf{X} = (X_0, X_1, \ldots, X_n), \quad \mathbf{w} = (w_0, w_1, \ldots, w_n).$$

Dado que la función de activación es la identidad, la salida de la neurona está dada por la variable aleatoria unidimencional

$$Y = \sum_{j=0}^{n} w_j X_j = \mathbf{w} \cdot \mathbf{X}.$$

La salida deseada de la neurona viene dado por una variable aleatoria unidimensional $Z$. La idea es ajustar el vector de parámetros $\mathbf{w}$ para que $Y$ aprenda $Z$ y el algoritmo en este caso es el de minimizar la esperanza del cuadrado de la diferencia, por lo que el parámetro óptimo es

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}[(Z - Y)^2].$$

Necesitamos encontrar $\mathbf{w}^*$ y veremos algunas formas de hacerlo, algunas más teóricas y otras más efectivas computacionalmente.

**Solución exacta.** En este caso $\mathbf{w}^*$ puede hallarse exactamente. En efecto

$$\begin{aligned}
\mathbb{E}[(Z - Y)^2] &= \mathbb{E}[Z^2 - 2ZY + Y^2] = \mathbb{E}[Z^2 - 2Z\mathbf{w} \cdot \mathbf{X} + (\mathbf{w} \cdot \mathbf{X})^2] \\
&= \mathbb{E}[Z^2] - 2\mathbb{E}[Z\mathbf{X}] \cdot \mathbf{w} + \mathbb{E}[(\mathbf{w}\mathbf{X}^t)(X\mathbf{w}^t)] \\
&= \mathbb{E}[Z^2] - 2\mathbb{E}[Z\mathbf{X}] \cdot \mathbf{w} + \mathbf{w}\mathbb{E}[X^t X]\mathbf{w}^t \\
&= c - 2\mathbf{b} \cdot \mathbf{w} + \mathbf{w}A\mathbf{w}^t,
\end{aligned}$$

donde $c = \mathbb{E}[Z^2]$ es el segundo momento de la variable objetivo $Z$, $\mathbf{b} = \mathbb{E}[Z\mathbf{X}]$ es el vector que mide la correlación cruzada entre la entrada y la salida, y

$$A = \mathbb{E}[X^t X] = \begin{pmatrix} \mathbb{E}[X_0X_0] & \mathbb{E}[X_0X_1] & \cdots & \mathbb{E}[X_0X_n] \\ \mathbb{E}[X_1X_0] & \mathbb{E}[X_1X_1] & \cdots & \mathbb{E}[X_1X_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}[X_nX_0] & \mathbb{E}[X_nX_1] & \cdots & \mathbb{E}[X_nX_n] \end{pmatrix}$$

es la matriz que describe la autocorrelación de las entradas. En el análisis que haremos vamos a asumir que $A$ es no-degenerada y definida positiva (es decir, las entradas son *coherentes*).

Vamos a notar a la función cuadrática por

$$\xi(\mathbf{w}) = c - 2\mathbf{b} \cdot \mathbf{w} + \mathbf{w}A\mathbf{w}^t,$$

su gradiente viene dado entonces por $(\nabla_{\mathbf{w}}\xi(\mathbf{w}))^t = 2A\mathbf{w}^t - 2\mathbf{b}^t$ y luego el valor del peso óptimo es la solución de $\nabla_{\mathbf{w}}\xi(\mathbf{w}) = 0$ que es la solución del sistema lineal $A\mathbf{w}^t = \mathbf{b}^t$. Como la matriz $A$ es no-degenerada, el sistema tiene una solución única dada por $(\mathbf{w}^*)^t = A^{-1}\mathbf{b}^t$. Este punto es un mínimo dado que la matriz $A$ es definida positiva.

**Método de descenso de gradiente.** En general, cuando $n$ es grande, el cálculo de $A^{-1}$ resulta muy costoso y si la matriz está mal condicionada puede traer grandes errores en su cálculo. Es por eso que resulta conveniente usar métodos de aproximación, como el método de descenso, para el cálculo efectivo de $\mathbf{w}^*$.

Empezamos entonces con un vector de pesos arbitrario $\mathbf{w}_0 \in \mathbb{R}^{n+1}$ y construimos la aproximación $\{\mathbf{w}_j\}_{j\geq 1}$ recursivamente como

$$\begin{aligned} \mathbf{w}_{j+1} &= \mathbf{w}_j - \eta\nabla_{\mathbf{w}}\xi(\mathbf{w}_j) \\ &= \mathbf{w}_j + 2\eta(A\mathbf{w}_j^t - \mathbf{b}^t)^t \\ &= [(\mathbb{I}_n - 2\eta A)\mathbf{w}_j^t + 2\eta\mathbf{b}^t]^t \\ &= [M\mathbf{w}_j^t + 2\eta\mathbf{b}^t]^t, \end{aligned}$$

donde $\mathbb{I}_n$ es la matriz identidad y $M = \mathbb{I}_n - 2\eta A$.

**Convergencia del método.** Veamos para este caso que la iteración converge a la solución buscada. Si iteramos la recursión, obtenemos la siguiente fórmula para $\mathbf{w}_j$

$$\begin{aligned} \mathbf{w}_j^t &= M^j\mathbf{w}_0^t + (M^{j-1} + M^{j-2} + \cdots + M + \mathbb{I}_n)2\eta\mathbf{b}^t \\ &= M^j\mathbf{w}_0^t + (\mathbb{I}_n - M^j)(\mathbb{I}_n - M)^{-1}2\eta\mathbf{b}^t \\ &= M^j\mathbf{w}_0^t + (\mathbb{I}_n - M^j)A^{-1}\mathbf{b}^t. \end{aligned}$$

Asumamos por un momento que la tasa de aprendizaje $\eta > 0$ es tal que

$$\lim_{j\to\infty} M^j = \mathbb{O}_n \quad \text{(la matriz nula)}.$$

Entonces, tomando el límite en la fórmula previa obtenemos que

$$(\mathbf{w}^*)^t = \lim_{j\to\infty} \mathbf{w}_j^t = A^{-1}\mathbf{b}^t$$

Veamos ahora la suposición de que $M^j \to \mathbb{O}_n$. Como $M$ es una matriz simétrica, resulta diagonalizable con autovalores reales y a partir de ahí es fácil ver que la suposición resulta equivalente a ver que los autovalores de $M$, $\{\lambda_i\}_{1\leq i\leq n}$, verifican que $|\lambda_i| < 1$

*Paso 1.* Veamos que $\lambda_i < 1$.

Observemos que si $\lambda_i$ es un autovalor de $M$, de la definición de $M$ sigue que $\alpha_i = \frac{1-\lambda_i}{2\eta}$ es un autovalor de $A$. Pero como $A$ es definida positiva implica que $\alpha_i > 0$ lo que nos da que $\lambda_i < 1$.

*Paso 2.* Veamos que si $\eta$ es pequeño, entonces $\lambda_i > 0$.

Observemos que $\lambda_i > 0$ es equivalente a $\frac{1-\lambda_i}{\alpha_i} < \frac{1}{\alpha_i}$ donde notamos por $\{\alpha_i\}_{1\leq i\leq n}$ a los autovalores de $A$. Por la observación del Paso 1, esto resulta equivalente a $2\eta < \frac{1}{\alpha_i}$. Luego, si elegimos la tasa de aprendizaje como

$$0 < \eta < \min_{1\leq i\leq n} \frac{1}{2\alpha_i},$$

obtenemos lo deseado.

**Estimación del error.** Estimemos ahora el error de la $j-$ésima iteración, $\varepsilon_j = \|\mathbf{w}_j - \mathbf{w}^*\|$, usando la fórmula para $\mathbf{w}_j$, recordando que $(\mathbf{w}^*)^t = A^{-1}\mathbf{b}^t$. Tenemos entonces

$$\mathbf{w}_j^t - (\mathbf{w}^*)^t = M^j\mathbf{w}_0^t + (\mathbb{I}_n - M^j)(\mathbf{w}^*)^t - (\mathbf{w}^*)^t = M^j(\mathbf{w}_0 - \mathbf{w}^*)^t.$$

Luego

$$\varepsilon_j = \|\mathbf{w}_j - \mathbf{w}^*\| = \|M^j(\mathbf{w}_0 - \mathbf{w}^*)^t\| \leq \|M\|^j\|\mathbf{w}_0 - \mathbf{w}^*\| = \mu^j d,$$

donde $\mu = \|M\|$ (la norma como operador) y $d = \|\mathbf{w}_0 - \mathbf{w}^*\|$.

Ahora, es fácil ver que $\|M\| = \max_{1\leq i\leq n} |\lambda_i|$, pero como $0 < \lambda_i < 1$ para $1 \leq i \leq n$, tenemos que $0 < \mu < 1$, por lo que $\mu^j d \to 0$ cuando $j \to \infty$.

## 5.6. Neurona de entrada continua

En esta sección introducimos el concepto de *neurona de entrada continua*, que es una neurona con una cantidad no numerable de pesos. Este concepto será usado más adelante en la sección 7.2.1 para la construcción de redes neuronales continuas.

Recomiendo consultar también el capítulo 10 de **[5]** donde se estudia el aprendizaje en el contexto de neuronas de entrada continua.

Una neurona de entrada continua tiene como entrada $x \in [a, b]$. El peso asociado al valor $x$ viene dado por $w(dx)$, donde $w$ es una medida sobre el intervalo $[a, b]$. La primera mitad de la unidad de cómputo integra las entradas con respecto a la medida $w$. La función de salida en este caso toma la forma

$$y = \sigma\left(\int_a^b x\, dw(x)\right). \tag{5.6.1}$$

En particular esto se aplica a la situación en donde la entrada es una variable aleatoria que toma valores en el intervalo $[a, b]$. Si $w$ representa la medida de distribución de $X$, entonces la salida de la neurona depende de su esperanza, $y = \sigma(\mathbb{E}[X])$.

Dependiendo del tipo particular de medida de peso que se utilice, la fórmula (5.6.1) provee de una representación general para la salida de una neurona tanto si tiene un número finito de entradas, como discutimos previamente, un número infinito numerable o un número no numerable de entradas.

Veamos algunos ejemplos.

EJEMPLO 5.6.1 (Neurona con una medida de Dirac). Sea $x_0 \in [0,1]$ fijo y consideremos la medida de Dirac $\delta_{x_0}$ definida por

$$\delta_{x_0}(A) = \begin{cases} 1, & \text{si } x_0 \in A \\ 0, & \text{si } x_0 \notin A, \end{cases}$$

para cualquier conjunto medible $A$. Esto corresponde al caso en que la variable aleatoria $X$ toma sólo el valor $x_0$. La función de salida en este caso es una constante

$$y = \sigma\left(\int_a^b x\, d\delta_{x_0}(x)\right) = \sigma(x_0).$$

Dado que no hay parámetros que ajustar, no hay nada que aprender.

EJEMPLO 5.6.2 (Neurona con una medida discreta). Sea $E = \{x_1, \ldots, x_n\}$ un conjunto finito de puntos del intervalo $[0,1]$ y consideremos la medida discreta

$$\mu(A) = \sum_{x_j \in E} w_j \delta_{x_j}(A),$$

donde el número positivo $w_j$ es la masa del punto $x_j$. La función de salida está dada por

$$y = \sigma\left(\int_a^b x\, d\mu(x)\right) = y = \sigma\left(\sum_{j=1}^n w_j x_j\right),$$

que corresponde al modelo de neurona clásico. El algoritmo de aprendizaje en este caso ajusta los pesos $w_j$ para minimizar la función de error, por ejemplo,

$$F(w) = \frac{1}{2}\left(\sigma\left(\sum_{j=1}^n w_j x_j\right) - z\right)^2,$$

donde $z$ es el valor deseado de la neurona dada la observación $\mathbf{x} = (x_1, \ldots, x_n)$.

EJEMPLO 5.6.3 (Neurona con medida continua). Sea $\mu$ una medida absolutamente continua con respecto a la medida de Lebesgue. Es decir $d\mu(x) = p(x)dx$ para una función no negativa y medible $p$ en el intervalo $[0,1]$. Luego, la función de salida de la neurona está dada por

$$y = \sigma\left(\int_0^1 x\, d\mu(x)\right) = \sigma\left(\int_0^1 xp(x)\, dx\right),$$

que depende de la función de peso $p(x)$. El algoritmo debe entonces optimizar la neurona eligiendo la función $p(x)$ que minimice el *funcional de error*

$$F(p) = \frac{1}{2}\left(\sigma\left(\int_0^1 xp(x)\, dx\right) - z\right)^2$$

Si $\mu$ es una *medida de probabilidad*, entonces $p(x)$ es una densidad de probabilidad lo que implica que $\int_0^1 p(x)\,dx = 1$. Luego, si suponemos que $p(x)$ minimiza la función de error, debe verificar que

$$F(p) \leq F(q)$$

para toda densidad de probabilidad $q(x)$. Para encontrar esta densidad óptima se usa el *cálculo de variaciones*. Asumamos que $p$ es óptima y consideremos $\varphi(x)$ de manera tal que $\int_0^1 \varphi(x)\,dx = 0$. Entonces $p_t(x) = p(x) + t\varphi(x)$ verifica que $\int_0^1 p_t(x)\,dx = 1$ y por ende

$$F(p) \leq F(p_t), \quad \forall t \in \mathbb{R}.$$

Luego, si $j(t) = F(p_t)$ tenemos que $j(t)$ alcanza su mínimo en $t = 0$ por lo que se debe tener $j'(0) = 0$.

$$\begin{aligned} j'(t) &= \frac{d}{dt}F(p_t) = \frac{d}{dt}\frac{1}{2}\left(\sigma\left(\int_0^1 xp_t(x)\,dx\right) - z\right)^2 \\ &= \left(\sigma\left(\int_0^1 xp_t(x)\,dx\right) - z\right)\sigma'\left(\int_0^1 xp_t(x)\,dx\right)\frac{d}{dt}\int_0^1 xp_t(x)\,dt \\ &= \left(\sigma\left(\int_0^1 xp_t(x)\,dx\right) - z\right)\sigma'\left(\int_0^1 xp_t(x)\,dx\right)\int_0^1 x\varphi(x)\,dt. \end{aligned}$$

Luego,

$$0 = j'(0) = \left(\sigma\left(\int_0^1 xp(x)\,dx\right) - z\right)\sigma'\left(\int_0^1 xp(x)\,dx\right)\int_0^1 x\varphi(x)\,dt.$$

Observemos que como $0 < x < 1$ tenemos que $\int_0^1 x\varphi(x)\,dx < \int_0^1 \varphi(x)\,dx = 0$. Si $\sigma'(x) \neq 0$ (por ejemplo para la función logística) entonces tenemos que

$$j'(0) = 0 \quad \text{si y sólo si} \quad \sigma\left(\int_0^1 xp(x)\,dx\right) = z$$

Lo que nos da una condición necesaria para encontrar al mínimo del funcional.

## 5.7. Resumen

En este capítulo se presentaron distintos tipos de neuronas. Obviamente existen más tipos, pero hemos elegido presentar las más clásicas: Perceptrón, sigmoide y lineal. Todas las neuronas se caracterizan por una entrada, un sistema de pesos, un sesgo, una función de activación y una salida.

El perceptrón es un modelo de neurona con una función escalón como función de activación. La neurona dispara si la señal de entrada supera un cierto umbral. Su salida tiene una discontinuidad de salto. Un perceptrón puede ser usado como un clasificador binario para clasificar dos clústeres que son linealmente separables. Puede ser también usado para aprender funciones Booleanas como AND y OR, pero no puede aprender la función XOR.

La neurona sigmoideal tiene una función de activación sigmoideal. Si salida es continua y puede saturar si la señal es o muy grande o muy pequeña. La neurona sigmoideal

puede aprender usando regresión logística, que es equivalente al echo que los pesos vengan dados por el método de máxima verosimilitud. Presentamos algunas aplicaciones de esta neurona como clasificación binaria y predicción de probabilidades.

Las neuronas lineales tienen entradas a variables aleatorias y la función de activación es lineal. Sus pesos óptimos pueden ser obtenidos de manera exacta aunque en aplicaciones prácticas es mejor y más simple entrenar a la neurona con un método de descenso de gradiente.

## 5.8. Ejercicios

EJERCICIO 5.8.1. Recordemos que $\neg x$ es la negación de la variable Booleana $x$.

(a) Mostrar que un perceptrón puede aprender la función Booleana $y = x_1 \wedge \neg x_2$, con $x_1, x_2 \in \{0, 1\}$
(b) Misma pregunta que en (a) para la función Booleana $y = x_1 \vee \neg x_2$.
(c) Mostrar que un perceptrón con una entrada Booleana $x$ puede aprender la negación $y = \neg x$. ¿Qué sucede si queremos usar una neurona lineal?
(d) Mostrar que un perceptrón con tres entradas Booleanas $x_1, x_2, x_3$ puede aprender $x_1 \wedge x_2 \wedge x_3$. ¿Qué sucede con $x_1 \vee x_2 \vee x_3$?

EJERCICIO 5.8.2. Mostrar que dos conjntos finitos y linealmente separables $A$ y $B$ pueden ser separados por un perceptrón con pesos racionales.

EJERCICIO 5.8.3. (a) Asumamos que las entradas de una neurona lineal son independientes y normalmente distribuidas, $X_i \sim N(0, \sigma_i^2)$, $i = 1, \ldots, n$. Hallas los pesos óptimos $\mathbf{w}^*$.
(b) Una variable aleatoria unidimensional con media cero, $Z$, es aprendida por una neurona lineal con entrada $X$. Asumamos que la entrada $\mathbf{X}$ y el objetivo $Z$ son independientes. Escribir la función de costo y hallar los parámetros óptimos $\mathbf{w}^*$. Interpretar el resultado.

EJERCICIO 5.8.4. Mostrar la equivalencia entre el algoritmo de regresión lineal y el de aprendizaje de una neurona lineal.

EJERCICIO 5.8.5. Considerar $n$ puntos, $P_1, \ldots, P_n$ incluidos en media circunferencia y notemos por $\mathbf{x}_1, \ldots, \mathbf{x}_n$ sus coordenadas. Un perceptrón puede aprender la mencionada media circunferencia usando el siguiente algoritmo:

1. Empezamos por una media circunferencia arbitraria determinada por su diámetro y un vector unitario $\mathbf{w}_0$. Entonces seleccionamos un punto incorrectamente clasificado $P_{i_0}$, i.e. un punto tal que $\mathbf{w}_0 \cdot \mathbf{x}_{i_0} < 0$. Ver Figura 5.8.
2. Rotemos el diámetro de manera tal que la nueva normal es $\mathbf{w}_1 = \mathbf{w}_0 + \mathbf{x}_{i_0}$. Mostrar que ahora el punto $P_{i_0}$ es correctamente clasificado.
3. Repitiendo las dos etapas previas, construimos inductivamente una sucesión de vectores $\{\mathbf{w}_m\}_{m \geq 1}$ tal que $\mathbf{w}_{m+1} = \mathbf{w}_m + \mathbf{x}_{i_m}$, donde $P_{i_m}$ es un punto mal clasificado en la etapa $m$. Mostrar que este proceso termina en un número finito de pasos.

EJERCICIO 5.8.6. Modificar el algoritmo del ejercicio anterior para el caso en que los puntos $P_1, \ldots, P_n$ están incluidos en un semi espacio.

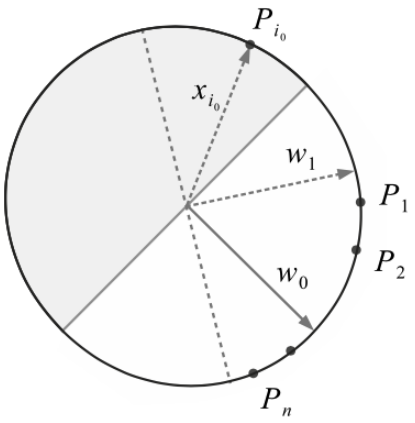



FIGURA 5.11. Figura para el Ejercicio 5.8.5

EJERCICIO 5.8.7. Notemos por $\mathbf{1}_A(x)$ a la función caraterística (o indicadora) del conjunto $A$, es decir $\mathbf{1}_A(x) = 1$ si $x \in A$ y $\mathbf{1}_A(x) = 0$ si $x \notin A$.

(a) Mostrar que la función

$$\varphi(x_1, x_2) = \mathbf{1}_{\{x_2 > x_1 + 0{,}5\}}(x_1, x_2) + \mathbf{1}_{\{x_2 z x_1 + 0{,}5\}}(x_1, x_2)$$

implementa XOR.

(b) Mostrar que XOR puede ser implementada por una combinación lineal de dos preceptrones.

Capítulo 6

# Redes neuronales

Hasta el momento hemos discutido el caso de una única neurona. En esta sección consideraremos varias capas de neuronas cuyas salidas alimentan otras capas de neuronas formando así *redes neuronales*. Una capa de neuronas es un paso de procesamiento en una red neuronal y puede haber de distintos tipos, dependiendo de los pesos y de las funciones de activación usada en sus neuronas. La parte principal de esta capítulo trata sobre el entrenamiento de redes neuronales usando el *algoritmo de retropropagación* (backpropagation algorithm en inglés).

Dado que el estudio de redes neuronales se basa fuertemente en notación, empezamos con un ejemplo de calentamiento.

## 6.1. Un ejemplo de red neuronal

Consideremos dos neuronas que tienen idénticas entradas $x_1, x_2$ y diferentes salidas $y_1$ e $y_2$ respectivamente. Las salidas se usan para alimentar una tercera neurona con salida $y$. Ver figura 6.1. Asumimos que la función de activación de todas las neuronas es la misma y la notamos por $\phi$. Cada neurona tiene un sesgo que es considerado como un peso de una "falsa" entrada igual a $-1$. Ahora ensamblamos estas neuronas juntas como una red neuronal equivalente, como se ve en la Figura 6.1. Esto forma una capa de dos neuronas en el medio que se la llama la *capa escondida*.

Las sinapsis entre las neuronas están representadas por ejes y cada uno de esos es asociado con un peso, notado por $w_{ij}^{(\ell)}$. Los índices tienen el siguiente significado: El superíndice, $(\ell)$, representa el índice de la capa en la que entra la sinapsis. El índice $\ell = 1$ es usado para los pesos que alimentan la capa escondida y el índice $\ell = 2$ es usado para los pesos que entran en la capa de salida, que consiste en la última neurona. El índice inferior $i$ indica la neurona de entrada. El índice $i = 0$ es siempre usado para el sesgo, mientras que $i = 1, 2$ representa el índice de la entrada. El segundo índice, $j$, corresponde a la neurona de salida, i.e., el índice de la neurona a la que apunta la sinapsis.

Las entradas forman la *primera capa* de la red. Estos se notan con un superíndice igual a 0: $x_0^{(0)} = -1$, $x_1^{(0)}$ y $x_2^{(0)}$. Notemos que los sesgos de las neuronas en la capa escondida son notados por $b_1^{(1)} = w_{01}^{(1)}$ y $b_2^{(1)} = w_{02}^{(1)}$, mientras que el sesgo de la neurona en la capa de salida es $b_1^{(2)} = w_{01}^{(2)}$.

La primera neurona de la capa escondida toma la información de la capa de entrada en la señal

$$s_1^{(1)} = w_{01}^{(1)} x_0^{(0)} + w_{11}^{(1)} x_1^{(0)} + w_{21}^{(1)} x_2^{(0)} = \sum_{i=1}^{2} w_{i1}^{(1)} x_i^{(0)} - b_1^{(1)}.$$

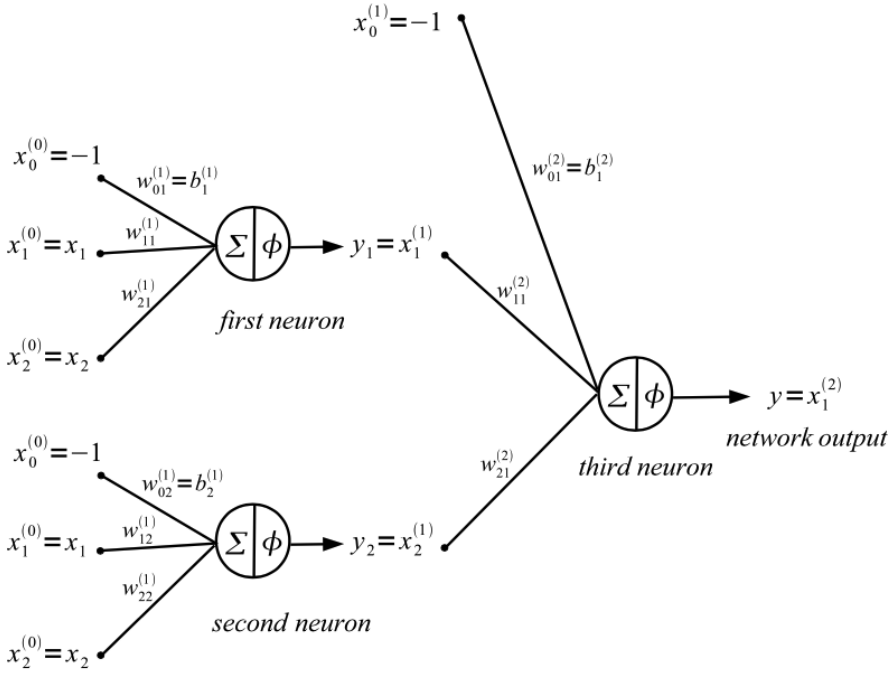



FIGURA 6.1. Dos neuronas con idénticas entradas $x_1, x_2$, y salidas $y_1, y_2$ que alimentan una tercera neurona con salida $y$.

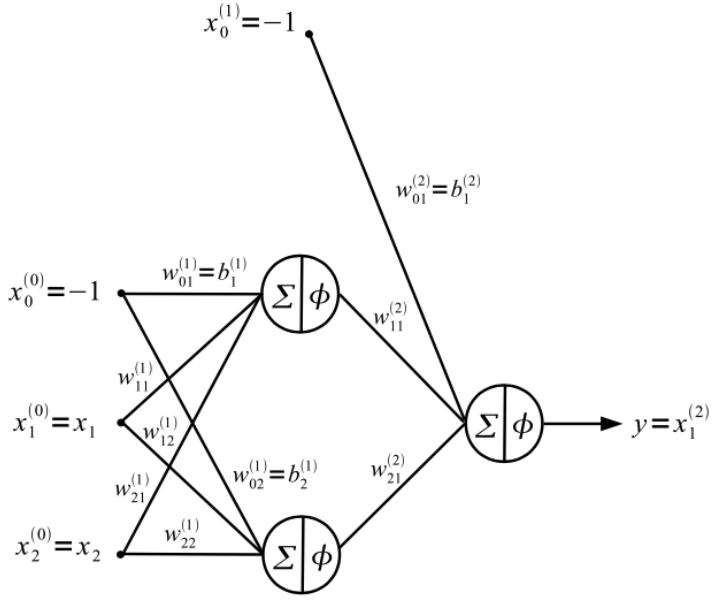



FIGURA 6.2. Red neuronal con una capa escondida, teniendo dos entradas $x_1, x_2$ y dos neuronas en la capa escondida.

Luego, se aplica la función de activación $\phi$ a la señal para obtener la salida

$$y_1 = x_1^{(1)} = \phi(s_1^{(1)}) = \phi\left(\sum_{i=1}^{2} w_{i1}^{(1)} x_i^{(0)} - b_1^{(1)}\right).$$

De manera análoga, la segunda neurona de la capa escondida toma la información en la señal

$$s_2^{(1)} = w_{02}^{(1)} x_0^{(0)} + w_{12}^{(1)} x_1^{(0)} + w_{22}^{(1)} x_2^{(0)} = \sum_{i=1}^{2} w_{i2}^{(1)} x_i^{(0)} - b_2^{(1)},$$

y luego le aplica la función de activación $\phi$ para obtener la salida

$$y_2 = x_2^{(1)} = \phi(s_2^{(1)}) = \phi\left(\sum_{i=1}^{2} w_{i2}^{(1)} x_i^{(0)} - b_2^{(1)}\right).$$

Estas relaciones pueden ser manipuladas de manera más simple si son escritas en forma matricial. Las señales en la capa escondida se pueden escribir como

$$\begin{pmatrix} s_1^{(1)} \\ s_2^{(1)} \end{pmatrix} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \end{pmatrix} \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \end{pmatrix} - \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \end{pmatrix},$$

o en forma condensada como

$$\mathbf{s}^{(1)} = {W^{(1)}}^t \mathbf{x}^{(0)} - \mathbf{b}^{(1)},$$

donde $\mathbf{s}^{(1)}$ representa el vector de señales, $W^{(1)}$ es la matriz de pesos, $\mathbf{x}^{(0)}$ es el vector de entrada y $\mathbf{b}^{(1)}$ es el vector de sesgos para las neuronas en la capa escondida. Observemos que se toma la transpuesta en $W^{(1)}$ por como hemos elegido la numeración de los coeficientes y en esta ocasión tomamos los vectores $\mathbf{s}^{(1)}$, $\mathbf{x}^{(0)}$ y $\mathbf{b}^{(1)}$ como vectores columna.

La salida de la capa escondida es el vector

$$\mathbf{x}^{(1)} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \end{pmatrix} = \begin{pmatrix} \phi(s_1^{(1)}) \\ \phi(s_2^{(1)}) \end{pmatrix} = \phi(\mathbf{s}^{(1)}),$$

donde adoptamos la convención de que aplicar la función $\phi$ a un vector da como resultado un nuevo vector que se calcular aplicando $\phi$ a cada componente.

Estas dos últimas fórmulas implican la relación

$$\mathbf{x}^{(1)} = \phi({W^{(1)}}^t \mathbf{x}^{(0)} - \mathbf{b}^{(1)}),$$

que nos da el vector de salida de la capa escondida como función de las entradas. La multiplicación de $\mathbf{x}^{(0)}$ por la matriz ${W^{(1)}}^t$ es una transformación lineal y restar el vector de sesgos $\mathbf{b}^{(1)}$ lo convierte en una transformación afín. La función de activación $\phi$ es la que agrega la no linealidad a la salida.

La neurona de la última capa convierte esta última información en la señal

$$s^{(2)} = s_1^{(2)} = w_{01}^{(2)} x_0^{(1)} + w_{11}^{(2)} x_1^{(1)} + w_{21}^{(2)} x_2^{(1)} = \sum_{i=1}^{2} w_{i1}^{(2)} x_i^{(1)} - b_1^{(2)}.$$

En la forma matricial, esto se escribe como

$$\mathbf{s}^{(2)} = {W^{(2)}}^t \mathbf{x}^{(1)} - \mathbf{b}^{(2)},$$

donde ahora la matriz de coeficientes y el vector de entrada vienes dados por

$$W^{(2)} = \begin{pmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \end{pmatrix} \qquad \text{y} \qquad \mathbf{x}^{(1)} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \end{pmatrix}.$$

En este caso, la señal que sale de la última neurona $\mathbf{s}^{(2)} = \mathbf{s}_1^{(2)}$ es una matriz unidimensional. El subíndice 1 indica que hay sólo una única neurona en la capa de salida. La salida de la red se obtiene aplicando la función de activación a la señal $\mathbf{s}^{(2)}$ de la forma

$$y = \mathbf{x}_1^{(2)} = \phi(\mathbf{s}^{(2)}) = \phi({W^{(2)}}^t \mathbf{x}^{(1)} - \mathbf{b}^{(2)}).$$

Notamos a la salida $y$ para ser consistente con los capítulos previos y también con $\mathbf{x}_1^{(2)}$ en donde el superíndice (2) indica que es la salida de la segunda capa (en realidad la tercera si contamos la capa de datos de entrada) y el subíndice 1 significa que es la salida de la primera (y única) neurona de esta capa.

Si usamos la fórmula para $\mathbf{x}^{(1)}$ que obtuvimos, podemos expresar la salida final en términos de las entradas originales como

$$y = \phi({W^{(2)}}^t \phi({W^{(1)}}^t \mathbf{x}^{(0)} - \mathbf{b}^{(1)}) - \mathbf{b}^{(2)}). \tag{6.1.1}$$

Notemos que las transformaciones afines que consisten en multiplicar por las matrices de pesos y restar los vectores de sesgo, se alternan con la transformación no lineal que es inducida por la función de activación $\phi$. Si se considera el mapa $f_{w,b}\colon \mathbb{R}^2 \to \mathbb{R}$ dado por

$$f_{w,b}(\mathbf{x}) = \phi({W^{(2)}}^t \phi({W^{(1)}}^t \mathbf{x} - \mathbf{b}^{(1)}) - \mathbf{b}^{(2)}),$$

lo llamamos el *mapa de entrada-salida* de la red neuronal.

Observación 6.1.1. Observemos que si las neuronas de la red son lineales, i.e. la función de activación es la identidad $\phi(x) = x$, entonces la salida es

$$y = (W^1 W^2)^t \mathbf{x}^{(0)} - {W^{(2)}}^t \mathbf{b}^{(1)} - \mathbf{b}^{(2)} = W^t \mathbf{x}^{(0)} - \mathbf{b},$$

donde $W = W^{(1)} W^{(2)}$ y $\mathbf{b} = {W^{(2)}}^t \mathbf{b}^{(1)} + \mathbf{b}^{(2)}$. Luego toda la red neuronal resulta equivalente a una única neurona. Es decir que el estudio de redes neuronales de neuronas lineales se reduce al estudio de una única neurona lineal.

Observación 6.1.2. Asumamos ahora que la red neuronal de la sección está compuesta por todos perceptrones con función de activación $\phi(x) = H(x)$. La salida de la red en este caso es dada por

$$y = H\left(w_1^{(2)} H(x_1 w_{11}^{(1)} + x_2 w_{21}^{(1)} - b_1^{(1)}) + w_2^{(2)} H(x_1 w_{12}^{(1)} + x_2 w_{22}^{(1)} - b_2^{(1)}) - b^{(2)}\right).$$

Notemos que esta red puede aprender exactamente la función Booleana XOR. En efecto, esto lo logramos eligiendo los pesos de la forma

$$y = f(x_1, x_2) = H(H(x_1 + x_2 - 0{,}5) - H(x_1 + x_2 - 1{,}5) - 0{,}5).$$

Se puede verificar fácilmente que $f(0,0) = f(1,1) = 0$ y $f(1,0) = f(0,1) = 1$.

La necesidad de usar tres perceptrones para aprender XOR se puede explicar de la siguiente manera. En el Ejercicio 5.8.1 vimos que las funciones $y_1 = x_1 \wedge \neg x_2$ e $y_2 = \neg x_1 \wedge x_2$ pueden ser aprendidas por un perceptrón cada una. Se necesita un perceptrón más para aprender la función $y_1 \vee y_2$, que es la función XOR, $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$.

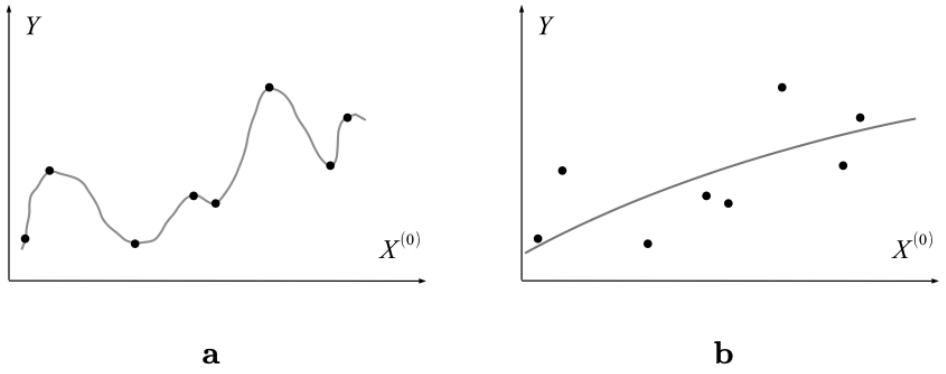



FIGURA 6.3. **a.** El modelo sobreajusta los datos de entrada. **b.** El modelo generaliza bien.

**6.1.1. Variación total y regularización.** El rendimiento de generalización de una red neuronal depende de la sensibilidad de la salida con respecto a su entrada. Asumamos que tenemos dos redes que han sido entrenadas con los mismos datos: la red que se muestra en la figura 6.1.1 **a**, que sobreajusta a los datos, y la que se muestra en la figura 6.1.1 **b**, que generaliza bien. La diferencia geométrica entre estos modelos consiste en el hecho de que en el primer caso la variación total de la salida es mayor a la de la segunda. La variación total está relacionada a la sensibilidad de la salida por la fórmula. Si $y = f(\mathbf{x}^{(0)})$ denota el mapa de entrada-salida de la red, la variación total representa la acumulación total de las variaciones locales

$$V(f) = \int |df(\mathbf{x})|,$$

donde la integral se calcula sobre el espacio de entrada. Luego, la variación total $V(f)$ juega un rol importante al evaluar la performance de evaluación de la red y es el motivo por el que es necesario contar con una fórmula para el diferncial $dy = df(\mathbf{x})$ y lo calcularemos en lo que sigue.

La fórmula (6.1.1) nos da la salida en función de la entrada. Asumiendo que los pesos y sesgos se mantienen fijos, los cambios en la salida $y$ cuando se producen pequeños cambios en las entradas $\mathbf{x}^{(0)}$ se calculan aplicando la regla de la cadena:

$$\begin{aligned} dy &= \frac{dy}{d\mathbf{s}^{(2)}} d\mathbf{s}^{(2)} = \frac{dy}{d\mathbf{s}^{(2)}} \frac{d\mathbf{s}^{(2)}}{d\mathbf{x}^{(1)}} d\mathbf{x}^{(1)} \\ &= \frac{dy}{d\mathbf{s}^{(2)}} \frac{d\mathbf{s}^{(2)}}{d\mathbf{x}^{(1)}} \frac{d\mathbf{x}^{(1)}}{d\mathbf{s}^{(1)}} d\mathbf{s}^{(1)} \\ &= \frac{dy}{d\mathbf{s}^{(2)}} \frac{d\mathbf{s}^{(2)}}{d\mathbf{x}^{(1)}} \frac{d\mathbf{x}^{(1)}}{d\mathbf{s}^{(1)}} \frac{d\mathbf{s}^{(1)}}{d\mathbf{x}^{(0)}} d\mathbf{x}^{(0)}. \end{aligned}$$

Usando que

$$\begin{aligned} \frac{dy}{d\mathbf{s}^{(2)}} &= \phi'(\mathbf{s}^{(2)}), \qquad \frac{d\mathbf{x}^{(1)}}{d\mathbf{s}^{(1)}} = \phi'(\mathbf{s}^{(1)}), \\ \frac{d\mathbf{s}^{(2)}}{d\mathbf{x}^{(1)}} &= W^{(2)^t}, \qquad \frac{d\mathbf{s}^{(1)}}{d\mathbf{x}^{(0)}} = W^{(1)^t}, \end{aligned}$$

obtenemos

$$dy = \phi'(\mathbf{s}^{(2)})W^{(2)^t}\phi'(\mathbf{s}^{(1)})W^{(1)^t}d\mathbf{x}^{(0)}. \tag{6.1.2}$$

A partir de la fórmula (6.1.2) vemos que existe una relación entre el valor de la variación $V(f)$ y los pesos $W^{(\ell)}$, $\ell = 1, 2$. Luego, si requerimos que estos pesos sean pequeños, por ejemplo en norma $L^1$ o $L^2$ (i.e. $\|W^{(\ell)}\|_1 \leq 1$ o $\|W^{(\ell)}\|_2 \leq 1$) esto hará disminuir $|dy|$ y por ende $V(f)$. Esto, como consecuencia, mejorará la performance de generalización.

En el caso cuando la función de activación es la función logística $\phi = \sigma$, usando propiedades de la logística, la fórmula (6.1.2) se puede escribir en la forma

$$dy = \sigma(\mathbf{s}^{(2)})(1 - \sigma(\mathbf{s}^{(2)}))W^{(2)^t}\sigma(\mathbf{s}^{(1)})(1 - \sigma(\mathbf{s}^{(1)}))W^{(1)^t}d\mathbf{x}^{(0)}.$$

Esta fórmula será útil en la resolución de algunos de los ejercicios.

**6.1.2. Retropropagación.** Consideremos una función de costo $C(w, b)$ que mide la proximidad entre la salida de la red $y = f_{w,b}(\mathbf{x})$ y el objetivo $z$. Asumamos que la función de costo es una función suave de los pesos y sesgos. Su gradiente está dado por

$$\nabla C = (\nabla_w C, \nabla_b C).$$

Para la red que vimos en el ejemplo, este vector tiene dimensión nueve, dependiendo de seis pesos y tres sesgos. Vamos a realizar el cálculo de este gradiente hacia atrás, desde los últimos pesos y sesgos hacia los primeros por un procedimiento que se llama *retropropagación*.

Empezamos calculando las derivadas parciales de la función de costo con respecto a los pesos que tienen superíndice $\ell = 2$. Vamos también a usar que la función de costo $C$ depende de $y$ que a su vez depende de $w_{ij}^{(2)}$ a través de $\mathbf{s}_1^{(2)}$ como $y = \phi(\mathbf{s}_1^{(2)})$. Aplicando la regla de la cadena obtenemos

$$\begin{aligned}
\frac{\partial C}{\partial b_1^{(2)}} = \frac{\partial C}{\partial w_{01}^{(2)}} &= \frac{\partial C}{\partial s_1^{(2)}}\frac{\partial s_1^{(2)}}{\partial w_{01}^{(2)}} = \delta_1^{(2)}x_0^{(1)} = -\delta_1^{(2)} \\
\frac{\partial C}{\partial w_{11}^{(2)}} &= \frac{\partial C}{\partial s_1^{(2)}}\frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}} = \delta_1^{(2)}x_1^{(1)} \\
\frac{\partial C}{\partial w_{21}^{(2)}} &= \frac{\partial C}{\partial s_1^{(2)}}\frac{\partial s_1^{(2)}}{\partial w_{21}^{(2)}} = \delta_1^{(2)}x_2^{(1)},
\end{aligned}$$

donde usamos el hecho de que $s^{(2)} = w_{01}^{(2)}x_0^{(1)} + w_{11}^{(2)}x_1^{(1)} + w_{21}^{(2)}x_2^{(1)}$, $x_0^{(1)} = -1$ y usamos la notación $\delta_1^{(2)} = \frac{\partial C}{\partial s_1^{(2)}}$.

Ahora calculamos las derivadas parciales con respecto a los pesos y sesgos con superíndice $\ell = 1$. El costo $C$ depende de $y$, que a su vez depende de $w_{ij}^{(1)}$ a través de $s_j^{(1)}$, $j = 1, 2$, dado que $y = \phi(W^{(2)^t}\phi(\mathbf{s}^{(1)}) - \mathbf{b}^{(2)})$ por (6.1.1). Para $j = 1$ la regla de la

cadena nos da

$$\frac{\partial C}{\partial b_1^{(1)}} = \frac{\partial C}{\partial w_{01}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}}\frac{\partial s_1^{(1)}}{\partial w_{01}^{(1)}} = \delta_1^{(1)} x_0^{(0)} = -\delta_1^{(1)}$$
$$\frac{\partial C}{\partial w_{11}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}}\frac{\partial s_1^{(1)}}{\partial w_{11}^{(1)}} = \delta_1^{(1)} x_1^{(0)}$$
$$\frac{\partial C}{\partial w_{21}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}}\frac{\partial s_1^{(1)}}{\partial w_{21}^{(1)}} = \delta_1^{(1)} x_2^{(0)},$$

donde ahora $\delta_1^{(1)} = \frac{\partial C}{\partial s_1^{(1)}}$. Para $j = 2$ nos queda

$$\frac{\partial C}{\partial b_2^{(1)}} = \frac{\partial C}{\partial w_{02}^{(1)}} = \frac{\partial C}{\partial s_2^{(1)}}\frac{\partial s_2^{(1)}}{\partial w_{02}^{(1)}} = \delta_2^{(1)} x_0^{(0)} = -\delta_2^{(1)}$$
$$\frac{\partial C}{\partial w_{12}^{(1)}} = \frac{\partial C}{\partial s_2^{(1)}}\frac{\partial s_2^{(1)}}{\partial w_{12}^{(1)}} = \delta_2^{(1)} x_1^{(0)}$$
$$\frac{\partial C}{\partial w_{22}^{(1)}} = \frac{\partial C}{\partial s_2^{(1)}}\frac{\partial s_2^{(1)}}{\partial w_{22}^{(1)}} = \delta_2^{(1)} x_2^{(0)},$$

donde $\delta_2^{(1)} = \frac{\partial C}{\partial s_2^{(1)}}$.

Resumiendo estos cálculos, tenemos

$$\frac{\partial C}{\partial b_j^{(\ell)}} = -\delta_j^{(\ell)} \tag{6.1.3}$$

$$\frac{\partial C}{\partial w_{ij}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}. \tag{6.1.4}$$

Es decir que el gradiente de la función de costo, $\nabla C$, lo conocemos si podemos calcular los valores de $\delta_j^{(\ell)}$. Veamos que las deltas de la capa oculta, $\delta_j^{(1)}$ $(j = 1, 2)$, dependen del delta de la capa final $\delta_1^{(2)}$. Usando que $s_1^{(1)}$ afecta a la función de costo $C$ sólo a través de $s_1^{(2)}$, la regla de la cadena nos da

$$\delta_1^{(1)} = \frac{\partial C}{\partial s_1^{(1)}} = \frac{\partial C}{\partial s_1^{(2)}}\frac{\partial s_1^{(2)}}{\partial s_1^{(1)}} = \delta_1^{(2)}\frac{\partial s_1^{(2)}}{\partial s_1^{(1)}}. \tag{6.1.5}$$

Ahora, la señal $s_1^{(2)}$ se escribe como

$$s_1^{(2)} = -w_{01}^{(2)} + w_{11}^{(2)}\phi(s_1^{(1)}) + w_{21}^{(2)}\phi(s_2^{(1)}),$$

entonces sigue que

$$\frac{\partial s_1^{(2)}}{\partial s_1^{(1)}} = w_{11}^{(2)}\phi'(s_1^{(1)}).$$

Reemplazando en (6.1.5), obtenemos

$$\delta_1^{(1)} = \delta_1^{(2)} w_{11}^{(2)}\phi'(s_1^{(1)}).$$

De manera análoga,

$$\delta_2^{(1)} = \delta_1^{(2)} w_{21}^{(2)} \phi'(s_2^{(1)}).$$

Estas fórmulas pueden escribirse vectorialmente como

$$\begin{pmatrix} \delta_1^{(1)} \\ \delta_2^{(1)} \end{pmatrix} = \delta_1^{(2)} \begin{pmatrix} w_{11}^{(2)} \phi'(s_1^{(1)}) \\ w_{21}^{(2)} \phi'(s_2^{(1)}). \end{pmatrix}. \tag{6.1.6}$$

Luego, para encontrar los deltas de la capa oculta, es suficiente calcular el delta de la capa final $\delta_1^{(2)}$. Veamos como realizar ese cálculo.

La función de costo $C(w, b)$ depende de $s_1^{(2)}$ sólo a través de la salida $y = f_{w,b}(\mathbf{x}^{(0)}) = \phi(s_1^{(2)})$, luego aplicando la regla de la cadena,

$$\delta_1^{(2)} = \frac{\partial C}{\partial s_1^{(2)}} = \frac{\partial C}{\partial y} \frac{\partial y}{\partial s_1^{(2)}} = \frac{\partial C}{\partial y} \phi'(s_1^{(2)}).$$

Reemplazando en (6.1.6) obtenemos

$$\begin{pmatrix} \delta_1^{(1)} \\ \delta_2^{(1)} \end{pmatrix} = \frac{\partial C}{\partial y} \phi'(s_1^{(2)}) \begin{pmatrix} w_{11}^{(2)} \phi'(s_1^{(1)}) \\ w_{21}^{(2)} \phi'(s_2^{(1)}). \end{pmatrix}.$$

El factor $\frac{\partial C}{\partial y}$ depende de la función de costo considerada. Por ejemplo para la función de costo $C = \frac{1}{2}(y - z)^2$, tenemos $\frac{\partial C}{\partial y} = y - z$. donde $z$ es la función objetivo que debe ser aprendida por la red. Ya a este pundo hemos calculado todos los deltas.

Finalmente, para minimizar la función de costo $C$, aplicamos el método de descenso de gradiente. La inicialización es dada por una elección arbitraria de pesos y sesgos $(w_{ij}^{(\ell)}(0), b_j^{(\ell)}(0))$, y la sucesión aproximada $(w_{ij}^{(\ell)}(k), b_j^{(\ell)}(k))$ para $k \geq 1$ se define recursivamente por

$$b_j^{(\ell)}(k+1) = b_j^{(\ell)}(k) - \eta \frac{\partial C}{\partial b_j^{(\ell)}}(w_{ij}^{(\ell)}(k), b_j^{(ell)}(k))$$

$$w_{ij}^{(\ell)}(k+1) = w_{ij}^{(\ell)}(k) - \eta \frac{\partial C}{\partial w_{ij}^{(\ell)}}(w_{ij}^{(\ell)}(k), b_j^{(ell)}(k)),$$

donde el argumento $k$ representa el número de iteraciones. Con la notación de deltas introducida, esta recursión se escribe como

$$b_j^{(\ell)}(k+1) = b_j^{(\ell)}(k) + \eta \delta_j^{(\ell)}$$

$$w_{ij}^{(\ell)}(k+1) = w_{ij}^{(\ell)}(k) - \eta \delta_j^{(\ell)} x_i^{(\ell-1)}(k),$$

donde $\eta > 0$ es la tasa de aprendizaje y $x_i^{(\ell-1)}(k)$ es la $i-$ésima entrada en la capa $\ell$ dado por los valores de los pesos y sesgos en la $k-$ésima iteración.

Cuando se han utilizado todos los ejemplos de entrenamiento para actualizar los parámetros de la red, decimos que se ha procesado un ciclo. El proceso de actualización de parámetros continúa hasta que el sistema de aprendizaje se ajusta a los datos de entrenamiento. El descenso de gradiente se utiliza hasta que el algoritmo ha convergido a una solución localmente óptima, lo que se puede verificar comprobando si la norma del gradiente ha convergido. Si el modelo representa perfectamente los datos de entrenamiento, entonces la pérdida de entrenamiento puede hacerse arbitrariamente

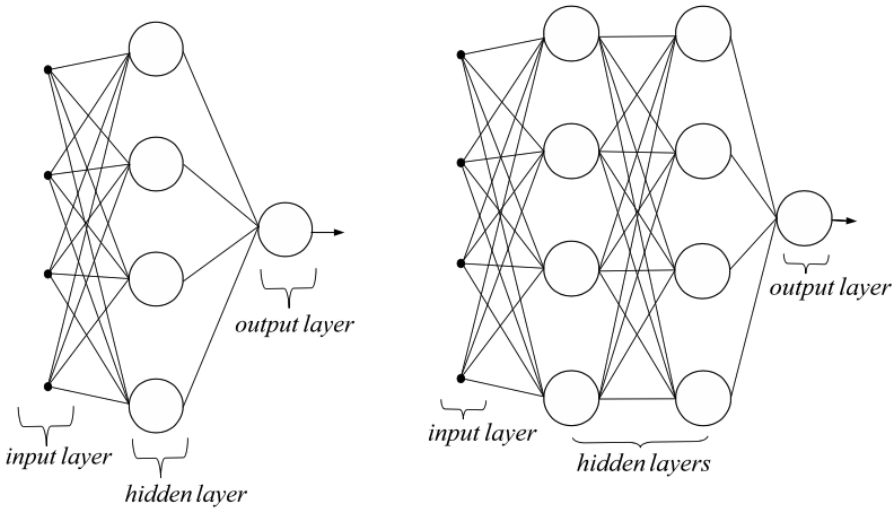



FIGURA 6.4. **a.** Una red neuronal con una capa escodida. **b.** Una red neuronal con dos capas escondidas.

pequeña. Si el modelo no se ajusta adecuadamente a los datos de entrenamiento, la pérdida de entrenamiento no puede disminuir por debajo de un cierto límite.

## 6.2. Redes neuronales generales

A continuación, discutiremos el caso de una red neuronal de *alimentación hacia adelante* (o *feedforward* en inglés), es decir, una red en la que la información fluye desde la entrada hasta la salida. La figura 6.2 **a** muestra el caso de una red neuronal con 3 capas: La capa de entrada que tiene 4 entradas, la capa del medio, o capa escondida que tiene 4 neuronas y la capa de salida con sólo 1 neurona. La Figura 6.2 **b** representa el caso de una red neuronal con 4 capas, que luce similar a la red anterior, pero posee 2 capas escondidas, cada una de ellas con 4 neuronas.

Comenzamos presentando la notación sobre capas, pesos, entradas y salidas de una red neuronal general.

**Capas.** Notaremos el número de la capa con un multiíndice $\ell$. Tenemos $\ell = 0$ para la capa de entrada, $\ell = 1$ para la primera capa escondida y $\ell = L$ para la capa de salida. Notemos que el número de capas escondidas es igual a $L - 1$, El número de neuronas en la capa $\ell$ se nota por $d^{(\ell)}$. En particular, $d^{(0)}$ es el número de entradas y $d^{(L)}$ es el número de salidas.

**Pesos.** El sistema de pesos es denotado por $\{w_{ij}^{(\ell)}\}$, con $1 \leq \ell \leq L$, $0 \leq i \leq d^{(\ell-1)}$, $1 \leq j \leq d^{(\ell)}$. El peso $w_{ij}^{(\ell)}$ se asocia al eje que conecta la $i-$ésima neurona de la capa $\ell-1$ con la $j-$ésima neurona de la capa $\ell$. Notemos que los ejes sólo conectan neuronas de capas consecutivas. Los pesos $w_{0j}^{(\ell)} = b_j^{(\ell)}$ son los sesgos y son los pesos correspondientes a las falsas entradas $x_0^{(\ell)} = -1$. El número de sesgos es igual al número de neuronas.

El número de pesos, sin los sesgos, entre las capas $\ell-1$ y $\ell$ es dado por $d^{(\ell-1)}d^{(\ell)}$, con lo cual el número de pesos en una red de alimentación hacia adelante es

$$\sum_{\ell=1}^{L} d^{(\ell-1)}d^{(\ell)}.$$

Por ejemplo, la red de la figura 6.2 **a** tiene $4\times 4+4\times 1=20$ pesos, mientras que la de la figura 6.2 **b** tiene $4\times 4+4\times 4+4\times 1=36$ pesos. Los pesos pueden ser tanto positivos, como negativos o incluso cero. Típicamente, un peso positivo estimula a la neurona a disparar, mientras que uno negativo la inhibe de hacerlo.

**Entradas y salidas.** Las entradas de la red se denotan por $x_j^{(0)}$, con $1\leq j\leq d^{(0)}$. Notamos por $x_j^{(\ell)}$ a la salida de la $j-$ésima neurona en la capa $\ell$. Luego, $x_j^{(L)}$ es la salida de la red, con $1\leq \ell\leq d^{(L)}$. Reservamos la notación $x_0^{(\ell)}=-1$ para las falsas entradas correspondientes a los sesgos de las neuronas de la capa $\ell+1$.

Consideremos la $j-$ésima neurona de la capa $\ell$. En la primera parte, la neurona recoge la información de la capa previa como un promedio ponderado en la señal

$$s_j^{(\ell)}=\sum_{i=0}^{d^{(\ell-1)}} w_{ij}^{(\ell)}x_i^{(\ell-1)}=\sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)}x_i^{(\ell-1)}-b_j^{(\ell)}.$$

En la segunda parte, la neurona aplica la función de activación $\phi$ a la señal que recogió y produce el valor de salida

$$x_j^{(\ell)}=\phi(s_j^{(\ell)})=\phi\left(\sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)}x_i^{(\ell-1)}-b_j^{(\ell)}\right). \tag{6.2.1}$$

Esto puede escribirse en forma matricial como

$$\mathbf{x}^{(\ell)}=\phi(W^{(\ell)^t}\mathbf{x}^{(\ell-1)}-\mathbf{b}^{(\ell)}),$$

donde

$$\mathbf{x}^{(\ell)}=(x_1^{(\ell)},\ldots,x_{d^{(\ell)}}^{(\ell)})^t,\quad W^{(\ell)}=\left(w_{ij}^{(\ell)}\right)_{i,j},\quad \mathbf{b}^{(\ell)}=(b_1^{(\ell)},\ldots,b_{d^{(\ell)}}^{(\ell)})^t,$$

con $1\leq i\leq d^{(\ell-1)}$ y $1\leq j\leq d^{(\ell)}$ y hemos usado la convención de que la función de activación $\phi$ aplicada a un vector actúa componente a componente. Observemos que la entrada de la red está dada por el vector $\mathbf{x}^{(0)}=(x_1^{(0)},\ldots,x_{d^{(0)}}^{(0)})^t$ mientras que la salida es $\mathbf{x}^{(L)}=(x_1^{(L)},\ldots,x_{d^{(L)}}^{(L)})^t$.

**6.2.1. Pasando a través de la red.** Dado un sistema de pesos y sesgos, uno puede usar la ecuación (6.2.1) para hallar las salidas de cada neurona en función de sus entradas. En particular uno encuentra la salida de la red, que es la salida de la última capa $\mathbf{x}^{(L)}$. Avanzar a través de la red significa calcular todas las salidas de las neuronas. Este paso hacia adelante representa una etapa de predicción del objetivo.

Esto prepara el terreno para la segunda etapa del método de recorrer la red hacia atrás para poder calcular el gradiente de la función de costo y así poder actualizar los parámetros fijados por el método de descenso de gradiente.

Estos pasajes hacia adelante y hacia atrás producen una sucesión de parámetros que corresponden a una sucesión de mejoras en la predicción del objetivo. Este proceso finaliza cuando la función de costo no puede mejorarse.

**6.2.2. Avanzando hacia atrás a través de la red.** Sea $C(w,b)$ la función de costo de la red, que mide la proximidad entre el objetivo $\mathbf{z}$ y la salida $\mathbf{y} = \mathbf{x}^{(L)}$. Para poder aplicar el método de descenso de gradiente, necesitamos poder calcular precisamente el gradiente de $C$, $\nabla C = (\nabla_w C, \nabla_b C)$.

Dado que los pesos $w_{ij}^{(\ell)}$ conectan la $i-$ésima neurona en la capa $\ell-1$ con la $j-$ésima neurona de la capa $\ell$, el mismo va a entrar en el cálculo de la señal $s_j^\ell$ producida por la $j-$ésima neurona de la capa $\ell$. Más aún, ninguna otra señal de la capa $\ell$ se ve afectada por ese peso en particular. Luego, la regla de la cadena nos da

$$\frac{\partial C}{\partial w_{ij}^{(\ell)}} = \frac{\partial C}{\partial s_j^{(\ell)}} \frac{\partial s_j^{(\ell)}}{\partial w_{ij}^{(\ell)}}.$$

Usaremos la notación delta

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial s_j^{(\ell)}}$$

para notar la sensibilidad del error con respecto a la señal $s_j^{(\ell)}$. El segundo factor de la fórmula se calcula fácilmente como

$$\frac{\partial s_j^{(\ell)}}{\partial w_{ij}^{(\ell)}} = x_i^{(\ell-1)},$$

con lo que obtenemos

$$\frac{\partial C}{\partial w_{ij}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}.$$

Haciendo ahora un análisis similar, se calcula la derivada del costo con respecto a los sesgos y obtenemos

$$\frac{\partial C}{\partial b_j^{(\ell)}} = -\delta_j^{(\ell)}.$$

De estas relaciones, obtenemos que el gradiente de $C$ se escribe como

$$\nabla C = (\nabla_w C, \nabla_b C) = \delta_j^{(\ell)}(x_i^{(\ell-1)}, -1),$$

de donde sigue que para el cálculo de $\nabla C$, sólo necesitamos calcular los deltas $\delta_j^{(\ell)}$.

Como vimos en el ejemplo de la sección previa, el cálculo de los deltas se realiza mediante un algoritmo de retropropagación que en el caso general lo discutiremos en la siguiente sección.

**6.2.3. Retropropagación de los deltas.** Empecemos con el cálculo de los deltas en la última capa $\delta_j^{(L)}$, $1 \leq j \leq d^{(L)}$. Esto depende esencialmente de la función de costo. Para simplificar los cálculos, asumamos que la función de costo viene dada por

$$C(w,b) = \frac{1}{2}\sum_{j=1}^{d^{(L)}} (x_j^{(L)} - z_j)^2 = \frac{1}{2}\|\mathbf{x}^{(L)} - \mathbf{z}\|^2.$$

Usando que $x_j^{(L)} = \phi(s_j^{(L)})$, la regla de la cadena implica que

$$\delta_j^{(L)} = \frac{\partial C}{\partial s_j^{(L)}} = (x_j^{(L)} - z_j)\phi'(s_j^{(L)}). \tag{6.2.2}$$

Observemos que para algunas funciones de activación particulares, esta expresión puede simplificarse. Por ejemplo, para la función logística $\phi = \sigma$, se tiene que $\sigma' = \sigma(1-\sigma)$ y para la función tangente hiperbólica $\phi = \mathbf{t}$ se tiene que $\mathbf{t}' = (1 - \mathbf{t}^2)$, lo que evita el cálculo de $\phi'$.

Ahora comienza la retropropagación. El objetivo es entonces calcular los deltas de la capa $\ell - 1$ en función de los deltas de la capa $\ell$. Así, partiendo de la fórmula (6.2.2) para los deltas de la última capa, podemos iterativamente calcular todos los deltas restantes.

Aplicamos una vez más la regla de la cadena y obtenemos

$$\delta_i^{(\ell-1)} = \frac{\partial C}{\partial s_i^{(\ell-1)}} = \sum_{j=1}^{d^{(\ell)}} \frac{\partial C}{\partial s_j^{(\ell)}} \frac{\partial s_j^{(\ell)}}{\partial s_i^{(\ell-1)}} = \sum_{j=1}^{d^{(\ell)}} \delta_j^{(\ell)} \frac{\partial s_j^{(\ell)}}{\partial s_i^{(\ell-1)}}.$$

Debemos entonces calcular la derivada $\frac{\partial s_j^{(\ell)}}{\partial s_i^{(\ell-1)}}$, i.e. la sensibilidad de la señal en la capa $\ell$ con respecto a la señal en la capa previa. Recordemos que

$$s_j^{(\ell)} = \sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)} = \sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} \phi(s_i^{(\ell-1)}) - b_j^{(\ell)},$$

de donde diferenciando con respecto a $s_i^{(\ell-1)}$, obtenemos

$$\frac{\partial s_j^{(\ell)}}{\partial s_i^{(\ell-1)}} = w_{ij}^{(\ell)} \phi'(s_i^{(\ell-1)}).$$

Reemplazando en la expresión para el delta, obtenemos la *fórmula de retropropagación*

$$\delta_i^{(\ell-1)} = \phi'(s_i^{(\ell-1)}) \sum_{j=1}^{d^{(\ell)}} \delta_j^{(\ell)} w_{ij}^{(\ell)}. \tag{6.2.3}$$

En resumen, para calcular el gradiente de la función de costo, uno usa la fórmula de retropropagación (6.2.3) para calcular $\delta_i^{(\ell)}$ y la fórmula de propagación hacia adelante (6.2.1) para el cálculo de las salidas $x_i^{(\ell-1)}$.

**6.2.4. Relaciones finales.** Los cálculos de las secciones previas se pueden resumir en el siguiente conjunto de *ecuaciones maestras*

$$x_j^{(\ell)} = \phi\left(\sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)}\right), \qquad 1 \le j \le d^{(\ell)} \tag{6.2.4}$$

$$\delta_j^{(L)} = (x_j^{(L)} - z_j)\phi'(s_j^{(L)}), \qquad 1 \le j \le d^{(L)} \tag{6.2.5}$$

$$\delta_i^{(\ell-1)} = \phi'(s_i^{(\ell-1)}) \sum_{j=1}^{d^{(\ell)}} \delta_j^{(\ell)} w_{ij}^{(\ell)}, \qquad 1 \le i \le d^{(\ell-1)} \tag{6.2.6}$$

$$\frac{\partial C}{\partial w_{ij}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)} \tag{6.2.7}$$

$$\frac{\partial C}{\partial b_j^{(\ell)}} = -\delta_j^{(\ell)}. \tag{6.2.8}$$

La primera de estas fórmulas nos da una expresión recursiva para el cálculo de las salidas en términos de los pesos y sesgos. La segunda nos da la expresión del delta de la capa de salida (suponiendo que la función de costo es la suma de los cuadrados de los errores). La tercera es la fórmula retropropagación de los deltas. Las últimas dos expresiones nos proveen las componentes del gradiente de $C$ con respecto a los pesos y sesgos.

**6.2.5. Forma matricial.** Para evitar confusiones con los índices es más conveniente reescribir las ecuaciones maestras de la sección anterior en forma matricial.

Para eso necesitamos introducir un nuevo producto entre vectores de la misma dimensión llamado el *producto de Hadamard*. Si $\mathbf{u}$ y $\mathbf{v}$ son vectores de $\mathbb{R}^n$ se define el producto de Hadamard de $\mathbf{u} = (u_1, \ldots, u_n)$ y $\mathbf{v} = (v_1, \ldots, v_n)$ como $\mathbf{u} \odot \mathbf{v} \in \mathbb{R}^n$ al vector con coordenadas $(\mathbf{u} \odot \mathbf{v})_j = u_j v_j$. Es decir es el producto coordenada a coordenada. Su uso simplifica mucho la forma de las ecuaciones.

Usamos las notaciones

$$\mathbf{x}^{(\ell)} = (x_1^{(\ell)}, \ldots, x_{d^{(\ell)}}^{(\ell)})^t, \quad W^{(\ell)} = \left(w_{ij}^{(\ell)}\right)_{i,j}, \quad \mathbf{b}^{(\ell)} = (b_1^{(\ell)}, \ldots, b_{d^{(\ell)}}^{(\ell)})^t,$$

$$\delta^{(\ell)} = (\delta_1^{(\ell)}, \ldots, \delta_{d^{(\ell)}}^{(\ell)})^t, \quad \mathbf{s}^{(\ell)} = (s_1^{(\ell)}, \ldots, s_{d^{(\ell)}}^{(\ell)})^t$$

junto con la convención de que $\phi$ actúa sobre un vector componente a componente, las ecuaciones maestras de la sección anterior se pueden escribir como:

$$\mathbf{x}^{(\ell)} = \phi\left(W^{(\ell)^t} \mathbf{x}^{(\ell-1)} - \mathbf{b}^{(\ell)}\right) \tag{6.2.9}$$

$$\delta^{(L)} = (\mathbf{x}^{(L)} - \mathbf{z}) \odot \phi'(\mathbf{s}^{(L)}) \tag{6.2.10}$$

$$\delta^{(\ell-1)} = (W^{(\ell)} \delta^{(\ell)}) \odot \phi'(\mathbf{s}^{(\ell-1)}) \tag{6.2.11}$$

$$\frac{\partial C}{\partial W^{(\ell)}} = \mathbf{x}^{(\ell-1)} \delta^{(\ell)^t} \tag{6.2.12}$$

$$\frac{\partial C}{\partial \mathbf{b}^{(\ell)}} = -\delta^{(\ell)}. \tag{6.2.13}$$

Notemos que el lado derecho de (6.2.12) es una matriz resultado del producto de dos vectores. Más precisamente, si $\mathbf{u} = (u_1, \ldots, u_n)^t$ y $\mathbf{v} = (v_1, \ldots, v_n)^t$ son dos vectores columna en $\mathbb{R}^n$, entonces $\mathbf{u}\mathbf{v}^t = (u_i v_j)_{i,j}$ es una matriz de $n \times n$. Sin embargo, si se realiza el producto $\mathbf{u}^t\mathbf{v} = \sum_i u_i v_i$ es un escalar. Observemos también que en (6.2.11) aparece $W^{(\ell)}$ y no la matriz transpuesta.

Observación 6.2.1. En el caso de una neurona lineal, las fórmulas previas pueden simplificarse. Dado que la función de activación es la identidad $\phi(x) = x$, la fórmula de retropropagación para los delta se simplifica a $\delta^{(\ell-1)} = W^{(\ell)}\delta^{(\ell)}$. Iterando esta expresión, obtenemos

$$\delta^{(\ell)} = W^{(\ell+1)}W^{(\ell+2)} \cdots W^{(L)}\delta^{(L)}.$$

Luego, obtenemos las fórmulas

$$\frac{\partial C}{\partial W^{(\ell)}} = \mathbf{x}^{(\ell-1)}(W^{(\ell+1)}W^{(\ell+2)} \cdots W^{(L)}\delta^{(L)})^t$$
$$\frac{\partial C}{\partial \mathbf{b}^{(\ell)}} = -W^{(\ell+1)}W^{(\ell+2)} \cdots W^{(L)}\delta^{(L)},$$

con lo que se obtiene una expresión explícita para el gradiente de la función de costo.

Observación 6.2.2. La fórmula (6.2.10) está basada en la elección particular de la función de costo. En caso de que la función de costo se cambie, *solamente* hay que modificar la fórmula para $\delta^{(L)}$ que hay que ajustar acorde a ese cambio. Veamos un ejemplo con otra función de costo.

Ejemplo 6.2.3. Supongamos que la función de costo es dada por la entropía cruzada

$$C = -\sum_k z_k \ln x_k^{(L)} = -\sum_k z_k \ln \phi(s_k^{(L)}),$$

que representa la ineficiencia de usar la salida predecida $\mathbf{x}^{(L)}$ en vez de el dato real $\mathbf{z}$. El delta de la $j-$ésima neurona en la capa de salida es dada por

$$\delta_j^{(L)} = \frac{\partial C}{\partial s_j^{(\ell)}} = -\frac{\partial}{\partial s_j^{(\ell)}} \sum_k z_k \ln \phi(s_k^{(L)}) = -z_j \frac{\phi'(s_j^{(L)})}{\phi(s_j^{(L)})}.$$

Esta expresión puede simplificarse aún más para algunas funciones de activación. Supongamos que $\phi(x) = \sigma(x)$, entonces en este caso se tiene

$$\delta_j^{(L)} = -z_j \frac{\sigma'(s_j^{(L)})}{\sigma(s_j^{(L)})} = -z_j \frac{\sigma(s_j^{(L)})(1 - \sigma(s_j^{(L)}))}{\sigma(s_j^{(L)})} = -z_j(1 - \sigma(s_j^{(L)})),$$

que puede escribirse vectorialmente como

$$\delta^{(L)} = -\mathbf{z} \odot (1 - \sigma(\mathbf{s}^{(L)})).$$

Esta fórmula reemplaza (6.2.10) en este caso.

**6.2.6. El algoritmo de descenso de gradiente.** Consideremos una red neuronal con un sistema de pesos y sesgos iniciales dados por $w_{ij}^{(\ell)}(0)$ y $b_j^{(\ell)}(0)$. Estos pesos y sesgos pueden inicializarse eficientemente con un procedimiento que describiremos más adelante. Sea $\eta > 0$ la tasa de aprendizaje. El valor óptimo de los pesos y sesgos luego de que la red es entrenada con el algoritmo de descenso de gradiente están dados por la sucesión aproximante definida por

$$\begin{aligned} w_{ij}^{(\ell)}(k+1) &= w_{ij}^{(\ell)}(k) - \eta \delta_j^{(\ell)}(k) x_i^{(\ell-1)}(k) \\ b_j^{(\ell)}(k+1) &= b_j^{(\ell)}(k) + \eta \delta_j^{(\ell)}(k), \end{aligned}$$

donde las salidas $x_i^{(\ell)}(k)$ y los deltas $\delta_j^{(\ell)}(k)$ dependen de los pesos $w_{ij}^{(\ell)}(k)$ y los sesgos $b_j^{(\ell)}(k)$ obtenidos en el $k-$ésimo paso.

**6.2.7. Definición formal de redes neuronales feedforward.** En esta sección daremos la definición matemática rigurosa de una red neuronal de propagación hacia adelante, o red neuronal *feedforward* (por su terminología en inglés).

Empecemos repasando los ingredientes usados hasta ahora en la construcción informal que hemos hecho. Hemos visto que una red neuronal feedforward contiene neuronas arregladas en $L+1$ capas distintas. La primera de esas capas corresponde a los datos de entrada se etiqueta con el superíndice $\ell = 0$ y la capa de salida se etiqueta con $\ell = L$. La $j-$ésima neurona en la capa $\ell$ tiene activación $x_j^{(\ell)}$. Para la capa de entrada, $x_j^{(0)}$, las activaciones vienen como los datos de entrada, mientras que para las otras capas las activaciones se calculan en términos de las activaciones de las previas capas por la fórmula de propagación hacia adelante

$$x_j^{(\ell)} = \phi^{(\ell)} \left( \sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)} \right),$$

donde $\phi^{(\ell)}$ es la función de activación de la capa $\ell$, $w_{ij}^{(\ell)}$ y $b_j^{(\ell)}$ son los pesos y sesgos de las neuronas en la capa $\ell$ respectivamente. Notemos también que la señal $s_j^{(\ell)} = \sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)}$ es una función afín de $x_i^{(\ell-1)}$ y que la activación $x_j^{(\ell)}$ se obtiene de aplicar la función $\phi^{(\ell)}$ a la señal $s_j^{(\ell)}$.

Realicemos ahora un nivel adicional de abstracción. Notemos por $\mathcal{F}(U) = \{f \colon U \to \mathbb{R}\}$ el conjunto de todas las funciones a valores reales definidas sobre el conjunto $U$. Consideremos $U_\ell = \{1, \ldots, d^{(\ell)}\}$ y definimos la función afín

$$\alpha_\ell \colon \mathcal{F}(U_{\ell-1}) \to \mathcal{F}(U_\ell)$$

como $\alpha_\ell(\mathbf{x}^{(\ell-1)}) = \mathbf{s}^{(\ell)}$. Luego $\mathbf{x}^{(\ell)} = (\phi^{(\ell)} \circ \alpha_\ell)(\mathbf{x}^{(\ell-1)})$ se obtiene como la composición de la función afín en la capa de activación previa, con la nolinealidad $\phi^{(\ell)}$.

La definición formal de una red neuronal feedforward es entonces la siguiente:

Definición 6.2.4. Consideremos $d^\ell \in \mathbb{N}$ con $\ell = 0, \ldots, L$ enteros positivos y $U_\ell = \{1, \ldots, d^{(\ell)}\}$. Sean $\alpha_1, \ldots, \alpha_L$ funciones afines

$$\alpha_\ell \colon \mathcal{F}(U_{\ell-1}) \to \mathcal{F}(U_\ell)$$

y sean $\phi^{(\ell)}\colon \mathbb{R} \to \mathbb{R}$ funciones de activación. Entonces la correspondiente red neuronal feerforward es la sucesión de mapas $f_0, f_1, \ldots, f_L$ dados por

$$f_\ell = \phi^{(\ell)} \circ \alpha_\ell \circ f_{\ell-1}, \quad 1 \leq \ell \leq L,$$

donde $f_0$ es dada.

Es decir, una red neuronal feedforward produce una sucesión de reparametrizaciones cada vez más abstractas $f_\ell$ que mapean la entrada $f_0$ a través de una seria de funciones paramétricas $\alpha_\ell$ y funciones de activación no lineales $\phi^{(\ell)}$. La salida de la red es dada por

$$f_L = \phi^{(L)} \circ \alpha_L \circ \phi^{(L-1)} \circ \alpha_{L-1} \circ \cdots \circ \phi^{(1)} \circ \alpha_1 \circ f_0.$$

El número $L$ se lo denomina la *profundidad* de la red y a máx$\{d^{(1)}, \ldots, d^{(L)}\}$ se lo llama anchura de la red.

Para problemas de regresión, la última función de activación es lineal $\phi^{(L)}(x) = x$. Para problemas de clasificación, se suele tomar la función softmax, definida como

$$\phi^{(L)}(\mathbf{x}) = \text{softmax}(\mathbf{x}) = \frac{1}{\sum_{i=1}^n e^{x_i}}(e^{x_1}, \ldots, e^{x_n}),$$

y en los casos de regresión logística la más usual es la sigmoideal $\phi^{(L)}(x) = \sigma(x)$.

## 6.3. Inicialización de los pesos

En el caso de redes con pocas capas, inicializar los pesos y sesgos fijándolos todos como 0 o sampleándolos con una densidad uniforme o Gaussiana de media cero suele dar resultados satisfactorios. Sin embargo, en el caso de redes neuronales profundas con un gran número de capas es necesario inicializar los pesos con cuidado dado que ese proceso tiene mucha importancia en el funcionamiento del algoritmo de minimización.

La magnitud de los pesos juega un rol importante en evitar tanto como sea posible el problema de gradiente pequeño o muy grande. Esto será descripto en las siguientes dos situaciones marginales.

1. *Si los pesos están muy cerca de cero, entonces la varianza de la entrada decrece mientras pasa por cada capa de la red.*

   Este enunciado heurístico está basado en el siguiente cálculo: sea $x_j^{(\ell)}$ la salida de la $j-$ésima neurona en la capa $\ell$. Las salidas entre dos capas consecutivas está dado por la ecuación de propagación

   $$x_j^{(\ell)} = \phi\left(\sum_i w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)}\right),$$

   Queremos ahora calcular la varianza de esta salida, $\mathbb{V}[x_j^{(\ell)}]$.

   Para este cálculo, observemos que si $X$ es una variable aleatoria y llamemos $m = \mathbb{E}[X]$. Si $f$ es una función de clase $C^2$, entonces tenemos que

   $$f(x) = f(m) + f'(m)(x - m) + O(|x - m|^2),$$

   de donde obtenemos la aproximación

   $$f(X) \simeq f(m) + f'(m)(X - m).$$

Tomando varianza a ambos lados llegamos a la fórmula

$$\mathbb{V}[f(X)] \simeq f'(m)^2 \mathbb{V}[X] \tag{6.3.1}$$

Usando ahora esta aproximación para la varianza y asumiendo que las variables son independencia de las variables, calculamos

$$\mathbb{V}[x_j^{(\ell)}] \simeq \left[\phi'\left(\sum_i w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)}\right)\right]^2 \sum_i w_{ij}^{(\ell)} \mathbb{V}[x_i^{(\ell-1)}].$$

Esta expresión nos da la varianza de las neuronas en la capa $\ell$ en término de las varianzas de las neuronas en la capa previa.

Ahora realicemos la siguiente estimación. Asumamos primero que la función de activación tiene derivada acotada, i.e. $|\phi'|^2 \leq C$ y usando la desigualdad de Cauchy obtenemos

$$\mathbb{V}[x_j^{(\ell)}] \leq C \sum_i w_{ij}^{(\ell)} \mathbb{V}[x_i^{(\ell-1)}] \leq C \left(\sum_i (w_{ij}^{(\ell)})^2\right)^{1/2} \left(\sum_i (\mathbb{V}[x_i^{(\ell-1)}])^2\right)^{1/2}.$$

Finalmente, elevamos al cuadrado y sumamos sobre $j$ para obtener

$$\sum_j (\mathbb{V}[x_j^{(\ell)}])^2 \leq C^2 \sum_{i,j} (w_{ij}^{(\ell)})^2 \sum_i (\mathbb{V}[x_i^{(\ell-1)}])^2.$$

Esto muestra que si los pesos son pequeños, entonces la varianza de la capa $\ell$ es más pequeña que la de la capa previa. Iterando la desigualdad, sigue que la varianza de la señal decae a 0 luego de pasar por un cierto número de capas con lo que la variación de la señal se vuelve insignificante.

2. *Si los pesos son muy grandes, entonces la varianza de la señal tiende a amplificarse cuando pasa por las capas de la red, o cuando la red se aproxima a una zona de plateau.*

   En este caso, si los pesos $w_{ij}^{(\ell)}$ son grandes, entonces la suma $\sum_i w_{ij}^{(\ell)} x_i^{(\ell-1)}$ también es grande.

   Supongamos que la función de activación es lineal, $\phi(x) = x$, entonces

   $$\mathbb{V}[x_j^{(\ell)}] = \sum_i w_{ij}^{(\ell)} \mathbb{V}[x_i^{(\ell-1)}],$$

   lo que implica que la varianza $\mathbb{V}[x_i^{(\ell)}]$ también se vuelve grande.

   Supongamos que la función de activación es sigmoide, entonces en ese caso la función $\sigma$ se vuelve saturada y por ende $\sigma'$ es muy pequeño, lo que trae el problema de gradiente cercano a cero.

Luego, para que la inicialización sea viable, es necesario que los pesos no sean ni muy grandes ni muy pequeños. Necesitamos que la inicialización sea en un rango *razonable* antes de iniciar el entrenamiento de la red. Veremos ahora cómo tratar con este problema.

La idea principal se basa en el hecho de que la propagación del error de la señal a través de una red neuronal profunda puede cuantificarse mediante la varianza de la señal, es decir, la varianza de las activaciones de las neuronas. Para mantener la varianza bajo control, evitando valores que se disparen o se anulen, la forma más simple es

encontrar valores de los pesos para los cuales la varianza se mantenga aproximadamente constante al pasar la señal por cada capa. Dado que la asignación de los pesos iniciales es un proceso aleatorio, asumimos que los pesos son variables aleatorias con distribución uniforme o Gaussiana y media cero.

Simplifiquemos la notación levemente. Fijemos una capa $\ell$ y notamos al vector de entrada de esa capa como $X = X^{(\ell-1)}$ (usamos la notación de letras mayúsculas dado que serán todos variables aleatorias) y al vector de salida de la capa como $Y = X^{(\ell)}$. El número de neuronas lo notaremos como $N = d^{(\ell-1)}$. La $j-$ésima componente de $Y$ está dada por

$$Y_j = \phi\left(\sum_i W_{ij}X_i - b_j\right),$$

donde la matriz de pesos $W_{ij}$ es una matriz aleatoria con media cero. Sobre los $X_i$ asumimos que son independientes, idénticamente distribuidos y con media 0 (por ejemplo pueden ser Gaussianos o con distribución uniforme). Asumamos también que $W_{ij}$ y $X_i$ son independientes. Sobre los sesgos $b_j$ asumimos que son constantes (usualmente se inicializan como 0).

Ahora, usando la fórmula de aproximación para la varianza dada por (6.3.1), obtenemos

$$\begin{aligned}\mathbb{V}[Y_j] &\simeq \phi'\left(\sum_i \mathbb{E}[W_{ij}X_i] - b_j\right)^2 \mathbb{V}\left[\sum_i W_{ij}X_i\right] \\ &= \phi'(b_j)^2 \sum_i \mathbb{V}[W_{ij}X_i] = \phi'(b_j)^2 \sum_i \mathbb{V}[W_{ij}]\mathbb{V}[X_i] \\ &= N\phi'(0)^2\mathbb{V}[W_{11}]\mathbb{V}[X_1],\end{aligned}$$

donde hemos usado que $\mathbb{E}[W_{ij}X_i] = \mathbb{E}[W_{ij}]\mathbb{E}[X_i] = 0$ por la independencia de las variables en la segunda igualdad, que $\mathbb{V}[WX] = \mathbb{V}[W]\mathbb{V}[X]$ para variables aleatorias independientes con media cero en la tercer igualdad y que las variables aleatorias son idénticamente distribuidas (y tomamos $b_j = 0$) para la última igualdad.

Si requerimos que la varianza no se modifique en el paso de capa a capa, esto equivale a pedir que $\mathbb{V}[Y_j] \simeq \mathbb{V}[X_i]$, luego obtenemos

$$N\phi'(0)^2\mathbb{V}[W_{11}]\mathbb{V}[X_1] \simeq \mathbb{V}[X_1].$$

Simplificando obtenemos la expresión para la varianza de los pesos

$$\mathbb{V}[W_{ij}] = \mathbb{V}[W_{11}] \simeq \frac{1}{N\phi'(0)^2}.$$

Tenemos dos casos importantes

1. Los pesos tienen una distribución normal $W_{ij} \sim N(0, \frac{1}{N\phi'(0)^2})$. En particular, si la función de activación es lineal, $\phi'(0) = 1$ y en ese caso $W_{ij} \sim N(0, \frac{1}{N})$.
2. Los pesos tienen una distribución uniforme $W_{ij} \sim U([-a, a])$ para $a > 0$. En este caso, igualando las varianzas obtenemos

$$\frac{1}{N\phi'(0)^2} = \frac{a^2}{3}, \quad \text{de donde} \quad a = \frac{\sqrt{3}}{\phi'(0)\sqrt{N}}.$$

Notemos que la pendiente de la función de activación en el origen afecta el rango del intervalo en donde debemos tomar los valores de los pesos. Mientras más grande sea esa pendiente, más angosto debe ser ese intervalo. Una forma de utilizar este hecho para obtener mejores resultados, es fijar $\phi'(0)$ como un nuevo hiperparámetro y luego ajustarlo en un conjunto de validación.

## 6.4. Resumen

Una red neuronal es obtenida por una concatenación de varias capas de neuronas. La primera capa corresponde con las entradas y la última con la salida. Las capas intermedias se llaman capas ocultas. Este capítulo trató sólo de las llamadas *redes feedforward*. Estas son redes en donde la información fluye hacia adelante desde la entrada a la salida.

Las redes neuronales son usadas para aprender características complejas, que una única neurona no es capaz de hacer. Las redes son entrenadas por el método de descenso de gradiente. Un ingrediente importante de este método es el cálculo del gradiente de la función de costo. Este gradiente es calculado por un método recursivo llamado retropropagación. Las ecuaciones maestras tanto para la propagación hacia adelante, como para la retropropagación son analizadas en detalle.

Para que el método de descenso de gradiente funcione adecuadamente, es necesario dar valores iniciales a los pesos y sesgos. Si la red es estrecha, se suelen inicializar por cero, pero cuando la red es profunda, es necesario desarrollar ciertas estrategias para esa inicialización.

## 6.5. Ejercicios

Ejercicio 6.5.1. Dibujar el esquema de la red dada por la Observación 6.1.2.

Ejercicio 6.5.2. (a) ¿Puede un único perceptrón aprender el siguiente mapa:

$$(0,0,0) \mapsto 1, \quad (1,0,0) \mapsto 0, \quad (0,1,0) \mapsto 0, \quad (0,0,1) \mapsto 0,$$
$$(0,1,1) \mapsto 1, \quad (1,1,0) \mapsto 0, \quad (1,0,1) \mapsto 0, \quad (1,1,1) \mapsto 1?$$

(b) Dibujar el esquema de una red neuronal con perceptrones que aprenda el mapa del ítem anterior. Dar los pesos y sesgos de todos los perceptrones de la red.

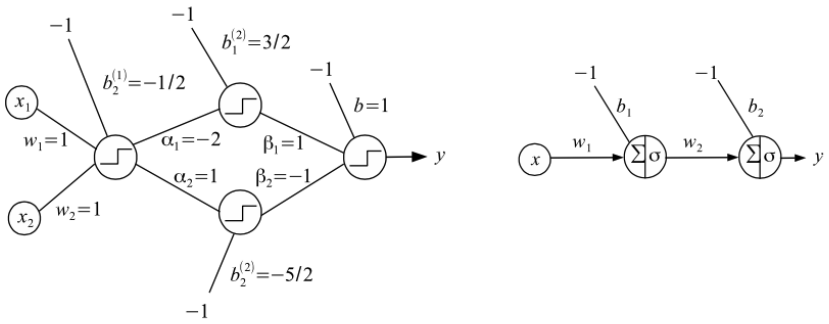
Ejercicio 6.5.3. Escribir una fórmula explícita para la salida de la red neuronal multi-perceptrón dada por la figura 6.5 **a**.

Ejercicio 6.5.4. Considerar una neurona sigmoide con una entrada unidimensional $x$, peso $w$, sesgo $b$ y salida $y = \sigma(wx + b)$. El objetivo es una variable unidimensional $z$. Consideremos la función de costo $C(w, b) = \frac{1}{2}(y - z)^2$.

(a) Hallar $\nabla C(w, b)$ y mostrar que $\|\nabla C(w, b)\| < \frac{1}{4}\sqrt{1 + x^2}(1 + |z|)$.
(b) Escribir la iteración de descenso de gradiente para la sucesión de pesos $(w_n, b_n)$.

Ejercicio 6.5.5. Consideremos la red neuronal dada por la figura 6.5 **b**, con la función de costo dada por $C(w, b) = \frac{1}{2}(y - z)^2$.

(a) Calcular las deltas $\delta^{(\ell)} = \frac{\partial C}{\partial s^{(\ell)}}$, $\ell = 1, 2$.
(b) Usar el algoritmo de retropropagación para hallar el gradiente $\nabla C(w, b)$.




**a** **b**

FIGURA 6.5. **a.** La red neuronal usada en le Ejercicio 6.5.3. **b.** La red neuronal usada en el Ejercicio 6.5.5.

EJERCICIO 6.5.6. Asumamos que la función de activación en la capa $\ell$ es lineal y que los pesos y entradas son variables aleatorias.

(a) Mostrar que $\delta_j^{(\ell)}$ y $W_{ij}^{(\ell)}$ son variables aleatorias independientes.

(b) Mostrar que $\delta_j^{(\ell)}$ y $X_i^{(\ell-1)}$ son variables aleatorias independientes.

EJERCICIO 6.5.7. Considerar una neurona sigmoide con entrada aleatoria normalmente distribuida, $X \sim N(0,1)$ y con salida $Y = \sigma(wX + b)$. Mostrar que $\mathbb{V}[Y] \simeq w^2\sigma'(b)^2$. Notar que la varianza de la salida decrece para valores pequeños del peso $w$ o grandes valores del sesgo $b$.

EJERCICIO 6.5.8. Considerar una red neuronal con una capa escondida que posee neuronas sigmoides en esa capa. Suponer que las entradas están normalmente distribuidas, $X \sim N(0,1)$ y que la salida es $Y = \sum_{i=1}^{N} \alpha_i \sigma(w_i X + b_i)$. Mostrar que $\mathbb{V}[Y] \simeq \sum_j \sigma'(b_i)^2 \alpha_i^2 w_i^2$.

# Parte 2

# Teoría analítica

Capítulo 7

# Teoremas de aproximación

Este capítulo presenta algunos resultados clásicos del análisis real con aplicaciones al aprendizaje de funciones continuas, integrables o de cuadrado integrables. Los resultados de aproximación incluidos en este capítulo abarcan el teorema de Dini, el teorema de Arzelà-Ascoli, el teorema de Stone-Weierstrass, el teorema tauberiano de Wiener y el principio de contracción. Algunas de sus aplicaciones al aprendizaje se proporcionarán dentro de este capítulo, mientras que otras se presentarán en capítulos posteriores.

## 7.1. El Teorema de Dini

Aprender una función continua en un intervalo compacto se reduce a demostrar que la red neuronal produce una secuencia de funciones continuas que converge uniformemente a la función objetivo continua. En la mayoría de los casos, no es difícil construir resultados que converjan puntualmente a la función objetivo. Sin embargo, demostrar la convergencia uniforme puede ser, a veces, una tarea complicada. En ciertas circunstancias, esto puede simplificarse mediante el uso de algunos resultados de aproximación, como los siguientes.

TEOREMA 7.1.1 (Dini). *Sea $f_n\colon [a,b] \to \mathbb{R}$ una sucesión de funciones continuas.*

(a) *Si $f_{n+1} \leq f_n$ para $n \geq 1$ y $f_n(x) \to 0$ puntualmente para $x \in [a,b]$, entonces $f_n$ converge uniformemente a 0 en $[a,b]$.*
(b) *Si $g \in C([a,b])$ es tal que $f_n(x) \searrow g(x)$ puntualmente para $x \in [a,b]$, entonces $f_n$ converge uniformemente a $g$ en $[a,b]$.*

DEMOSTRACIÓN. Obviamente (b) es una consecuencia inmediata de (a). Veamos la demostración de (a).

Sea $\varepsilon > 0$ y consideremos los conjuntos

$$A_n(\varepsilon) = f_n^{-1}([\varepsilon, \infty)) = \{x \in [a,b]\colon f_n(x) \geq \varepsilon\}.$$

Lo que queremos ver, que $f_n \rightrightarrows 0$, es equivalente a probar que existe $N \in \mathbb{N}$ tal que $A_n(\varepsilon) = \emptyset$ para $n \geq N$.

Como las funciones $f_n$ son continuas, entonces los conjuntos $A_n(\varepsilon)$ son cerrados. Más aún, como $A_n(\varepsilon) \subset [a,b]$, resultan compactos. Adicionalmente, como $\{f_n\}_{n\in\mathbb{N}}$ es decreciente, los conjuntos $A_n(\varepsilon)$ están encajados, i.e. $A_{n+1}(\varepsilon) \subset A_n(\varepsilon)$ para $n \in \mathbb{N}$. Supongamos que $A_n(\varepsilon) \neq \emptyset$ para todo $n \geq 1$. Entonces, tenemos que $\cap_{n\geq 1} A_n(\varepsilon) \neq \emptyset$ por ser la intersección de una sucesión de conjuntos compactos encajados. Pero si $x_0 \in \cap_{n\geq 1} A_n(\varepsilon)$, entonces $f_n(x_0) \geq \varepsilon$ para todo $n \in \mathbb{N}$ lo que contradice el hecho de que $f_n(x_0) \to 0$ cuando $n \to \infty$. □

OBSERVACIÓN 7.1.2. El Teorema de Dini sigue siendo cierto si se reemplaza el intervalo $[a, b]$ por un espacio métrico compacto $(S, d)$ y la convergencia es considerada respecto de la métrica $d$.

EJEMPLO 7.1.3. Sea $g\colon [0,1] \to \mathbb{R}$ de clase $C^2$ y consideremos la partición $\Delta_1$ dada por $0 = x_0 < x_1 < \cdots < x_n = 1$ donde $x_i$ son puntos de inflexión de $g$. Es decir, $g|_{[x_{i-1},x_i]}$ es cóncava o convexa para todo $i = 1, \ldots, n$. Notemos por $f_1$ a la función lineal a trozos que se obtiene de unir a los puntos $(x_i, g(x_i))$, $i = 0, \ldots, n$.

Consideremos ahora la partición $\Delta_2$ que consiste en agregar a $\Delta_1$ los puntos medios $(x_i + x_{i+1})/2$, y notemos a la nueva función lineal a trozos que une los puntos dados por la partición $\Delta_2$ como $f_2$.

Si continuamos con este proceso, obtenemos una sucesión de particiones $\Delta_n$ y una sucesión de funciones lineales a trozos $f_n$. Esta sucesión, verifica la propiedad

$$|g(x) - f_{n+1}(x)| \leq |g(x) - f_n(x)|, \quad \forall x \in [0,1].$$

El Teorema de Dini, Teorema 7.1.1, aplicado a la sucesión $|g - f_n|$ implica que la sucesión $f_n$ converge a $g$ uniformemente en $[0, 1]$.

## 7.2. Teorema de Arzela-Ascoli

Otro resultado para construir una sucesión que converja uniformemente, es extraer de una sucesión dada, una subsucesión que tenga esa propiedad. El Teorema de Arzela-Ascoli nos provee de este resultado de existencia. Al ser un resultado teórico y no constructivo, tiene implicancias importantes teóricas y no en la construcción de algoritmos.

Necesitamos unas definiciones para poder enunciar el teorema. Una familia de funciones $\mathcal{F}$ definidas sobre un conjunto $A$ a valores reales se dice *uniformemente acotada* si existe $M > 0$ tal que

$$|f(x)| \leq M, \qquad \forall x \in A, \quad \forall f \in \mathcal{F}.$$

EJEMPLO 7.2.1. Por ejemplo, la familia

$$\mathcal{F} = \left\{ \sum_{j=1}^{N} \alpha_j \sigma(w_j x + b_j) \colon \alpha_j, w_j, b_j \in \mathbb{R},\ \sum_{j=1}^{N} \alpha_j^2 \leq 1 \right\},$$

donde $\sigma(x)$ es una función sigmoidea tal que $|\sigma(x)| \leq 1$ resulta ser uniformemente acotada.

En efecto, si llamamos $\alpha = (\alpha_1, \ldots, \alpha_N)$ y $\bar{\sigma}(x) = (\sigma(w_1 x + b_1), \ldots, \sigma(w_N x + b_N))$, usando la desigualdad de Cauchy-Schwartz, tenemos

$$\left| \sum_{j=1}^{N} \alpha_j \sigma(w_j x + b_j) \right| = |\alpha \cdot \bar{\sigma}(x)| \leq \|\alpha\| \|\bar{\sigma}(x)\| \leq 1 \cdot \sqrt{N} = \sqrt{N}.$$

Una familia de funciones $\mathcal{F}$ definidas en un conjunto $A \subset \mathbb{R}^n$ se dice *equicontinua*, si para todo $\varepsilon > 0$, existe $\delta > 0$ tal que para toda $f \in \mathcal{F}$ se verifica que

$$|f(x) - f(y)| < \varepsilon, \quad \forall x, y \in A \text{ con } |x - y| < \delta.$$

El ejemplo más importante de una familia de funciones equicontinua es un conjunto de funciones de clase $C^1$ con derivada acotada. En efecto, supongamos que tenemos una familia $\mathcal{F} \subset C^1([a,b])$ y que existe $L > 0$ tal que

$$\sup_{x\in[a,b]} |f'(x)| \leq L, \qquad \forall f \in \mathcal{F}.$$

Luego, usando el Teorema del valor medio, obtenemos

$$|f(x) - f(y)| = |f'(\xi)(x-y)| \leq \sup_{\xi\in[a,b]} |f'(\xi)||x-y| \leq L|x-y|,$$

de donde tomando $\delta = \varepsilon/L$ obtenemos lo deseado.

Ejemplo 7.2.2. Consideremos la familia

$$\mathcal{F} = \left\{ \sum_{j=1}^{N} \alpha_j \sigma(w_j x + b_j) \colon \alpha_j, w_j, b_j \in \mathbb{R},\ \sum_{j=1}^{N} (\alpha_j^2 + w_j^2) \leq 1 \right\},$$

donde $\sigma$ verifica que $|\sigma'| \leq \lambda$. Entonces $\mathcal{F}$ es equicontinua. En efecto, si $f \in \mathcal{F}$, tenemos que

$$\begin{aligned} |f'(x)| &= \left| \sum_{j=1}^{N} \alpha_j w_j \sigma'(w_j x + b_j) \right| \leq \lambda \sum |\alpha_j||w_j| \\ &\leq \lambda \sum_{j=1}^{N} \frac{\alpha_j^2 + w_j^2}{2} \leq \frac{\lambda}{2}. \end{aligned}$$

Luego, por lo visto en el párrafo anterior, $\mathcal{F}$ resulta equicontinua.

Tenemos entonces el teorema

Teorema 7.2.3 (Arzela-Ascoli). *Sea $K \subset \mathbb{R}^n$ un conjunto compacto y $\mathcal{F} \subset C(K)$ una familia de funciones continuas. Son equivalentes:*

(a) *Toda sucesión $\{f_n\}_{n\in\mathbb{N}} \subset \mathcal{F}$ contiene una subsucesión $\{f_{n_k}\}_{k\in\mathbb{N}} \subset \{f_n\}_{n\in\mathbb{N}}$ que es uniformemente convergente.*
(b) *La familia $\mathcal{F}$ es equicontinua y uniformemente acotada.*

La demostración de este teorema se suele dar tanto en el curso de Cálculo Avanzado como en el de Análisis Avanzado y lo pueden encontrar en el libro **[20]**.

Como aplicación del Teorema de Arzela-Ascoli, veamos un resultado de existencia para aprendizaje no supervisado. Una red neuronal de tres capas tiene la capacidad de *potencialmente aprender* una función continua, dada una restricción en las condiciones del tamaño de la capa oculta, la pendiente de la función de activación y el tamaño de los pesos.

Teorema 7.2.4. *Sea $N \geq 1$ un entero fijo y consideremos una red neuronal con una capa oculta de manera tal que se verifican:*

(a) *La entrada de la red es una variable real acotada $x \in [a,b]$.*
(b) *La salida de la red es una neurona unidimensional con una función de activación lineal y sin sesgo.*
(c) *Tenemos $N$ neuronas en la capa oculta con una función de activación $\sigma$ que es derivable y tal que $|\sigma'| \leq \lambda$.*

(d) *Los pesos satisface la condición* $\sum_{j=1}^{N}(\alpha_j^2 + w_j^2) \leq 1$*, donde* $w_j$ *son los pesos de entrada a la capa oculta y* $\alpha_j$ *son los pesos que conectan la capa oculta con la salida.*

*Entonces existe una función continua* $g$ *definida en el intervalo* $[a, b]$ *que puede ser aproximadamente representada por la red.*

DEMOSTRACIÓN. La función de salida de la red viene dada por la suma

$$f_{\alpha,w,b}(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j x + b_j),$$

donde $w_j$ son los pesos de la entrada a la capa oculta y $\alpha_j$ son los pesos de la capa oculta a la salida. Los sesgos son notados por $b_j$. La restricción sobre los pesos del ítem (d), implican que la familia $\mathcal{F} = \{f_{\alpha,w,b} \colon \alpha, w, b \in \mathbb{R}^N\}$ es uniformemente acotada (ver el Ejemplo 7.2.1). Por lo visto en el Ejemplo 7.2.2, la familia $\mathcal{F}$ resulta también equicontinua.

Luego, podemos aplicar el Teorema de Arzela-Ascoli y concluir que toda sucesión de salidas $\{f_n\}_{n\in\mathbb{N}} \subset \mathcal{F}$ dada por

$$f_n(x) = \sum_{j=1}^{N} \alpha_j(n) \sigma(w_j(n) x + b_j(n)),$$

posee una subsucesión $\{f_{n_k}\}_{k\in\mathbb{N}} \subset \{f_n\}_{n\in\mathbb{N}}$ que es uniformente convergente en $[a, b]$ y su límite es una cierta función continua $g \in C([a, b])$. □

OBSERVACIÓN 7.2.5. En el resultado anterior, la función continua $g$ no es conocida a priori. La red neuronal aprende *potencialmente* una función continua, en el sentido de que, entre cualquier sucesión infinita de intentos de entrenamiento de la red, se puede extraer una subsuseción a lo largo de la cual la red aproxima alguna función continua. Esto es un resultado de existencia; encontrar un algoritmo para construir la subsuesión convergente es un problema diferente y será tratado en otro capítulo.

EJEMPLO 7.2.6 (Neurona sigmoidal). Consideremos una neurona sigmoidal con salida

$$f_{\mathbf{w},b}(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b),$$

donde $\sigma$ es la función logística, con pesos $\mathbf{w} \in \mathbb{R}^n$, sesgo $b \in \mathbb{R}$ y entrada $\mathbf{x} \in I_n = [0, 1] \times \cdots \times [0, 1] \subset \mathbb{R}^n$.

Asumamos que los pesos son acotados con $\|\mathbf{w}\| \leq 1$. Entonces la familia de funciones continuas

$$\mathcal{F} = \{f_{\mathbf{w},b} \colon \|\mathbf{w}\| \leq 1,\ b \in \mathbb{R}\},$$

es uniformemente acotada (pues $\sup |f_{\mathbf{w},b}| = \sup |\sigma| < 1$) y equicontinua, pues si $f \in \mathcal{F}$,

$$\left| \frac{\partial f}{\partial x_i}(\mathbf{x}) \right| = |w_i \sigma'(\mathbf{w} \cdot \mathbf{x} + b)| < 1.$$

Luego, por el Teorema de Arzela-Ascoli, toda sucesión de salidas $\{f_{\mathbf{w}_k,b_k}\}_{k\in\mathbb{N}}$ contiene una subsucesión uniformemente convergente a una función continua $g$ definida en $I_n$. Observemos que no se hacen hipótesis sobre los sesgos.

**7.2.1. Aplicaciones a redes neuronales.** Las nociones anteriores pueden aplicarse al estudio de redes neuronales de una capa oculta con un continuo de neuronas en la capa oculta; ver también la sección 5.6.

Supongamos que la entrada pertenece a un intervalo compacto, $x \in [a, b]$. Es bien sabido que una red neuronal de una capa oculta, con $N$ neuronas en la capa oculta, tiene una salida dada por

$$g(x) = \sum_{j=1}^{N} \sigma(w_j x + b_j)\alpha_j,$$

donde $\sigma$ es una sigmoide logística. Además, $w_j$ son los pesos entre la entrada y la $j-$ésima neurona, $\alpha_j$ es el peso entre la $j-$ésima neurona y la salida, y $b_j$ denota el sesgo de la $j-$ésima neurona en la capa oculta.

Queremos generalizar al caso en que el número de unidades ocultas es infinito. En particular, queremos considerar el caso en que el número de unidades ocultas es *infinitamente numerable*. Suponemos que las neuronas ocultas están parametrizadas de forma continua por $t$, tomando valores en el compacto $[c, d]$. De este modo, el parámetro $t$ reemplazará al índice $j$.

Además, los pesos de suma $\alpha_j$ serán reemplazados por una medida, que bajo la hipótesis de ser absolutamente continua con respecto a la medida de Lebesgue $dt$ en $[c, d]$, tomará la forma $h(t)dt$.

La salida de la $j-$ésima neurona oculta, $\sigma(w_j x + b_j)$, es una función continua de $x$ que depende de $j$. Esto se transformará en el núcleo

$$K(x, t) = \sigma(w(t)x + b(t)).$$

La suma pasará a ser una integral con respecto a la medida mencionada.

Por lo tanto, el análogo continuo de la salida

$$g(x) = \sum_{j=1}^{N} \sigma(w_j x + b_j)\alpha_j,$$

es la integral

$$g(x) = \int_c^d K(x, t)h(t)\, dt.$$

Si $\mathcal{K}$ es la transformación integral con el núcleo $K$, entonces $g = \mathcal{K}(h)$, es decir, la salida es la transformación integral de la densidad de medida $h$.

Suponiendo la condición de regularización

$$\|h\|_2^2 \leq M,$$

es decir, que las densidades de medida están uniformemente acotadas en $L^2$, entonces el conjunto de salidas es equicontinuo y uniformemente acotado (ver Ejercicio 7.7.8). Por el Teorema de Arzela-Ascoli, existe una sucesión $\{h_n\}_{n\in\mathbb{N}}$ tal que la sucesión de salidas $g_n = \mathcal{K}(h_n)$ converge uniformemente en $[a, b]$ a una función continua $g$.

En el Capítulo 9, se mostrará que la red neuronal puede aprender exactamente cualquier función continua en $[a, b]$.

Cabe destacar que podemos considerar un caso aún más general donde el número de unidades ocultas puede ser finito, infinitamente numerable o infinitamente no numerable. Esto se logra eligiendo la salida

$$g(x) = \int K(x,t)\, d\mu(t)$$

donde la medida de ponderación $\mu$ se toma como en la sección 5.6.

## 7.3. El Teorema de Stone-Weierstrass

Notemos por $K$ un conjunto compacto en $\mathbb{R}^n$. Para fijar ideas (y será suficiente para futuras aplicaciones) podemos pensar que $K = I_n = [0,1] \times \cdots \times [0,1]$. Notamos al conjunto de funciones continuas sobre $K$ a valores en $\mathbb{R}$ como $C(K)$. Este conjunto es un espacio normado con la norma

$$\|f\| = \max_{x \in K} |f(x)|.$$

Esta norma, induce una distancia dada por $d(f,g) = \|f - g\|$ y $C(K)$ con esta métrica resulta un espacio métrico completo.

Observación 7.3.1. Observemos que si tenemos una sucesión $\{f_n\}_{n\in\mathbb{N}} \subset C(K)$ y una cierta función $g \in C(K)$, la convergencia de $f_n$ a $g$ es equivalente a la convergencia uniforme, es decir

$$\|f_n - g\| \to 0 \text{ cuando } n \to \infty \qquad \Longleftrightarrow \qquad f_n \rightrightarrows g \text{ cuando } n \to \infty.$$

Recordemos dos definiciones.

Definición 7.3.2. Sea $\mathcal{A} \subset C(K)$. Decimos que $\mathcal{A}$ es denso en $C(K)$ si para toda $g \in C(K)$ y todo $\varepsilon > 0$, existe $f \in C(K)$ tal que $d(f,g) = \|f - g\| < \varepsilon$.

Observación 7.3.3. Si $\mathcal{A}$ es un conjunto denso de $C(K)$ entonces se tiene que dada $g \in C(K)$ tenemos una sucesión $\{f_n\}_{n\in\mathbb{N}} \subset \mathcal{A}$ de manera tal que $f_n \rightrightarrows g$ cuando $n \to \infty$.

Definición 7.3.4. Un conjunto $\mathcal{A} \subset C(K)$ se denomina un álgebra si verifica

1. $\forall f, g \in \mathcal{A} \Rightarrow f + g \in \mathcal{A}$;
2. $\forall f \in \mathcal{A},\ \forall c \in \mathbb{R} \Rightarrow cf \in \mathcal{A}$;
3. $\forall f, g \in \mathcal{A} \Rightarrow fg \in \mathcal{A}$.

Las condiciones 1 y 2 dicen que $\mathcal{A}$ es un espacio vectorial.

Ejemplo 7.3.5. El conjunto de los polinomios trigonométricos es un álgebra en $C([0, 2\pi])$. En efecto, sea

$$\mathcal{A} = \left\{ \frac{a_0}{2} + \sum_{k=1}^{N} (a_k \cos(kx) + b_k \sin(kx)) \colon a_k, b_k \in \mathbb{R}, N = 0, 1, 2, \dots \right\}.$$

Obviamente, $\mathcal{A}$ verifica 1 y 2. Es también fácil ver, usando las fórmulas de adición del seno y del coseno que $\mathcal{A}$ cumple 3. Luego $\mathcal{A} \subset C([0, 2\pi])$ es un álgebra.

Definición 7.3.6. Sea $\mathcal{A} \subset C(K)$ un álgebra. Se dice que $\mathcal{A}$ separa puntos en $K$ si para todo $x, y \in K$, $x \neq y$, existe $f \in \mathcal{A}$ tal que $f(x) \neq f(y)$.

Es fácil ver que el álgebra del Ejemplo 7.3.5 separa puntos.

DEFINICIÓN 7.3.7. Sea $\mathcal{A} \subset C(K)$ un álgebra. Se dice que $\mathcal{A}$ contiene a las constantes si para todo $c \in \mathbb{R}$, la función $f(x) = c$ pertenece a $\mathcal{A}$.

Por la propiedad 2 si la función $f(x) = 1$ pertenece al álgebra $\mathcal{A}$ entonces $\mathcal{A}$ contiene a las constantes.

Es también fácil ver que el álgebra del Ejemplo 7.3.5 contiene a las constantes.

EJEMPLO 7.3.8. Sea $\mathcal{A}$ el conjunto de polinomios definidos en $[a, b]$

$$\mathcal{A} = \left\{ \sum_{k=0}^{N} c_k x^k \colon x \in [a, b], c_k \in \mathbb{R}, N = 0, 1, 2, \dots \right\}.$$

Entonces $\mathcal{A} \subset C([a, b])$ es un álgebra que separa puntos y contiene a las constantes.

El siguiente teorema se conoce como el Teorema de Stone-Weierstrass y su demostración suele verse en Cálculo Avanzado y en Análisis Avanzado. La demostración también está en el libro **[20]**.

TEOREMA 7.3.9 (Stone-Weierstrass). *Sea $K \subset \mathbb{R}^n$ un conjunto compacto y $\mathcal{A} \subset C(K)$ un álgebra que separa puntos y contiene a las constantes. Entonces $\mathcal{A}$ es denso en $C(K)$.*

Veamos una aplicación inmediata de este teorema.

PROPOSICIÓN 7.3.10. *Sea $f \colon [a, b] \times [c, d] \to \mathbb{R}$ una función continua. Entonces para todo $\varepsilon > 0$, existe $N \geq 1$ y funciones continuas $g_i \in C([a, b])$, $h_i \in C([c, d])$, $i = 1, \dots, N$, tales que*

$$\max_{x \in [a,b], y \in [c,d]} \left| f(x, y) - \sum_{i=1}^{N} g_i(x) h_i(y) \right| < \varepsilon.$$

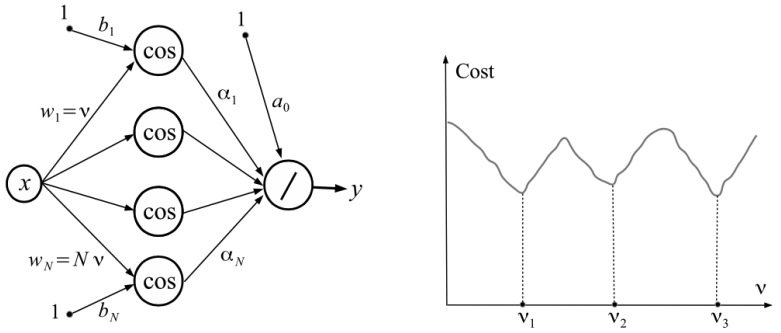
DEMOSTRACIÓN. Consideremos el conjunto

$$\mathcal{A} = \left\{ \sum_{i=1}^{N} g_i(x) h_i(y) \colon g_i \in C([a, b]), h_i \in C([c, d]), N = 1, 2, \dots \right\}.$$

Es sencillo ver que $\mathcal{A}$ es un álgebra de funciones que separa puntos y contiene a las constantes. Luego, aplicando el Teorema de Stone-Weierstrass obtenemos el resultado deseado. □

Obviamente, este resultado puede extenderse a cualquier número finito de producto de intervalos compactos.

**7.3.1. Aplicación a redes neuronales.** Como aplicación del Teorema de Stone-Weierstrass a redes neuronales, veamos como una red neuronal que tenga al coseno como función de activación (o al seno) puede aprender, potencialmente, cualquier función periódica.




**a** **b**

FIGURA 7.1. **a.** Red neuronal con una capa escondida y función de activación $\cos(x)$. **b.** La función costo alcanza sus mínimos locales para ciertos valores del hiperparámetro $\nu$.

Consideremos una red neuronal con una capa escondida, entrada real $x$, una salida unidimensional $y$ y $N$ neuronas en la capa escondida. Asumimos que la función de activación es $\phi(x) = \cos(x)$. La salida de la red está dada por la suma

$$y = a_0 + \sum_{j=1}^{N} \alpha_j \cos(w_j x + b_j),$$

donde $w_j$ y $\alpha_j$ son los pesos del ingreso hacia la capa oculta y de la capa oculta hacia la salida respectivamente. Los sesgos en la capa oculta se notan por $b_j$ mientras que $a_0$ denota el sesgo de la neurona en la capa de salida. Ver la Figura 7.3.1 **a**.

Afirmamos que esta red puede representar cualquier función continua y periódica. En efecto, consideremos $f\colon \mathbb{R} \to \mathbb{R}$ una función continua y periódica con período $T$. Es decir $f(x+T) = f(x)$. Sea $\nu = \frac{2\pi}{T}$ la frecuencia asociada. Consideremos ahora pesos de la forma $w_j = j\nu$. Aplicando fórmulas trigonométricas elementales, tenemos

$$\begin{aligned} y &= a_0 + \sum_{j=1}^{N} \alpha_j \cos(j\nu x + b_j) \\ &= a_0 + \sum_{j=1}^{N} \alpha_j \cos(j\nu x)\cos(b_j) - \alpha_j \sin(j\nu x)\sin(b_j) \\ &= a_0 + \sum_{j=1}^{N} a_j \cos(j\nu x) - c_j \sin(j\nu x), \end{aligned}$$

con coeficientes $a_j = \alpha_j \cos(b_j)$ y $c_j = \alpha_j \sin(b_j)$. Por el Ejercicio 7.7.7, los polinomios trigonométricos son densos en el conjunto de funciones periódicas sobre $\mathbb{R}$ con período $T$. Luego, dado $\varepsilon > 0$ existen $N > 1$, $\alpha_j, b_j, \nu \in \mathbb{R}$ tales que

$$\left| a_0 + \sum_{j=1}^{N} \alpha_j \cos(j\nu x + b_j) - f(x) \right| < \varepsilon.$$

Como el período $T$ siempre se puede considerar positivo, alcanza con considerar $\nu > 0$. Si tomamos $\varepsilon = \frac{1}{n}$, obtenemos una sucesión de frecuencias $\nu_n$ que aproximan a la frecuencia propia $\nu = \frac{2\pi}{T}$ de la función objetivo.

Es posible modificar este método levemente para encontrar frecuencias escondidas en una señal continua. Consideremos la señal objetivo $z = f(x)$ donde $x$ denota el tiempo. El peso $\nu$ del cálculo anterior se toma como un hiperparámetro. Luego, para cada valor de $\nu$ fijo se ajustan los otros parámetros, por ejemplo, usando el método de descenso de gradiente. Después se varía el valor de $\nu$ y se obtienen diferentes valores de la función de costo. Elegimos aquellos valores de $\nu$ para los que la función de costo alcanza mínimos locales, ver Figura 7.3.1 **b**. Estos mínimos se corresponden a frecuencias propias contenidas en la señal. El mínimo global se supone que se obtiene para el valor de $\nu$ que corresponde a la frecuencia principal.

## 7.4. Teoremas Tauberianos de Wiener

Para poder enunciar los teoremas Tauberianos es necesario introducir a la *Transformada de Fourier*.

Recordemos que $f \in L^1(\mathbb{R})$ si se verifica que

$$\int_{-\infty}^{\infty} |f(x)|\, dx < \infty,$$

Tenemos entonces la siguiente definición

DEFINICIÓN 7.4.1 (Transformada de Fourier). Dada $f \in L^1(\mathbb{R})$ se define su transformada de Fourier, notada por $\mathcal{F}[f]$ o por $\hat{f}$ a

$$\mathcal{F}[f](\xi) = \hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi}\, dx.$$

Es fácil ver que si $f \in L^1(\mathbb{R})$ entonces $\hat{f}$ está bien definida y resulta una función acotada,

$$|\hat{f}(\xi)| \leq \int_{-\infty}^{\infty} |f(x)|\, dx, \quad \forall \xi \in \mathbb{R}.$$

Algunos ejemplos:

1. *Doble exponencial.* Sea $f(x) = e^{-\lambda|x|}$, con $\lambda > 0$. Entonces
$$\hat{f}(\xi) = \frac{2\lambda}{\lambda^2 + \xi^2}.$$
2. *Potencial de Laplace.* Sea $f(x) = \frac{1}{a^2+x^2}$, con $a > 0$. Entonces
$$\hat{f}(\xi) = \frac{\pi}{a} e^{-a|\xi|}.$$
3. *Gaussiana.* Sea $f(x) = e^{-ax^2}$, con $a > 0$. Entonces
$$\hat{f}(\xi) = \sqrt{\frac{\pi}{a}} e^{-\frac{\xi^2}{4a}}.$$

**7.4.1. Aprendiendo señales en $L^1(\mathbb{R})$.** Tenemos el siguiente teorema que fue probado por N. Wiener en [**22**]

TEOREMA 7.4.2. *Sea $f \in L^1(\mathbb{R})$ y consideremos las traslaciones $f_\tau(x) = f(x+\tau)$ con $\tau \in \mathbb{R}$. Entonces el subespacio generado por la familia $\{f_\tau : \tau \in \mathbb{R}\}$ es denso en $L^1(\mathbb{R})$ si y sólo si $\hat{f}(\xi) \neq 0$ para todo $\xi \in \mathbb{R}$.*

Observemos que los ejemplos 1, 2, 3 verifican las hipótesis del Teorema 7.4.2.

La manera en la que se usa este resultado es la siguiente. Consideremos una red neuronal con función de activación $f \in L^1(\mathbb{R})$ que verifica $\hat{f}(\xi) \neq 0$. Luego, por el Teorema 7.4.2, para toda $g \in L^1(\mathbb{R})$ y $\varepsilon > 0$, tenemos una función

$$G(x) = \sum_{j=1}^{N} \alpha_j f(x+\tau_j), \qquad \alpha_j, \tau_j \in \mathbb{R},\ N = 1, 2, \ldots,$$

tal que

$$\|g - G\|_1 = \int_{\mathbb{R}} |g(x) - G(x)|\, dx < \varepsilon.$$

La función $G(x)$ es la salida de una red neuronal con una capa oculta con función de activación $f$, pesos $\alpha_j$ y sesgos $\tau_j$.

**7.4.2. Aprendiendo señales en $L^2(\mathbb{R})$.** Recordemos que una función $f$ pertenece a $L^2(\mathbb{R})$ si

$$\int_{-\infty}^{\infty} |f(x)|^2\, dx < \infty,$$

En general, una función $f$ puede pertenecer a $L^2(\mathbb{R})$ pero no a $L^1(\mathbb{R})$ y viceversa. Si $f$ pertenece a $L^2(\mathbb{R})$ pero no a $L^1(\mathbb{R})$ la expresión para la transformada de Fourier de $f$ no puede ser dada por la fórmula de la Definición 7.4.1. Sin embargo, es posible extender la definición de la transformada de Fourier al espacio $L^2(\mathbb{R})$ de manera tal que si $f \in L^2(\mathbb{R}) \cap L^1(\mathbb{R})$ entonces ambas definiciones coincidan. Ver, por ejemplo, [**11**].

En [**22**] se demuestra el siguiente resultado, análogo al Teorema 7.4.2, para funciones de $L^2(\mathbb{R})$.

TEOREMA 7.4.3. *Sea $f \in L^2(\mathbb{R})$ y consideremos las funciones trasladadas $f_\tau(x) = f(x+\tau)$ para $\tau \in \mathbb{R}$. Entonces el subespacio generado por la familia $\{f_\tau : \tau \in \mathbb{R}\}$ es denso en $L^2(\mathbb{R})$ si y sólo si $\hat{f}(\xi) \neq 0$ para casi todo $\xi \in \mathbb{R}$.*

Observemos que los ejemplos 1, 2, 3 también verifican las hipótesis del Teorema 7.4.3.

El Teorema 7.4.3 se utiliza de manera similar al Teorema 7.4.2. Consideremos una red neuronal con función de activación $f \in L^2(\mathbb{R})$ que verifica $\hat{f}(\xi) \neq 0$ en casi todo punto. Luego, por el Teorema 7.4.3, para toda $g \in L^2(\mathbb{R})$ y $\varepsilon > 0$, tenemos una función

$$G(x) = \sum_{j=1}^{N} \alpha_j f(x+\tau_j), \qquad \alpha_j, \tau_j \in \mathbb{R},\ N = 1, 2, \ldots,$$

tal que

$$\|g - G\|_2^2 = \int_{\mathbb{R}} |g(x) - G(x)|^2\, dx < \varepsilon.$$

La función $G(x)$ es la salida de una red neuronal con una capa oculta con función de activación $f$, pesos $\alpha_j$ y sesgos $\tau_j$.

OBSERVACIÓN 7.4.4. Obviamente, la aplicación de los Teoremas 7.4.2 y 7.4.3 tiene limitaciones. Por ejemplo las funciones que vimos en los ejemplos 1, 2, 3 son de forma *campana*, mientras que las funciones de activación más usadas son de tipo sigmoide o de tipo *palo de hockey* (función logística, ReLU, etc.) y no pertenecen a $L^1(\mathbb{R})$ o a $L^2(\mathbb{R})$.

## 7.5. Principio de contracción

En esta sección veremos aplicaciones del principio de contracción (Teorema de punto fijo de Banach) a redes neuronales.

Repasemos el principio de contracción. Si $(X, d)$ es un espacio métrico, una función $A\colon X \to X$ se dice una contracción si existe $\lambda \in (0, 1)$ tal que

$$d(A(x), A(y)) \leq \lambda d(x, y), \qquad \forall x, y \in X.$$

Observemos que toda contracción resulta ser uniformemente continua.

EJEMPLO 7.5.1. Si consideramos $X = \mathbb{R}$, entonces toda función $A\colon \mathbb{R} \to \mathbb{R}$ diferenciable con $|A'(x)| \leq \lambda < 1$ es una contracción. En efecto, por el Teorema de valor medio,

$$|A(x) - A(y)| = |A'(\xi)(x - y)| \leq \lambda |x - y|.$$

Un ejemplo de función $A$ que verifica estas hipótesis es $A(x) = \frac{1}{2}\cos(x)$.

El teorema de punto fijo de Banach dice lo siguiente.

TEOREMA 7.5.2 (Banach). *Sean $(X, d)$ un espacio métrico completo y $A\colon X \to X$ una contracción. Entonces $A$ posee un único punto fijo. Es decir, existe un único punto $x_0 \in X$ tal que $A(x_0) = x_0$.*

Este teorema se demuestra habitualmente en los cursos de Cálculo Avanzado y de Análisis Avanzado.

La salida de una red neuronal con una capa oculta teniendo función de activación $\sigma$ es dada por

$$A(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j x + b_j), \tag{7.5.1}$$

donde asumimos que las entradas y salidas son números reales, es decir $A\colon \mathbb{R} \to \mathbb{R}$.

Tenemos entonces la siguiente proposición.

PROPOSICIÓN 7.5.3. *Sea $A(x)$ dada por la expresión (7.5.1) y asumamos que la función $\sigma\colon \mathbb{R} \to \mathbb{R}$ verifica que $|\sigma'(x)| \leq M$ para todo $x \in \mathbb{R}$. Entonces $A(x)$ es una contracción si se verifica*

$$\sum_{j=1}^{N} \alpha_j^2 + w_j^2 < \frac{2}{M}. \tag{7.5.2}$$

DEMOSTRACIÓN. Observemos primero que

$$|\sigma(x) - \sigma(y)| = |\sigma'(\xi)||x - y| \leq M|x - y|, \qquad \forall x, y \in \mathbb{R}.$$

Ahora,

$$\begin{aligned} |A(x) - A(y)| &= \left| \sum_{j=1}^{N} \alpha_j (\sigma(w_j x + b_j) - \sigma(w_j y + b_j)) \right| \\ &\leq \sum_{j=1}^{N} |\alpha_j||\sigma(w_j x + b_j) - \sigma(w_j y + b_j)| \\ &\leq M|x - y| \sum_{j=1}^{N} |\alpha_j||w_j| \\ &\leq \frac{M}{2} \left( \sum_{j=1}^{N} \alpha_j^2 + w_j^2 \right) |x - y|. \end{aligned}$$

Luego, si llamamos

$$\lambda = \frac{M}{2} \left( \sum_{j=1}^{N} \alpha_j^2 + w_j^2 \right),$$

tenemos, por (7.5.2), que $0 < \lambda < 1$ como queríamos ver. □

OBSERVACIÓN 7.5.4. En el caso en que tomemos como función de activación a la función logística $\sigma(x) = (1 + e^{-x})^{-1}$, tenemos que $0 < \sigma'(x) \leq \frac{1}{4}$, de donde obtenemos que $A(x)$ es una contracción si se verifica que

$$\sum_{j=1}^{N} \alpha_j^2 + w_j^2 < 8.$$

COROLARIO 7.5.5. *Sea $A(x)$ dada por (7.5.1) tal que la función de activación $\sigma$ verifica $|\sigma'(x)| \leq M$ para todo $x \in \mathbb{R}$. Asumamos que los coeficientes $\alpha_j$ y $w_j$ verifican (7.5.2). Entonces existe un único $x_0 \in \mathbb{R}$ tal que $A(x_0) = x_0$.*

**7.5.1. Aplicaciones a redes neuronales recurrentes.** En esta sección, una *unidad neuronal* significará una red neuronal con una capa escondida para las que se verifica la condición (7.5.2) de la Proposición 7.5.3, teniendo un ingreso $x \in \mathbb{R}$ y una salida $A(x) \in \mathbb{R}$ dada por la fórmula (7.5.1). Una unidad neuronal se muestra en la figura 7.5.1

Ahora, concatenamos varias unidades neuronales para construir una *red neuronal recurrente* como se muestra en la Figura 7.5.1. Esto significa que la salida de la $n-$ésima unidad neuronal es la entrada de la $(n+1)$ unidad neuronal, para $n \geq 1$. La salida de la $n-$ésima unidad es $A^n(x) = (A \circ \cdots \circ A)(x)$. En un primer momento, supongamos que los pesos y sesgos de todas las unidades neuronales son los mismos y el aprendizaje de la red no está supervisado, es decir que no hay una función objetivo específica.

La pregunta es: *¿Qué aprende este tipo de red?*

Para ser más precisos, asumamos que para cada valor de entrada $x \in \mathbb{R}$, la $n-$ésima salida, $y_n = A^n(x)$ es una sucesión convergente con límite $y$, que depende obviamente de $x$. Si definimos $y = \Lambda(x)$, entonces la función aprendida por la red es $\Lambda(x)$.

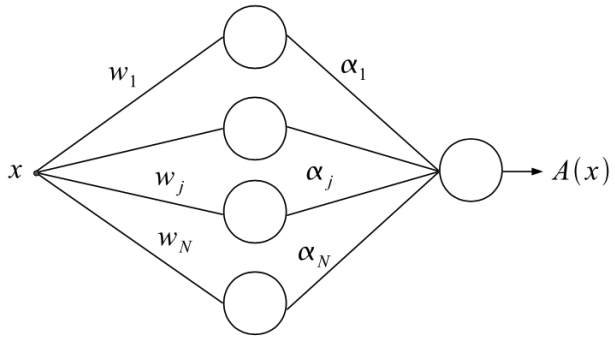



FIGURA 7.2. Una unidad neuronal de tres capas cuya salida es una contracción.

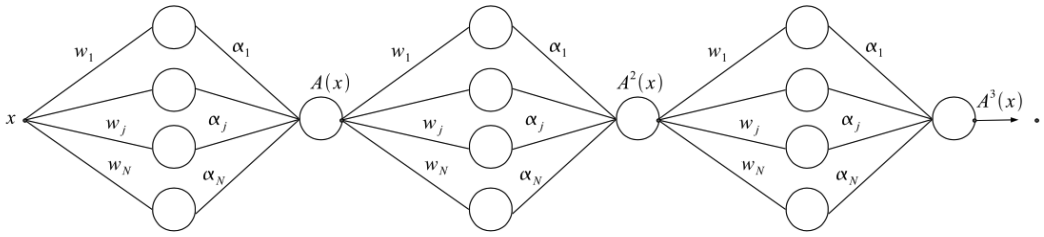



FIGURA 7.3. Una red neuronal recurrente construida a partir de unidades neuronales de tres capas.

El siguiente resultado trata sobre existencia, unicidad y la forma de esta función.

PROPOSICIÓN 7.5.6. *La red neuronal recurrente construida en la Figura 7.5.1 aprende una constante $x_0$, que es el único punto fijo del mapa $y = A(x)$.*

DEMOSTRACIÓN. La demostración de este resultado es muy similar a la demostración del Teorema de Banach, Teorema 7.5.2.

En efecto, para cada valor de entrada $x$, consideramos la sucesión de números reales $y_n = A^n(x)$. Debemos probar que $\{y_n\}_{n\in\mathbb{N}}$ es convergente, pero para eso veamos que es una sucesión de Cauchy.

Usando la propiedad de contracción de la función $A(x)$, estimamos la diferencia entre dos elementos consecutivos de la sucesión

$$\begin{aligned} |y_{n+1} - y_n| &= |A^{n+1}(x) - A^n(x)| = |A(A^n(x)) - A(A^{n-1}(x))| \\ &\leq \lambda |A^n(x) - A^{n-1}(x)| = \lambda |y_n - y_{n-1}|. \end{aligned}$$

Iterando esta desigualdad, llegamos a

$$|y_{n+1} - y_n| \leq \lambda^n |y_1 - x|.$$

Sea ahora $m = n + k$ y estimamos la diferencia entre dos términos arbitrarios de la sucesión usando la desigualdad triangular, y recordando que $\lambda \in (0, 1)$,

$$\begin{aligned} |y_{n+k} - y_n| &\leq \sum_{j=0}^{k-1} |y_{n+j+1} - y_{n+j}| \leq \sum_{j=0}^{k-1} \lambda^{n+j} |y_1 - x| \\ &\leq \lambda^n |y_1 - x| \sum_{j=0}^{\infty} \lambda^j = \frac{\lambda^n}{1-\lambda} |y_1 - x|, \end{aligned} \tag{7.5.3}$$

lo que prueba que $\{y_n\}_{n\in\mathbb{N}}$ es de Cauchy, por lo que existe $y \in \mathbb{R}$ tal que $y_n \to y$ cuando $n \to \infty$. Notemos $y = \Lambda(x)$.

Veamos ahora que $y$ es el único punto fijo de $A(x)$ con lo que se concluye la demostración. Pero tenemos que

$$|A(y) - y| \leq |A(y) - y_n| + |y_n - y| = |A(y) - A(y_{n-1})| + |y_n - y|.$$

Al ser $A$ una contracción, resulta continua. Luego tomando límite $n \to \infty$ en la última desigualdad, obtenemos que $|A(y) - y| = 0$, con lo que $A(y) = y$, es decir, $y$ es un punto fijo de $A$. La unicidad es una consecuencia directa de la contracción, ya que si $y$ e $y'$ son dos puntos fijos, entonces

$$|y - y'| = |A(y) - A(y')| \leq \lambda |y - y'|,$$

y como $\lambda \in (0, 1)$ sigue que $|y - y'| = 0$, o lo que es lo mismo que $y = y'$. □

Observación 7.5.7. Es sencillo estimar el error cometido al aproximar el punto fijo $x_0$ de $A$ por la sucesión aproximante $y_n$. Tomando límite $k \to \infty$ en la cota (7.5.3), obtenemos

$$|y_n - x_0| \leq \frac{\lambda^n}{1-\lambda} |A(x) - x|.$$

El valor de $n \in \mathbb{N}$, que es la profundidad de la red, puede ser elegido de manera tal que el error sea tan pequeño como queramos.

Si cambiamos los pesos de la red, esto lleva al aprendizaje de diferentes constantes. Veamos que estas constantes dependen continuamente de los pesos. Empecemos con el concepto de proximidad de redes.

Definición 7.5.8. Dos redes neuronales se dicen $\varepsilon-$cercanas en $I \subset (-R, R) \subset \mathbb{R}$ si para algún $\varepsilon > 0$ sus mapas de salida verifican

$$|A(x) - \bar{A}(x)| < \varepsilon, \qquad \forall x \in I \subset \mathbb{R}.$$

En otras palabras, dos redes son próximas si sus mapas de salida están próximos con respecto a la distancia de supremo.

Lema 7.5.9. *Consideremos dos unidades neuronales con mapas de salida dados por*

$$A(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j x + b_j) \quad y \quad \bar{A}(x) = \sum_{j=1}^{N} \bar{\alpha}_j \sigma(\bar{w}_j x + \bar{b}_j),$$

*donde la función de activación $\sigma$ verifica $|\sigma(x)| \leq C$ y $|\sigma'(x)| \leq M$ para todo $x \in \mathbb{R}$.*

*Entonces, si*

$$\sum_{j=1}^{N} \left( M|\alpha_j|(R|w_j - \bar{w}_j| + |b_j - \bar{b}_j|) + C|\alpha_j - \bar{\alpha}_j| \right) < \varepsilon,$$

*entonces las unidades neuronales están $\varepsilon-$cercanas en $I$.*

DEMOSTRACIÓN. Observemos primero que

$$\begin{aligned} |\alpha_j\sigma(w_jx + b_j) - \bar{\alpha}_j\sigma(\bar{w}_jx + \bar{b}_j)| \leq & |\alpha_j||\sigma(w_jx + b_j) - \sigma(\bar{w}_jx + \bar{b}_j)| \\ & + |\alpha_j - \bar{\alpha}_j||\sigma(\bar{w}_jx + \bar{b}_j)| \\ \leq & |\alpha_j||\sigma'(\xi)|(|w_j - \bar{w}_j||x| + |b_j - \bar{b}_j|) \\ & + C|\alpha_j - \bar{\alpha}_j| \\ \leq & |\alpha_j|M(|w_j - \bar{w}_j||x| + |b_j - \bar{b}_j|) + C|\alpha_j - \bar{\alpha}_j| \end{aligned}$$

Luego,

$$\begin{aligned} |A(x) - \bar{A}(x)| &\leq \sum_{j=1}^{N} |\alpha_j\sigma(w_jx + b_j) - \bar{\alpha}_j\sigma(\bar{w}_jx + \bar{b}_j)| \\ &\leq \sum_{j=1}^{N} |\alpha_j|M(|w_j - \bar{w}_j||x| + |b_j - \bar{b}_j|) + C|\alpha_j - \bar{\alpha}_j| < \varepsilon, \end{aligned}$$

como queríamos probar. □

OBSERVACIÓN 7.5.10. De la demostración es evidente de que en el caso $I = \mathbb{R}$, entonces se debe pedir que $w_j = \bar{w}_j$, $b_j = \bar{b}_j$ y

$$\sum_{j=1}^{N} |\alpha_j - \bar{\alpha}_j| < \frac{\varepsilon}{C}.$$

LEMA 7.5.11. *Consideremos dos unidades neuronales $\varepsilon-$cercanas con mapas de entrada-salida dados por $A(x)$ y $\bar{A}(x)$ respectivamente. Notemos por $x_0$ y $\bar{x}_0$ los respectivos puntos fijos. Entonces $|x_0 - \bar{x}_0| < \frac{1}{1-\lambda}\varepsilon$, donde $\lambda \in (0, 1)$ es el factor de contracción de $A$.*

DEMOSTRACIÓN. Sea $\bar{x}_0$ el punto fijo del mapa $\bar{A}$ y sea $\lambda \in (0, 1)$ el factor de contracción de $A$. Si llamamos $y_n = A(\bar{x}_0)$, entonces por la Proposición 7.5.6, tenemos que $y_n \to x_0$ donde $x_0$ es el punto fijo de $A$. Por otro lado, si en la desigualdad (7.5.3) tomamos $n = 0$, obtenemos que

$$|y_k - \bar{x}_0| \leq \frac{1}{1-\lambda}|A(\bar{x}_0) - \bar{x}_0|.$$

Tomando límite cuando $k \to \infty$, obtenemos

$$|x_0 - \bar{x}_0| \leq \frac{1}{1-\lambda}|A(\bar{x}_0) - \bar{x}_0|.$$

Por otro lado,

$$|A(\bar{x}_0) - \bar{x}_0| = |A(\bar{x}_0) - \bar{A}(\bar{x}_0)| < \varepsilon.$$

Esto concluye la demostración. □

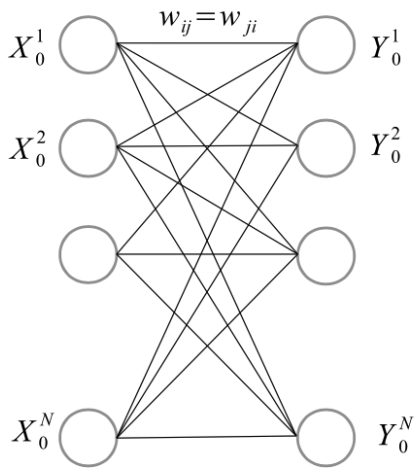



FIGURA 7.4. Una red neuronal resonante.

Tenemos entonces el siguiente corolario inmediato.

COROLARIO 7.5.12. *Consideremos dos unidades neuronales como en el Lema 7.5.9. Entonces, dado $\varepsilon > 0$, existe $\delta > 0$ tal que*

$$\sum_{j=1}^{N} |\alpha_j - \bar{\alpha}_j| < \delta \Longrightarrow |x_0 - \bar{x}_0| < \varepsilon,$$

*donde $x_0$ y $\bar{x}_0$ son los puntos fijos de los mapas de entrada-salida de las unidades neuronales.*

**7.5.2. Redes resonantes.** Consideramos una red neuronal con dos capas, ver Figura 7.5.2. La información se transmite de un lado a otro entre las capas hasta que eventualmente converge a un estado estacionario.

Sea $N$ el número de neuronas en cada capa. El estado inicial de la capa de la izquierda está dado por el vector $X_0$. Esto induce el estado en la segunda capa

$$Y_0 = \varphi(WX_0 + b),$$

donde los pesos se asumen simétricos, es decir, $w_{ij} = w_{ji}$.

Esta información se retroalimenta a la primera capa, lo que proporciona la salida

$$X_1 = \varphi(WY_0 + b).$$

Aquí, $\varphi$ es la función de activación para ambas capas y $b$ es el vector de sesgo común.

De esta manera, obtenemos las sucesiones $\{X_n\}_{n\in\mathbb{N}}$ y $\{Y_n\}_{n\in\mathbb{N}}$ definidas recursivamente por el sistema

$$Y_n = \varphi(WX_n + b), \quad X_n = \varphi(WY_{n-1} + b), \quad n \geq 1.$$

Esto puede reformularse en términos de dos recurrencias como

$$Y_n = F(Y_{n-1}), \quad X_n = F(X_{n-1}),$$

donde $F\colon \mathbb{R}^n \to \mathbb{R}^n$ está dada por

$$F(U) = \varphi\left(W\varphi(WU + b) + b\right).$$

El estado estacionario de la red corresponde al par de puntos fijos $(X^*, Y^*)$, donde

$$F(X^*) = X^*, \quad F(Y^*) = Y^*.$$

Abordaremos este problema utilizando el Principio de Contracción, Teorema 7.5.2. Primero, reescribimos el problema de manera equivalente de la siguiente forma. Definimos

$$Z_0 = X_0, \quad Z_1 = Y_0, \quad Z_2 = X_1,$$

y en general,

$$Z_{n+1} = G(Z_n),$$

donde

$$G(Z) = \varphi(WZ + b).$$

Basta con demostrar que $\{Z_n\}_{n\in\mathbb{N}}$ es una sucesión de vectores que converge a algún vector $Z^*$. Como $G\colon \mathbb{R}^n \to \mathbb{R}^n$, se obtiene la estimación

$$\|G(Z') - G(Z)\|_2 \leq \|\nabla G\|\|Z' - Z\|_2 \leq \sqrt{N}\|\varphi'\|_\infty\|W\|\|Z' - Z\|_2, \quad \forall Z, Z' \in \mathbb{R}^n.$$

Si los pesos satisfacen la desigualdad

$$\|W\| < \frac{1}{\sqrt{N}\|\varphi'\|_\infty},$$

entonces $G$ se convierte en una aplicación contractiva. El Teorema 7.5.2 garantiza que $G$ tiene un único punto fijo $Z^*$, al cual converge la sucesión $\{Z_n\}_{n\in\mathbb{N}}$.

La conclusión es que si los pesos $w_{ij}$ son suficientemente pequeños, el estado de la red tiende a un *estado estable* o *estado resonante*.

## 7.6. Resumen

Este capítulo contiene algunos resultados clásicos de análisis real útiles para la representación de redes neuronales.

Para el aprendizaje de funciones continuas $f \in C([a, b])$, se requiere la convergencia uniforme. La red neuronal genera una sucesión de funciones continuas, la cual converge puntualmente a la función objetivo $f$. Para demostrar que esta convergencia es uniforme, podemos utilizar, por ejemplo, el Teorema de Dini, Teorema 7.1.1.

Otra técnica para construir una sucesión de funciones uniformemente convergente en un conjunto compacto es mediante el Teorema de Arzela-Ascoli. En este caso, se puede extraer una subsucesión convergente de un conjunto de funciones que es uniformemente acotado y equicontinuo. Como consecuencia, las redes neuronales de una capa oculta con función de activación sigmoide pueden aprender una función continua en un conjunto compacto.

Otro método de aproximación de funciones continuas en un conjunto compacto es mediante el Teorema de Stone-Weierstrass. Este resultado también puede utilizarse en redes neuronales para aprender funciones periódicas.

Las funciones objetivo en los espacios $L^1$ y $L^2$ pueden ser aprendidas utilizando los Tteoremas tauberianos de Wiener. Estos teoremas se aplican principalmente a funciones de activación con forma de campana.

El principio de contracción, el cual garantiza la existencia de un punto fijo, puede aplicarse a las funciones de mapeo entrada-salida de una red neuronal de una capa oculta, así como a redes resonantes. Mediante la iteración del mapeo entrada-salida, se obtiene que la red neuronal aprende cierta función.

## 7.7. Ejercicios

EJERCICIO 7.7.1. Sea $\mathcal{F} = \{\tanh(ax+b)\colon a, b \in \mathbb{R}\}$. Mostrar que la familia $\mathcal{F}$ es uniformemente acotada.

EJERCICIO 7.7.2. Sea $\mathcal{F} = \{ax+b\colon |a|+|b|<1,\ x\in[0,1]\}$. Mostrar que la familia es equicontinua y uniformemente acotada.

EJERCICIO 7.7.3. Sea $M>0$ y consideremos la familia

$$\mathcal{F} = \left\{ f \in C^1([a,b])\colon \int_a^b |f'(x)|^2\,dx \le M \right\}.$$

Mostrar que $\mathcal{F}$ es equicontinua.

EJERCICIO 7.7.4. Notemos por $\mathcal{D}\subset(a,b)$ un conjunto denso. Sea $\mathcal{F}$ la familia de funciones tales que

(a) Es equicontinua sobre el intervalo $(a,b)$.
(b) Para todo $x\in\mathcal{D}$, la familia es uniformemente acotada en $x$, es decir, existe $M>0$ tal que $|f(x)|\le M$ para toda $f\in\mathcal{F}$.

Mostrar que $\mathcal{F}$ es uniformemente acotada en todos los puntos.

EJERCICIO 7.7.5. Sean $\mathcal{F}$ una familia equicontinua, $k\ge 1$ un entero fijo y $\{w_1,\dots,w_k\}$ un conjunto de pesos tales que $|w_j|<1$ para $j=1,\dots,k$. Mostrar que el conjunto sigue siendo equicontinuo si se extiende para incluir a las combinaciones lineales de la forma $\sum_{j=1}^k w_j f_j$ con $f_j\in\mathcal{F}$, $j=1,\dots,k$.

EJERCICIO 7.7.6. Una red neuronal usando neuronas sigmoides es diseñada para aprender una función continua $f\in C([a,b])$ por el método de descenso de gradiente. La salida de la red obtenida luego de la $n-$ésima actualización de los parámetros es denotada por $G_n(x)=G(x;w(n),b(n))$. Asumamos que en cada paso la aproximación mejora, es decir $|f(x)-G_{n+1}(x)|<|f(x)-G_n(x)|$ para todo $x\in[a,b]$ y todo $n\ge 1$. Probar que $G_n$ converge uniformemente a $f$ en $[a,b]$ (es decir, la red aprende cualquier función continua en $[a,b]$).

EJERCICIO 7.7.7. Sea $f\colon\mathbb{R}\to\mathbb{R}$ una función continua y periódica con período $T>0$, i.e. $f(x+T)=f(x)$ para todo $x\in\mathbb{R}$. Mostrar que para todo $\varepsilon>0$, existe una función

$$F(x) = a_0 + \sum_{n=1}^{N}\left(a_n\cos\frac{2\pi n x}{T} + b_n\sin\frac{2\pi n x}{T}\right),$$

tal que $|F(x)-f(x)|<\varepsilon$ para todo $x\in\mathbb{R}$.

EJERCICIO 7.7.8 (El conjunto de transformaciones integrales de funciones $L^2$). Sea $M>0$ y sea $K\colon[a,b]\times[c,d]\to\mathbb{R}$ una función continua. Se define la clase

$$\mathcal{F}_M = \left\{ g(x) = \int_c^d K(x,y)h(y)\,dy\colon h\in C([c,d]),\ \int_c^d |h(y)|^2\,dy \le M \right\}.$$

Mostrar que $\mathcal{F}_M$ es equicontinuo y uniformemente acotado.

EJERCICIO 7.7.9. Se define la clase

$$\mathcal{A} = \left\{ \sum_{i=1}^{n} \alpha_i e^{m_i x} \colon \alpha_i \in \mathbb{R},\ m_i \in \mathbb{N}_0,\ n = 1,2\ldots. \right\}.$$

Mostrar que $\mathcal{A} \subset C([a,b])$ es un álgebra de funciones. Probar que dada $f \in C([a,b])$ y $\varepsilon > 0$, existen $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ y $m_1, \ldots, m_n \in \mathbb{N}_0$ tales que

$$\left| f(x) - \sum_{i=1}^{n} \alpha_i e^{m_i x} \right| < \varepsilon, \quad \forall x \in [a,b].$$

Formular una interpretación en términos de aprendizaje automático de este resultado.

Capítulo 8

# Aprendizaje con entradas unidimensionales

Este capitulo analiza el caso de una red neuronal cuya entrada es acotada y unidimensional, por ejemplo, $x \in [0,1]$. Aparte de su simplicidad, este caso es importante desde una serie de puntos de vista: puede ser tratado de manera elemental sin hacer uso del arsenal de herramientas del análisis funcional (como lo haremos en el Capítulo 9) y, por la naturaleza constructiva de este caso, provee un algoritmo explícito para hallar los pesos de la red.

Los casos de redes neuronales con una capa oculta y neuronas de tipo perceptron y sigmoides serán estudiados. Adicionalmente estudiaremos con detalle el aprendizaje usando ReLU y Softplus.

## 8.1. Resultados preliminares

En esta sección usaremos las nociones de derivadas generalizadas, medida, integración y medida de Dirac. A no asustarse que introduciremos esos conceptos de manera simple y accesible.

1. Veamos primero que toda función simple (función escalón), puede escribirse como una combinación lineal de traslaciones de la función de Heaviside.

   Suponiendo que la función simple tiene soporte en el intervalo $[0,1]$, la misma puede escribirse como

   $$c(x) = \sum_{i=1}^{N} \alpha_i \mathbf{1}_{[x_{i-1},x_i)}(x),$$

   donde $0 = x_0 < x_1 < \cdots < x_N = 1$ es una partición del intervalo $[0,1]$ y $\mathbf{1}_{[x_{i-1},x_i)}(x)$ es la función indicadora del intervalo $[x_{i-1}, x_i)$.

   Observemos que la indicadora $\mathbf{1}_{x_{i-1},x_i)}(x)$ se puede escribir como diferencia de dos funciones de Heaviside trasladadas,

   $$\mathbf{1}_{[x_{i-1},x_i)}(x) = H(x - x_{i-1}) - H(x - x_i).$$

   Luego obtenemos fácilmente que

   $$c(x) = \sum_{i=1}^{N} \alpha_i \mathbf{1}_{[x_{i-1},x_i)}(x) = \sum_{i=0}^{N} c_i H(x - x_i). \tag{8.1.1}$$

2. Una consecuencia del ítem anterior es que la derivada de una función simple es una combinación lineal de deltas de Dirac,

   $$c'(x) = \frac{d}{dx} \sum_{i=0}^{N} c_i H(x - x_i) = \sum_{i=0}^{N} c_i \delta(x - x_i) = \sum_{i=0}^{N} c_i \delta_{x_i}(x).$$

Esta relación se interpreta como que la sensibilidad de la salida con respecto a la entrada, $c'(x)$ es la superposición de $N$ golpes $(\delta_{x_i})$ de magnitud $c_i$.

3. La delta de Dirac se puede aproximar (débilmente) por medidas de probabilidad con densidades de forma de campana. Más precisamente, tenemos

PROPOSICIÓN 8.1.1. *Consideremos una función de activación $\varphi\colon \mathbb{R} \to \mathbb{R}$ que satisface:*
(i) *$\varphi$ es creciente;*
(ii) *$\varphi(\infty) - \varphi(-\infty) = 1$;*
(iii) *$\varphi$ es de clase $C^1$ con $\varphi'(x)$ acotada.*
*Para $\varepsilon > 0$ definimos $\varphi_\varepsilon(x) = \varphi(\frac{x}{\varepsilon})$ y consideremos la medida de probabilidad $\mu_\varepsilon$ con densidad dada por $\varphi'_\varepsilon(x)$. Entonces $\mu_\varepsilon \to \delta$ cuando $\varepsilon \downarrow 0$ (en sentido débil).*

DEMOSTRACIÓN. Observemos que si $g \in C_c^\infty(\mathbb{R})$ (es decir, $g$ es infinitamente diferenciable y tiene soporte compacto), tenemos

$$\int_{-\infty}^{\infty} \varphi'_\varepsilon(x)g(x)\,dx = \int_{-\infty}^{\infty} \frac{1}{\varepsilon}\varphi'(\tfrac{x}{\varepsilon})g(x)\,,dx = \int_{-\infty}^{\infty} \varphi'(y)g(\varepsilon y)\,dy.$$

Tomando límite cuando $\varepsilon \downarrow 0$, obtenemos

$$\lim_{\varepsilon\downarrow 0} \int_{-\infty}^{\infty} \varphi'(y)g(\varepsilon y)\,dy = g(0)\int_{-\infty}^{\infty} \varphi'(y)\,dy = g(0)(\varphi(\infty) - \varphi(-\infty)) = g(0).$$

Luego

$$\lim_{\varepsilon\downarrow 0} \int_{-\infty}^{\infty} g(x)\,d\mu_\varepsilon(x) = \lim_{\varepsilon\downarrow 0} \int_{-\infty}^{\infty} \varphi'_\varepsilon(x)g(x)\,dx = g(0) = \int_{-\infty}^{\infty} g(x)\,d\delta(x).$$

Como $g \in C_c^\infty(\mathbb{R})$ es arbitraria, obtenemos que $\mu_\varepsilon \to \delta$ cuando $\varepsilon \downarrow 0$ en sentido débil. □

Un ejemplo de función $\varphi$ que satisface las hipótesis de la Proposición 8.1.1 es la función logística

$$\varphi(x) = \frac{1}{1 + e^{-x}}.$$

Observemos que en general se obtiene que tanto $\varphi'$ como $\varphi'_\varepsilon$ tiene la forma de *función de salto* y ese salto se hace más elevado y fino a medida que $\varepsilon \downarrow 0$.

Observemos también que

$$\int_{\mathbb{R}} d\mu_\varepsilon = \int_{\mathbb{R}} \varphi'_\varepsilon(x)\,dx = \int_{\mathbb{R}} \frac{1}{\varepsilon}\varphi'(\tfrac{x}{\varepsilon})\,dx = \int_{\mathbb{R}} \varphi'(y)\,dy = 1,$$

lo que implica que $\mu_\varepsilon$ es una medida de probabilidad para todo $\varepsilon > 0$.

4. La convolución de una función $f(x)$ con la delta de Dirac centrada en un punto $x_0$, $\delta_{x_0}(x) = \delta(x - x_0)$, coincide con la composición de $f$ con la traslación de tamaño $x_0$:

$$(\delta_{x_0} * f)(x) = \int_{\mathbb{R}} \delta_{x_0}(y)f(x - y)\,dy = f(x - x_0).$$

Esto puede enunciarse de manera equivalente diciendo que una señal $f(x)$ filtrada a través de la medida de Dirac $\delta_{x_0}$ da la misma señal a la que se le aplicó un cambio de fase.

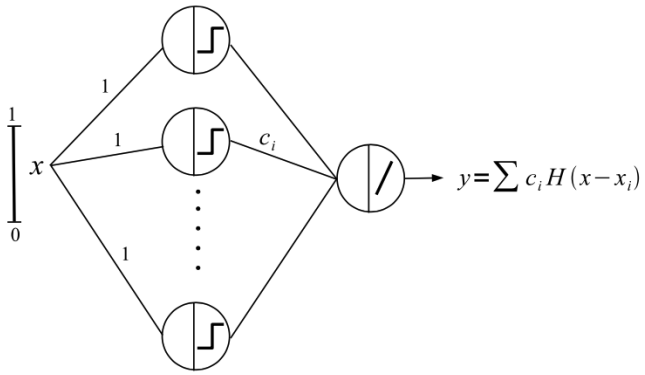



FIGURA 8.1. Una red neuronal con una capa oculta compuesta por perceptrones con pesos iguales a 1 y pesos de salida igual a $c_i$, los sesgos están dados por $x_i$.

5. Un resultado importante que se estudia tanto en Cálculo Avanzado como en Análisis Avanzado es que toda función continua definida en un intervalo compacto $g\colon [a,b] \to \mathbb{R}$ resulta ser *uniformemente continua*.

## 8.2. Red de perceptrones con una capa escondida

Consideremos una red neuronal con una capa escondida, funciones de activación en la capa escondida dada por la función de Heaviside y una función de activación lineal en la capa de salida. Vamos a mostrar que esta red puede aprender, potencialmente, cualquier función continua $g \in C([0,1])$. Equivalentemente, cualquier función continua definida en un intervalo compacto puede ser aproximada uniformemente por funciones simples. La red está representada en la Figura 8.2.

Notemos primero que el lado derecho de la ecuación (8.1.1) es una salida particular de esta red. El siguiente resultado muestra que la red aproxima funciones continuas por funciones escalón.

PROPOSICIÓN 8.2.1. *Sea $g \in C([0,1])$. Entonces, para todo $\varepsilon > 0$, existe una función simple $c(x) = \sum_{i=0}^{N-1} c_i H(x - x_i)$ tal que*

$$|g(x) - c(x)| < \varepsilon, \quad \forall x \in [0,1].$$

DEMOSTRACIÓN. Sea $\varepsilon > 0$ y sea $\delta > 0$ el asociado a la continuidad uniforme de $g(x)$ en el intervalo $[0,1]$. Tomemos $N \in \mathbb{N}$ tal que $\frac{1}{N} < \delta$ y consideremos la partición uniforme del intervalo $[0,1]$, $x_i = \frac{i}{N}$, $i = 0, 1, \ldots, N$. Observemos que, por la continuidad uniforme, tenemos que

$$|g(x) - g(x_i)| < \varepsilon, \quad \forall x \in [x_i, x_{i+1}).$$

Definimos inductivamente los números $c_i$, $i = 0, 1, \ldots, N$ de la forma

$$\begin{aligned} g(x_0) &= c_0 \\ g(x_1) &= c_0 + c_1 \\ &\vdots \\ g(x_{N-1}) &= c_0 + c_1 + \cdots + c_{N-1}. \end{aligned}$$

Observemos también que si $x \in [x_i, x_{i+1})$, entonces

$$H(x - x_j) = \begin{cases} 1, & \text{si } j \leq i \\ 0, & \text{si } j > i. \end{cases}$$

Luego, si definimos la función simple $c(x)$ como

$$c(x) = \sum_{i=0}^{N-1} c_i H(x - x_i),$$

tenemos que $c(x) = g(x_i)$ si $x \in [x_i, x_{i+1})$.

En conclusión, obtenemos que

$$|g(x) - c(x)| = |g(x) - g(x_i)| < \varepsilon, \quad \text{si } x \in [x_i, x_{i+1}),\ i = 0, 1, \ldots, N-1,$$

lo que concluye la demostración. □

Observación 8.2.2. Observemos que en la demostración del resultado anterior, tenemos que los pesos $c_i$ vienen dados por

$$c_i = g(x_{i+1}) - g(x_i),$$

y dado que $x_{i+1} - x_i = \frac{1}{N} < \delta$ obtenemos que $|c_i| < \varepsilon$.

Notemos finalmente que para poder tener pesos de la capa de salida pequeños, el precio que se paga es que $N$ (el número de neuronas) sea grande.

## 8.3. Red sigmoide con una capa escondida

Consideremos una red neuronal con una capa escondida con función de activación logística o cualquier función de tipo sigmoide, que notamos por $\varphi$, para la capa oculta y función de activación lineal para la capa de salida. Asumimos que $\varphi$ verifica las hipótesis de la Proposición 8.1.1. Veremos que esta red es capaz de aprender cualquier función continua $g \in C([0, 1])$.

El resultado principal de esta sección es el siguiente.

Teorema 8.3.1. *Consideremos una función de activación que verifica las hipótesis de la Proposición 8.1.1 y sea $g \in C([0, 1])$. Entonces, dado $\varepsilon > 0$, existen $c_i, w, \theta_i \in \mathbb{R}$ y $N \geq 0$ tales que*

$$\left| g(x) - \sum_{i=1}^{N} c_i \varphi(wx + \theta_i) \right| < \varepsilon, \quad \forall x \in [0, 1].$$

DEMOSTRACIÓN. La idea general de la demostración es la siguiente. Como primer paso, usamos la Proposición 8.2.1 para obtener una función simple $c(x)$ que aproxime $g(x)$. Luego, usamos un proceso de *regularización por convolución* para aproximar la función de Heaviside por versiones suaves de la misma y veremos que esas regularizaciones se pueden escribir como combinaciones lineales de nuestra función sigmoide $\varphi$.

Vayamos ahora desarrollando cada paso de la idea.

Paso 1: *Aproximación de g por una función simple.* Por la Proposición 8.2.1, existe $c(x)$ simple tal que

$$|g(x) - c(x)| < \frac{\varepsilon}{2}, \quad \forall x \in [0,1].$$

Por la Observación 8.2.2, podemos asumir que los coeficientes de $c(x)$ satisfacen $|c_i| < \frac{\varepsilon}{4}$, para todo $i$.

Paso 2: *Regularización de la función simple.* Definimos la familia $\varphi_\alpha(x) = \varphi(\frac{x}{\alpha})$, $\alpha > 0$, donde $\varphi$ es la función sigmoide de activación de la capa oculta de la red. Consideremos la función de salto $\psi_\alpha(x) = \varphi'_\alpha(x) = \frac{1}{\alpha}\varphi'(\frac{x}{\alpha})$. Por la Proposición 8.1.1, $\psi_\alpha(x)$ es la densidad de una medida de probabilidad, $\mu_\alpha$, que converge débil a la delta de Dirac $\delta(x)$, cuando $\alpha \downarrow 0$. Definimos entonces la función regularizada de $c(x)$, $c_\alpha(x)$ por

$$c_\alpha(x) = (c * \psi_\alpha)(x) = \int_{\mathbb{R}} c(y)\psi_\alpha(x-y)\,dy.$$

A este proceso se lo denomina *regularización por convolución.*

Veamos que la función regularizada $c_\alpha(x)$ es de la forma

$$c_\alpha(x) = \sum_{i=1}^{N} c_i \varphi(wx + \theta_i),$$

para ciertos $c_i, w, \theta_i \in \mathbb{R}$.

Usando las propiedades de la convolución, tenemos

$$\begin{aligned} c_\alpha(x) &= (c * \psi_\alpha)(x) = (c * \varphi'_\alpha)(x) = (c' * \varphi_\alpha)(x) \\ &= \left( \left( \frac{d}{dx} \sum_{i=1}^{N} c_i H(\cdot - x_i) \right) * \varphi_\alpha \right)(x) \\ &= \left( \left( \sum_{i=1}^{N} c_i \delta_{x_i} \right) * \varphi_\alpha \right)(x) = \sum_{i=1}^{N} c_i (\delta_{x_i} * \varphi_\alpha)(x) \\ &= \sum_{i=1}^{N} c_i \varphi_\alpha(x - x_i) = \sum_{i=1}^{N} c_i \varphi\left( \frac{x - x_i}{\alpha} \right), \end{aligned}$$

lo que prueba lo deseado tomando como peso $w = \frac{1}{\alpha}$ y como sesgos $\theta_i = -\frac{x_i}{\alpha}$.

Paso 3: *Aproximación de la función simple por una función regular.* Vamos a mostrar que

$$|c(x) - c_\alpha(x)| < \frac{\varepsilon}{2}, \quad \forall x \in \mathbb{R},$$

si $\alpha$ es suficientemente chico. Para eso, usamos que $\psi_\alpha(x)$ es una densidad de probabilidad y escribimos

$$c(x) - c_\alpha(x) = \int_{\mathbb{R}} \psi_\alpha(y)(c(x) - c(x-y))\, dy.$$

Ahora a la integral en la igualdad de arriba la dividimos en la región donde $|y|$ es pequeño y el complemento

$$\begin{aligned}\int_{\mathbb{R}} \psi_\alpha(y)(c(x) - c(x-y))\, dy = &\int_{\{|y|<\delta\}} \psi_\alpha(y)(c(x) - c(x-y))\, dy \\ &+ \int_{\{|y|\geq\delta\}} \psi_\alpha(y)(c(x) - c(x-y))\, dy \\ =&I(\alpha) + II(\alpha).\end{aligned}$$

Estudiemos primero $II(\alpha)$. Para eso, usamos que $c(x)$ es acotado,

$$|c(x)| = \left|\sum_{i=1}^N c_i H(x - x_i)\right| \leq \sum_{i=1}^N |c_i| = M,$$

de donde $|c(x) - c(x-y)| \leq 2M$. Por otro lado, como $\psi_\alpha$ tiende a $\delta(x)$ cuando $\alpha \downarrow 0$, entonces $\psi_\alpha$ tiende a 0 en el conjunto $\{|y| \geq \delta\}$, de donde podemos concluir que

$$II(\alpha) \to 0, \quad \text{cuando } \alpha \downarrow 0,$$

luego existe $|II(\alpha)| < \frac{\varepsilon}{4}$ si $\alpha$ es chico.

Analicemos ahora $I(\alpha)$. Observemos que si $\delta > 0$ es suficientemente chico, tenemos que si $x \in [x_i, x_{i+1})$ entonces $x - y \in [x_{i-1}, x_{i+1})$ o $x - y \in [x_i, x_{i+1})$ para todo $|y| < \delta$. Luego, existe $j = 1, \ldots, N$ tal que

$$|c(x) - c(x-y)| \leq |c_j| < \frac{\varepsilon}{4}, \quad \forall y \in (-\delta, \delta).$$

De esta estimación, deducimos que

$$|I(\alpha)| \leq \int_{-\delta}^{\delta} \psi_\alpha(y)|c(x) - c(x-y)|\, dy < \frac{\varepsilon}{4}\int_{-\delta}^{\delta} \psi_\alpha(y)\, dy \leq \frac{\varepsilon}{4}.$$

Juntando la estimación para $I(\alpha)$ y para $II(\alpha)$ obtenemos

$$|c(x) - c_\alpha(x)| \leq |I(\alpha)| + |II(\alpha)| < \frac{\varepsilon}{4} + \frac{\varepsilon}{4} = \frac{\varepsilon}{2}.$$

Paso 4: *Combinamos los pasos anteriores para finalizar.* Simplemente, usando la desigualdad triangular, tenemos

$$\begin{aligned}|g(x) - c_\alpha(x)| &\leq |g(x) - c(x)| + |c(x) - c_\alpha(x)| \\ &< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.\end{aligned}$$

Finalmente, por el paso 2 tenemos que $c_\alpha$ es de la forma $\sum_{i=1}^N c_i \varphi(wx + \theta_i)$ y esto concluye la demostración. □

Observación 8.3.2. (i) Observemos que en el caso unidimensional, el peso $w$ no depende de la neurona $i$.

(ii) La salida de la red con función de activación sigmoide,

$$y = \sum_{i=1}^{N} c_i \varphi(wx + \theta_i),$$

se obtuvo *filtrando* la salida de la red con función de activación Heaviside

$$y = \sum_{i=1}^{N} c_i H(x - x_i),$$

a través del filtro $\varphi'_\alpha(x)$.

(iii) Los resultados de estas dos últimas secciones muestran que cualquier función continua en un intervalo compacto, puede ser aprendida en dos maneras: la Proposición 8.2.1 asegura que puede ser aproximada por funciones simples, mientras que el Teorema 8.3.1 asegura que se puede aproximar por funciones suaves. Sin embargo el resultado del Teorema 8.3.1 es más útil, dado que el aproximante es suave, lo que hace posible el uso del algoritmo de retropropagación.

## 8.4. Aprendizaje con funciones ReLU

Recordemos que la función ReLU (Rectified Linear Unit) se define como

$$ReLU(x) = xH(x) = \begin{cases} x, & \text{si } x \geq 0 \\ 0, & \text{si } x < 0, \end{cases}$$

y observemos que $ReLU'(x) = (xH(x))' = H(x) + xH'(x) = H(x) + x\delta(x) = H(x)$.

Veremos en lo que sigue que la función de salida de una red neuronal con una capa oculta, función de activación ReLU en la capa oculta y función lineal de activación en la capa de salida produce funciones que son lineales a trozos, así que nuestro primer resultado será mostrar que esta clase de funciones (las lineales a trozos) son densas en el conjunto de las funciones continuas.

LEMA 8.4.1. *Sea $g \in C([a,b])$. Entonces, dado $\varepsilon > 0$ existe una partición $a = x_0 < x_1 < \cdots < x_N = b$ de manera tal que la función $h(x)$ que pasa por los puntos $(x_i, g(x_i))$, $i = 0, \ldots, N$ y es lineal en cada intervalo $I_i = (x_{i-1}, x_i)$, $i = 1, \ldots, N$, verifica*

$$|g(x) - h(x)| < \varepsilon, \quad \forall x \in [a,b].$$

DEMOSTRACIÓN. Dado $\varepsilon > 0$, como $g$ es uniformemente continua en $[a,b]$, existe $\delta > 0$ tal que

$$|g(x) - g(y)| < \frac{\varepsilon}{2} \quad \forall x, y \in [a,b], \text{ tal que } |x - y| < \delta.$$

Tomemos $N \geq 1$ tal que $\frac{b-a}{N} < \delta$ y definimos la partición uniforme como $x_i = a + i\frac{b-a}{N}$, $i = 0, \ldots, N$.

La función lineal a trozos asociada a $g$ y a la partición $\{x_i\}_{i=0}^N$ queda entonces definida como

$$h(x) = g(x_{i-1}) + m_i(x - x_{i-i}), \quad \text{si } x \in I_i = [x_{i-1}, x_i],$$

donde la pendiente $m_i$ se define por

$$m_i = \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}.$$

Como $h$ es lineal en $I_i$, tenemos que

$$|h(x_i) - h(x)| \leq h(x_i) - h(x_{i-1})| = |g(x_i) - g(x_{i-1})| < \frac{\varepsilon}{2}, \quad \forall x \in I_i.$$

Luego, si $x \in [a, b]$, entonces existe $i = 1, \ldots, N$ tal que $x \in I_i$ y tenemos

$$\begin{aligned} |g(x) - h(x)| &\leq |g(x) - g(x_i)| + |g(x_i) - h(x)| \\ &\leq |g(x) - g(x_i)| + |h(x_i) - h(x)| \\ &< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \end{aligned}$$

Esto concluye la demostración. □

Con la ayuda de este lema, podemos demostrar que una red neuronal con una capa oculta que usa ReLU como función de activación para la capa oculta y una función lineal como función de activación para la capa de salida puede aprender cualquier función continua.

TEOREMA 8.4.2. *Consideremos una red neuronal con una capa oculta con ReLU como función de activación para esta capa oculta y función de activación lineal y sesgo $\beta$ para la capa de salida, Sea $g \in C([a, b])$ dada. Entonces, dado $\varepsilon > 0$, existen $\alpha_j, \theta_j \in \mathbb{R}$ y $N \geq 1$ tales que*

$$\left| g(x) - \sum_{j=0}^{N-1} \alpha_j ReLU(x + \theta_j) - \beta \right| < \varepsilon, \quad \forall x \in [a, b].$$

DEMOSTRACIÓN. La demostración consiste en ver que la función de salida de esta red,

$$y = G(x) = \sum_{j=0}^{N-1} \alpha_j ReLU(x + \theta_j) + \beta, \tag{8.4.1}$$

puede representar a cualquier función lineal a trozos como, por ejemplo, las consideradas en el Lema 8.4.1.

Para eso, consideramos una partición uniforme del $[0, 1]$ dada por

$$0 = x_0 < x_1 < \cdots < x_N = 1, \qquad x_j = \frac{j}{N}.$$

Vamos a mostrar que podemos elegir los parámetros $\alpha_j$, $x_j$ y $\beta$ de manera tal que la función dada por (8.4.1) sea la función lineal a trozos definida en el Lema 8.4.1.

Empecemos eligiendo los sesgos $\theta_j = -x_j$ y notemos que

$$ReLU(x_k - x_j) = \begin{cases} 0, & \text{si } k \leq j \\ \frac{k-j}{N}, & \text{si } k > j, \end{cases}$$

luego, la función $G(x)$ dada por (8.4.1) evaluada en los puntos de la partición $x_k$ toma los valores

$$\begin{aligned} G(x_k) &= \sum_{j=0}^{N-1} \alpha_j ReLU(x_k - x_j) + \beta \\ &= \frac{1}{N} \sum_{j<k} \alpha_j (k-j) + \beta. \end{aligned}$$

Luego, debemos determinar los $N+1$ parámetros, $\alpha_j$, $j = 0, \ldots, N-1$ y $\beta$ de manera tal que $G(x_k) = g(x_k)$, $k = 0, \ldots, N$. Luego, si llamamos $y_k = g(x_k)$ obtenemos el sistema de ecuaciones

$$\begin{cases} y_0 = G(x_0) = G(0) = \beta \\ y_1 = G(x_1) = \frac{1}{N}\alpha_0(1-0) + \beta \\ y_2 = G(x_2) = \frac{1}{N}(\alpha_0(2-0) + \alpha_1(2-1)) + \beta \\ \cdots \\ y_N = G(x_N) = \frac{1}{N}(\alpha_0(N-0) + \cdots + \alpha_{N-1}(N-(N-1)) + \beta. \end{cases}$$

Este sistema tiene una única solución dada por

$$\begin{cases} \beta = y_0 \\ \alpha_0 = N(y_1 - y_0) \\ \alpha_1 = N(y_2 - 2y_1 + y_0) \\ \cdots \\ \alpha_{N-1} = N(y_N - 2y_{N-1} + y_{N-2}). \end{cases}$$

Finalmente, observemos que la función $G(x)$ dada por (8.4.1) es lineal a trozos, dado que su derivada

$$G'(x) = \sum_{j=0}^{N-1} \alpha_j ReLU'(x - x_j) = \sum_{j=0}^{n-1} \alpha_j H(x - x_j),$$

es constante en cada intervalo $(x_i, x_{i+1})$. Luego, el resultado se sigue del Lema 8.4.1. □

## 8.5. Aprendizaje con Softplus

La función de activación *softplus* se define como

$$\varphi(x) = sp(x) = \ln(1 + e^x).$$

Es una versión suave de la función $ReLU(x)$. Esto se puede formalizar en una proposición.

PROPOSICIÓN 8.5.1. *La función softplus, se puede escribir como*

$$sp(x) = (ReLU * K)(x) = \int_{-\infty}^{\infty} ReLU(y) K(x-y)\, dy,$$

*donde el* núcleo de convolución $K$ *viene dado por*

$$K(x) = \frac{1}{(1+e^x)(1+e^{-x})}.$$

DEMOSTRACIÓN. Observemos primero que $K(x)$ es una función positiva que decrece a 0 exponencialmente rápido cuando $x \to \pm\infty$. Esta propiedad hace que la convolución y las integrales impropias que siguen estén todas bien definidas. Ahora, observemos que el núcleo $K$ es la derivada de la función logística $\sigma(x) = (1 + e^{-x})^{-1}$, i.e. $K(x) = \sigma'(x)$. Por otro lado, $sp(x)$ es la primitiva de $\sigma(x)$, es decir $sp'(x) = \sigma(x)$, luego,

$$sp(x) = \int_{-\infty}^{x} \sigma(t)\, dt.$$

Con estas observaciones, ya podemos calcular el producto de convolución:

$$\begin{aligned}
(ReLU(x) * K)(x) &= \int_{-\infty}^{\infty} ReLU(y)K(x-y)\, dy = \int_{0}^{\infty} yK(x.y)\, dy \\
&= \int_{-\infty}^{x} (x-t)K(t)\, dt = x\int_{-\infty}^{x} K(t)\, dt - \int_{-\infty}^{x} tK(t)\, dt \\
&= x\sigma(x) - \int_{-\infty}^{x} t\sigma'(x)\, dt \\
&= x\sigma(x) - \left( t\sigma(x)\Big|_{-\infty}^{x} - \int_{-\infty}^{x} \sigma(t)\, dt \right) \\
&= x\sigma(x) - (x\sigma(x) - sp(x)) = sp(x),
\end{aligned}$$

lo que prueba el resultado. □

OBSERVACIÓN 8.5.2. Una prueba alternativa y *formal* de esta proposición, puede hacerse usando las relaciones

$$ReLU'(x) = H(x), \quad H'(x) = \delta(x), \quad sp'(x) = \sigma(x), \quad \sigma'(x) = K(x),$$

y las fórmulas

$$(f * g)'(x) = (f' * g)(x) = (f * g')(x).$$

Efectivamente, si llamamos $f(x) = (ReLU * K)(x) - sp(x)$, entonces tenemos que $f(-\infty) = 0$. Por otro lado,

$$\begin{aligned}
f(x) &= (ReLU * K)'(x) - sp'(x) = (ReLU' * K)(x) - \sigma(x) \\
&= (H * \sigma')(x) - \sigma(x) = (H' * \sigma)(x) - \sigma(x) \\
&= (\delta * \sigma)(x) - \sigma(x) = \sigma(x) - \sigma(x) = 0.
\end{aligned}$$

Luego $f(x) = 0$ para todo $x \in \mathbb{R}$.

Veamos las propiedades del núcleo de convolución $K(x)$.

PROPOSICIÓN 8.5.3. *El núcleo de convolución $K(x)$ definido en la Proposición 8.5.1 es una densidad de probabilidad simétrica en $\mathbb{R}$.*

*Por otro lado, dado $\alpha > 0$, consideramos los núcleos rescalados $K_\alpha(x) = \frac{1}{\alpha}K(\frac{x}{\alpha})$. Entonces $K_\alpha$ son densidades de probabilidad simétricas en $\mathbb{R}$ y las medidas de probabilidad $d\mu_\alpha = K_\alpha\, dx$ convergen a $\delta$ cuando $\alpha \downarrow 0$ en sentido débil.*

DEMOSTRACIÓN. De la expresión $K(x) = (1 + e^{x})^{-1}(1 + e^{-x})^{-1}$ es evidente que $K(x) = K(-x)$ y $K(x) > 0$. Por otro lado,

$$\int_{-\infty}^{\infty} K(x)\, dx = \int_{-\infty}^{\infty} \sigma'(x)\, dx = \sigma(\infty) - \sigma(-\infty) = 1.$$

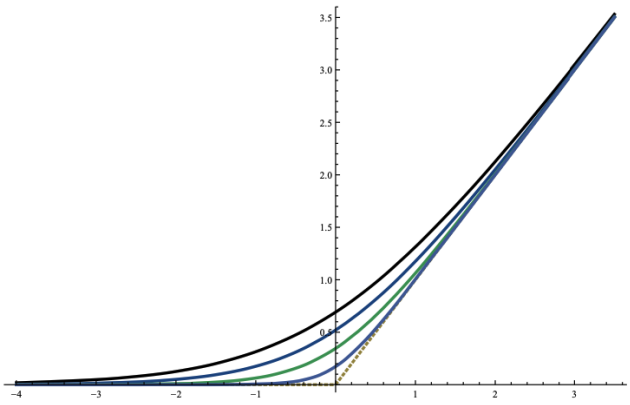



FIGURA 8.2. Funciones softplus rescaladas $\varphi_\alpha(x)$ para diferentes valores de $\alpha$.

Ahora, usando la sustitución $x = \alpha y$ es fácil ver que

$$\int_{-\infty}^{\infty} K_\alpha(x)\,dx = \int_{-\infty}^{\infty} K(x)\,dx = 1.$$

La demostración de que $d\mu_\alpha = K_\alpha\,dx \to d\delta$ cuando $\alpha \downarrow 0$ es análoga a la de la Proposición 8.1.1. □

Si ahora definimos las funciones softplus rescaladas

$$\varphi_\alpha(x) = \alpha sp(\frac{x}{\alpha}) = \alpha \ln(1 + e^{x/\alpha}),$$

usando las relaciones $sp'(x) = \sigma(x)$ y $\sigma'(x) = K(x)$ obtenemos que

$$\varphi''_\alpha(x) = K_\alpha(x).$$

Es sencillo ver que las funciones rescaladas $\varphi_\alpha(x)$ convergen monótonamente a $ReLU(x)$ (ver Figura 8.5)

El siguiente resultado nos provee de una fórmula para la regularización de la función lineal a trozos $G(x)$ dada por el Teorema 8.4.2 cuando la misma es pasada por el filtro $K_\alpha$.

LEMA 8.5.4. *Consideremos la convolución $G_\alpha(x) = (G * K_\alpha)(x)$, donde*

$$G(x) = \sum_{j=0}^{N-1} \alpha_j ReLU(x - x_j) + \beta.$$

*Entonces, existen parámetros $c_j, w$ y $\theta_j$ tales que*

$$G_\alpha(x) = \sum_{j=0}^{N-1} c_j sp(wx - \theta_j).$$

DEMOSTRACIÓN. La demostración consiste simplemente en calcular $G_\alpha(x)$. En efecto,

$$\begin{aligned}
G_\alpha(x) = (G * K_\alpha)(x) &= \sum_{j=0}^{N-1} \alpha_j (ReLU(\cdot - x_j) * K_\alpha)(x) + \beta \\
&= \sum_{j=0}^{N-1} \alpha_j (ReLU(\cdot - x_j) * \varphi_\alpha'')(x) + \beta \\
&= \sum_{j=0}^{N-1} \alpha_j (ReLU(\cdot - x_j) * \varphi_\alpha)''(x) + \beta \\
&= \sum_{j=0}^{N-1} \alpha_j (ReLU(\cdot - x_j)'' * \varphi_\alpha)(x) + \beta \\
&= \sum_{j=0}^{N-1} \alpha_j (\delta(\cdot - x_j) * \varphi_\alpha)(x) + \beta = \sum_{j=0}^{N-1} \alpha_j \varphi_\alpha(x - x_j) + \beta \\
&= \sum_{j=0}^{N-1} \alpha_j \alpha \varphi \left( \frac{x - x_j}{\alpha} \right) + \beta = \sum_{j=0}^{N-1} c_j sp(x - \theta_j) + \beta,
\end{aligned}$$

donde $c_j = \alpha_j \alpha$, $w = 1/\alpha$ y $\theta_j = x_j/\alpha$. □

OBSERVACIÓN 8.5.5. La función $G_\alpha$ es la salida de una red neuronal con una capa oculta con función de activación softplus para la misma y función de activación lineal para la capa de salida.

Como consecuencia de la Proposición 8.5.3, el Lema 8.5.4 y el Teorema de Dini, Teorema 7.1.1, obtenemos:

COROLARIO 8.5.6. *La sucesión $G_\alpha$ definida en el Lema 8.5.4 converge uniformemente a $G$ cuando $\alpha \downarrow 0$.*

DEMOSTRACIÓN. Como hemos observado, $\varphi_\alpha(x) \downarrow ReLU(x)$ cuando $\alpha \downarrow 0$ (ver el Ejercicio 8.7.4) por lo que $G_\alpha(x) \downarrow G(x)$ para todo $x \in [0,1]$. Aplicando ahora el Teorema 7.1.1 se concluye lo deseado. □

Combinando ahora la Proposición 8.5.3, el Lema 8.5.4 y el Teorema 8.4.2, obtenemos el siguiente resultado:

TEOREMA 8.5.7. *Consideremos una red neuronal con una capa escondida con función de activación softplus y función de activación lineal en la capa de salida. Sea $g \in C([0,1])$. Entonces dado $\varepsilon > 0$, existen $N \geq 1$, $c_j, w, \theta_j, \beta \in \mathbb{R}$, $j = 0, \ldots, N-1$, tales que*

$$\left| g(x) - \sum_{j=0}^{N-1} c_j sp(wx - \theta_j) - \beta \right| < \varepsilon, \quad \forall x \in [0,1].$$

DEMOSTRACIÓN. Sea $\varepsilon > 0$. Por el Teorema 8.4.2, existe $G(x) = \sum_{j=0}^{N-1} \alpha_j ReLU(x - x_j) + \beta$ tal que

$$|g(x) - G(x)| < \frac{\varepsilon}{2}, \quad \forall x \in [0, 1].$$

El Corolario 8.5.6 implica que existe $\alpha > 0$ suficientemente chico ral que

$$|G(x) - G_\alpha(x)| < \frac{\varepsilon}{2}, \quad \forall x \in [0, 1].$$

Luego, usando la desigualdad triangular junto con el Lema 8.5.4 se concluye lo pedido. □

## 8.6. Resumen

Este capítulo muestra que las redes neuronales de una capa oculta con entrada unidimensional, $x \in [0, 1]$, pueden aprender funciones continuas en $C([0, 1])$. Probamos este resultado en los casos en que la función de activación es una función escalón, una sigmoide, una ReLU o una función Softplus.

El resultado de una red con función de activación sigmoide es una versión suavizada de una red con función de activación escalón. De manera similar, el aprendizaje con Softplus es una versión suavizada del aprendizaje con ReLU. Es importante notar la razón matemática detrás del hecho, observado experimentalmente, de que el uso de la función de activación sigmoide produce errores más pequeños que el uso de funciones escalón. Del mismo modo, en muchos casos es preferible utilizar la función de activación Softplus en lugar de ReLU.

La importancia práctica del método radica en la capacidad de la red para generalizar bien cuando se aplica a datos discretos. La red potencialmente aprende una función continua que subyace a los datos dados.

## 8.7. Ejercicios

EJERCICIO 8.7.1 (Derivada generalizada). Decimos que $f^{(n)} = g$ en sentido generalizado (o sentido débil), si para toda función suave de soporte compacto $\phi \in C_c^\infty(\mathbb{R})$ se tiene

$$(-1)^n \int_{\mathbb{R}} f(x)\phi^{(n)}(x)\, dx = \int_{\mathbb{R}} g(x)\phi(x)\, dx.$$

Probar las siguientes relaciones en sentido generalizado:

(a) $H'(x - x_0) = \delta(x - x_0)$;
(b) $ReLU'(x) = H(x)$;
(c) $ReLU''(x) = \delta(x)$;
(d) $(xReLU(x))' = 2ReLU(x)$;
(e) $(xReLU(x))'' = 2H(x)$.

EJERCICIO 8.7.2 (Continuidad uniforme). Sea $g\colon [a, b] \to \mathbb{R}$ una función continua. Probar que es *uniformemente continua*.

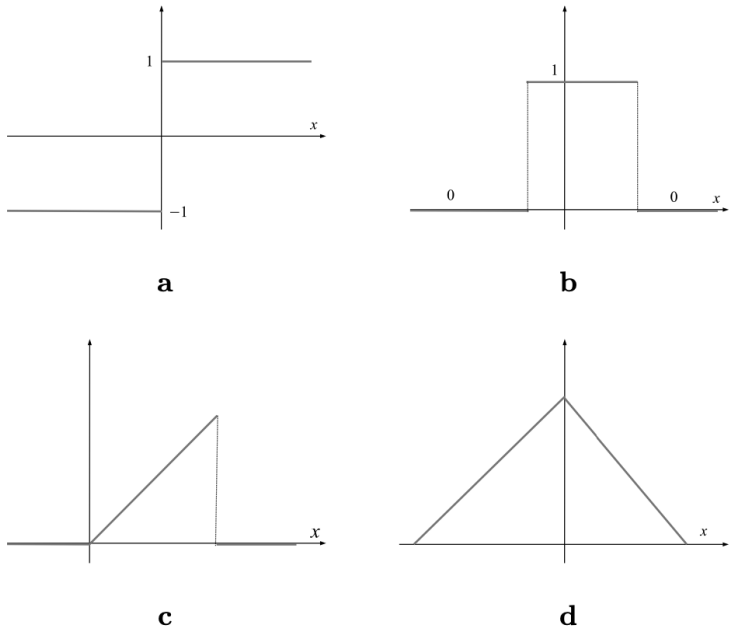



FIGURA 8.3. **a**. Función de salto bipolar. **b**. Función cajón. **c**. Función diente de sable. **d**. Función diente de tiburón.

EJERCICIO 8.7.3. Sea $f$ una de las funciones de la Figura 8.7. ¿Cuál de ellas tiene la propiedad de que la familia

$$\mathcal{F} = \left\{ \sum_{i=1}^{N} \alpha_i f(w_i x - b_i) \colon N \geq 1,\ \alpha_i, w_i, b_i \in \mathbb{R} \right\}$$

es densa en $C([0,1])$?

EJERCICIO 8.7.4. Sea $\varphi_\alpha(x) = \alpha \ln(1 + e^{x/\alpha})$. Probar que $\varphi_\alpha(x) \downarrow ReLU(x)$ cuando $\alpha \downarrow 0$.

Capítulo 9

# Aproximadores universales

La respuesta a la pregunta "¿Por qué las redes neuronales funcionan tan bien en la práctica?" se basa, sin duda, en el hecho de que las redes neuronales pueden aproximar con gran precisión una amplia familia de funciones del mundo real que dependen de variables de entrada. El objetivo de este capítulo es proporcionar demostraciones matemáticas de este comportamiento para diferentes tipos de funciones objetivo.

El hecho de que las redes neuronales actúen como aproximadores universales significa que su salida es una aproximación precisa, con cualquier grado de exactitud deseado, de funciones en ciertos espacios de funciones bien conocidos. El proceso de obtener una aproximación precisa se interpreta como el aprendizaje de una función objetivo dada.

Este capítulo trata sobre la existencia del proceso de aprendizaje, pero no proporciona ninguna construcción explícita de los pesos de la red. La idea subyacente es que tiene sentido saber primero que una solución existe antes de comenzar a buscarla.

La solución deseada existe siempre que se considere un número suficiente de neuronas en la capa oculta y la relación entre las variables de entrada y la función objetivo sea determinista. Para obtener la solución, se debe emplear un algoritmo de aprendizaje adecuado. Esta técnica garantiza el éxito del uso de redes neuronales en muchas aplicaciones prácticas.

## 9.1. Ejemplos introductorios

Una red neuronal, como hemos visto, tiene la capacidad de aprender de los datos. Suponiendo que se tiene una función objetivo $z = f(x)$, la red tiene que aprender esta función cambiando sus parámetros internos. La mayoría de las veces, el resultado de este aprendizaje es sólo una aproximación aceptable. Veamos algunos ejemplos.

Ejemplo 9.1.1 (Aprendiendo funciones continuas). Supongamos que queremos armar una red neuronal para aprender la ley de gravitación universal de Newton. Esto es

$$f(m_1, m_2, d) = \frac{km_1m_2}{d^2},$$

es decir, la fuerza entre dos cuerpos de masas $m_1$ y $m_2$ es inversamente proporcional al cuadrado de la distancia entre ellos.

Construimos una red con entradas $(x_1, x_2, x_3) = (m_1, m_2, d)$ que puede tomar valores continuos en una cierta región $K \subset \mathbb{R}^3$. La variable objetivo, que tiene que ser aprendida, es $z = f(m_1, m_2, d)$, con $f\colon K \to \mathbb{R}$ una función continua. La red tendrá salida $y = g(m_1, m_2, d, w, b)$, que depende tanto de las entradas como de los parámetros (pesos $w$ y sesgos $b$).

Luego, si uno provee las masas de dos cuerpos, junto con la distancia que los separa, la red debe ser capaz de proporcionar la fuerza entre ellos con una precisión mejor que

el error de tolerancia prefijado. En el Teorema 9.3.16 demostraremos la existencia de esta red.

EJEMPLO 9.1.2 (Aprendiendo funciones de energía finita). Una señal de audio tiene que ser aprendida por una red neuronal. Es habitual en la teoría asumir que la señal es de *energía finita*, es decir que la señal pertenece al espacio $L^2$.

La entrada de la red es $x = (t, \nu)$ donde $t$ es el tiempo y $\nu$ es la frecuencia de la señal. La función objetivo, $z = f(x) = f(t, \nu)$ provee la amplitud en términos del tiempo y de la frecuencia. La salida de la red, $y = g(t, \nu, w, b)$ nos da una aproximación del objetivo que puede ser ajustado variando los parámetros $w$ y el sesgo $b$. La existencia de una red neuronal con esta propiedad la veremos en el Corolario 9.3.21.

## 9.2. Planteo general

El concepto de red neuronal como aproximador universal es el siguiente. Sea $x$ la variable de entrada y $z$ la variable objetivo. La función objetivo es entonces $z = f(x)$ donde $f$ es una cierta función que pertenece a un espacio $E$ de funciones, un espacio métrico con distancia $d$, i.e. $f \in (E, d)$.

El conjunto de funciones que son salidas posibles de la red neuronal es un espacio que notamos por $U$ y se asume que $U \subset E$. De hecho, se elige la función de activación de la red de manera tal que la función de salida quede definida dentro del espacio $E$.

Luego, decimos que la red neuronal es un *aproximador universal* para el espacio $(E, d)$ si el conjunto de salida $U$ es denso en $E$, i.e.

$$\forall f \in E,\ \forall \varepsilon > 0,\ \exists g \in U \text{ tal que } d(f, g) < \varepsilon.$$

El proceso por el que la red neuronal aproxima a la función objetivo $f \in E$ por funciones $g \in U$ se llama el aprendizaje y al conjunto $U$ se lo llama el espacio aproximante del espacio objetivo $E$.

Veamos algunos de los ejemplos más usuales de espacios objetivos $(Ed)$.

EJEMPLO 9.2.1 (Espacio de funciones continuas). El primer ejemplo es $E = C(K)$ donde $K \subset \mathbb{R}^n$ es un conjunto compacto. La distancia en este espacio es

$$d(f, g) = \sup_{x \in K} |f(x) - g(x)|.$$

Notemos que $(E, d)$ resulta un espacio métrico completo. De hecho, la métrica $d$ es inducida por una norma

$$\|f\| = \sup_{x \in K} |f(x)|.$$

Luego, $d(f.g) = \|f - g\|$.

Es habitual notar por $\|\cdot\|_\infty$ y $d_\infty(\cdot, \cdot)$ a esta norma y métrica respectivamente.

EJEMPLO 9.2.2 (Espacio de funciones integrables). En este caso $E = L^1(K)$ es el espacio de funciones integrables sobre $K$. La métrica en este caso viene dada por

$$d(f, g) = \int_K |f(x) - g(x)|\, dx.$$

Esta métrica mide el área encerrada entre los gráficos de las funciones sobre $K$. Al igual que en el ejemplo anterior, esta métrica viene inducida por una norma

$$\|f\| = \int_K |f(x)|\, dx.$$

Esta norma y métrica se notan habitualmente como $\|\cdot\|_1$ y $d_1(\cdot,\cdot)$ respectivamente.

Ejemplo 9.2.3 (Espacio de funciones de energía finita). Este espacio es $E = L^2(K)$ y se define como

$$L^2(K) = \left\{ f\colon K \to \mathbb{R}\colon \int_K |f(x)|^2\, dx < \infty \right\}.$$

La métrica viene dada por

$$d(f,g) = d_2(f,g) = \int_K |f(x) - g(x)|^2\, dx.$$

Esta métrica se interpreta como la diferencia de energía entre dos señales de energía finita $f$ y $g$. Al igual que en los ejemplos anteriores, esta métrica es inducida por una norma:

$$\|f\| = \|f\|_2 = \left( \int_K |f(x)|^2\, dx \right)^{\frac{1}{2}}.$$

Ejemplo 9.2.4 (Espacios discretos). En este caso consideramos un conjunto finito de puntos $K = \{x_1, \ldots, x_r\} \subset \mathbb{R}^n$ y consideramos el conjunto $E = \{f\colon K \to \mathbb{R}\}$ al conjunto de funciones reales definidas sobre $K$. En este conjunto se pueden definir diferentes métricas, por ejemplo

$$d_s(f,g) = \left( \sum_{x_j \in K} |f(x_j) - g(x_j)|^2 \right)^{\frac{1}{2}},$$

que es la distancia Euclídea de puntos en $\mathbb{R}^n$, o

$$d_a(f,g) = \sum_{x_j \in K} |f(x_j) - g(x_j)|,$$

que es el error absoluto.

## 9.3. Redes con una capa oculta

Comenzaremos el estudio del proceso de aprendizaje en el caso más simple en el que la red tiene una única capa oculta.

En esta sección mostraremos que una red neuronal con una capa oculta puede ser usada como un aproximador para los espacios $C(I_n)$, $L^1(I_n)$ y $L^2(I_n)$, domde $I_n = [0,1] \times \cdots \times [0,1] \subset \mathbb{R}^n$ es un hipercubo.

**9.3.1. Aprendiendo funciones continuas $f \in C(I_n)$.** Para poder desarrollar los resultados necesitaremos unos resultados preliminares de Análisis Funcional. Este resultado se lo conoce como el *lema de separación.*

LEMA 9.3.1. *Sea $E$ un espacio normado, $U \subset E$ un subespacio y $f_0 \in E$ tal que*

$$d(f_0, U) \geq \delta,$$

*para algún $\delta > 0$. Luego, existe un funcional $L\colon E \to \mathbb{R}$ lineal y continuo tal que*

(a) $\|L\| \leq 1$ *(i.e.* $\sup_{\|f\|\leq 1} |L(f)| \leq 1$*).*
(b) $L(u) = 0$ *para todo $u \in U$.*
(c) $L(f) = \delta$.

Este lema es una consecuencia relativamente inmediata del Teorema de Hahn-Banach que se estudia en el curso de Análisis Funcional. Pueden encontrar una demostración del Teorema de Hahn-Banach, por ejemplo, en [**4**].

El Teorema de Hah-Banach dice:

TEOREMA 9.3.2 (Hahn-Banach). *Sea $E$ un espacio vectorial normado, con norma $\|\cdot\|$ y $U \subset E$ un subespacio. Asumamos que tenemos una función $\ell\colon U \to \mathbb{R}$ lineal y acotada. Es decir que existe $M > 0$ tal que*

$$|\ell(u)| \leq M\|u\|, \quad \forall u \in U.$$

*Entonces existe $L\colon E \to \mathbb{R}$ lineal que extiende a $\ell$ (es decir, $L(u) = \ell(u)$ para $u \in U$) tal que*

$$|L(f)| \leq M\|f\|, \quad \forall f \in E.$$

OBSERVACIÓN 9.3.3. El Teorema de Hahn-Banach es más general que el que enunciamos acá. Sin embargo esta versión será suficiente para nuestros propósitos.

Con la ayuda del Teorema de Hahn-Banach, resulta sencillo demostrar el Lema 9.3.1.

DEMOSTRACIÓN DEL LEMA 9.3.1. Sea $T$ el subespacio lineal generado por $U$ y $f_0$,

$$T = \{u + \lambda f_0 \colon u \in U,\ \lambda \in \mathbb{R}\},$$

y definamos el funcional $\ell\colon T \to \mathbb{R}$ como

$$\ell(u + \lambda f_0) = \lambda\delta.$$

Resulta evidente que $\ell$ es lineal. Veamos que es acotado. Para eso, observamos que

$$\|u + \lambda f_0\| = |\lambda|\|f_0 - \left(-\tfrac{u}{\lambda}\right)\| \geq |\lambda|\delta,$$

donde hemos usado que $-\frac{u}{\lambda} \in U$ y la hipótesis sobre $f_0$.

Luego,

$$|\ell(u + \lambda f_0)| = |\lambda|\delta \leq \|u + \lambda f_0\|.$$

Es decir, $\ell$ resulta acotado en $T$ con constante $M = 1$.

Podemos entonces aplicar el Teorema de Hahn-Banach, Teorema 9.3.2, y concluir que existe $L\colon E \to \mathbb{R}$ lineal tal que

$$L(u) = \ell(u) = 0, \quad \forall u \in U, \quad L(f_0) = \ell(0 + 1 \cdot f_0) = 1 \quad \text{y} \quad |L(f)| \leq \|f\|, \quad \forall f \in E,$$

lo que prueba el Lema. □

La importancia del Lema 9.3.1 es la de proporcionar una herramienta para determinar si un subespacio $U$ de un espacio normado $E$ resulta denso o no.

COROLARIO 9.3.4. *Sea $E$ un espacio normado y $U \subset E$ un subespacio. Entonces $U$ es denso en $E$ si y sólo si el único funcional $L\colon E \to \mathbb{R}$ lineal y acotado que verifica que $L(u) = 0$ para todo $u \in U$ es $L = 0$.*

DEMOSTRACIÓN. Es claro que si $U$ es denso en $E$ y $L|_U = 0$, como $L$ es continua, sigue que $L = 0$.

Para el recíproco, asumamos por el absurdo que $U$ no es denso, entonces existe $f_0 \in E$ y $\delta > 0$ tal que $d(f_0, U) \geq \delta$. Pero por el Lema 9.3.1, existe $L\colon E \to \mathbb{R}$ lineal y acotada tal que $L|_U = 0$ y $L(f_0) = \delta \neq 0$ lo que es una contradicción. □

Para poder aplicar los resultados de aproximación al espacio de funciones continuas, necesitamos entender cómo son los funcionales definidos sobre $C(K)$, donde $K \subset \mathbb{R}^n$ es un conjunto compacto. Empecemos con unas definiciones.

DEFINICIÓN 9.3.5. Sea $X$ un conjunto no vacío, una subconjunto $\Sigma$ de partes de $X$, $\Sigma \subset \mathcal{P}(X)$, se dice una $\sigma-$álgebra si verifica

(i) $\emptyset, X \in \Sigma$;
(ii) $\{A_i\}_{i\in\mathbb{N}} \subset \Sigma \Longrightarrow A = \bigcup_{i=1}^{\infty} A_i \in \Sigma$;
(iii) $A \in \Sigma \Longrightarrow A^c = X \setminus A \in \Sigma$.

Es muy simple ver que si $\Sigma$ es una $\sigma-$álgebra, entonces la misma es cerrada por intersecciones numerables.

DEFINICIÓN 9.3.6. Dado un conjunto $X$ no vacío y una $\sigma-$álgebra $\Sigma \subset \mathcal{P}(X)$, dedimos que $(X, \Sigma)$ es un espacio medible.

DEFINICIÓN 9.3.7. Si tenemos un espacio medible $(X, \Sigma)$, una medida (finita) sobre $(X, \Sigma)$ es una función $\mu\colon \Sigma \to [0, \infty)$ que verifica

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i),$$

para toda colección $\{A_i\}_{i\in\mathbb{N}} \subset \Sigma$ que verifica $A_i \cap A_j = \emptyset$ si $i \neq j$.

OBSERVACIÓN 9.3.8. Si a la medida $\mu$ se le pide adicionalmente que $\mu(X) = 1$ se la llama *medida de probabilidad*.

Ahora vamos a necesitar el concepto de medida con signo.

DEFINICIÓN 9.3.9. Dado un espacio medible $(X, \Sigma)$, una medida signada es una función $\nu\colon \Sigma \to \mathbb{R}$ que verifica

$$\nu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \nu(A_i),$$

para toda colección $\{A_i\}_{i\in\mathbb{N}} \subset \Sigma$ que verifica $A_i \cap A_j = \emptyset$ si $i \neq j$ tal que la suma de la derecha sea absolutamente convergente.

Sobre medidas signadas tenemos el siguiente teorema fundamental que las caracterizan completamente en términos de medidas. No vamos a hacer esta demostración, la misma se estudia en el curso de Análisis Real/Medida y Probabilidad. Pueden encontrar la demostración en el libro [**21**].

TEOREMA 9.3.10 (de descomposición de Hahn). *Sea $\nu$ una medida signada definida en el espacio medible $(X, \Sigma)$. Entonces existe un conjunto $P \in \Sigma$ tal que*

$$\nu(A \cap P) \geq 0, \quad \nu(A \setminus P) \leq 0 \quad \forall A \in \Sigma.$$

*En otras palabras,*

$$\nu^+(A) = \nu(A \cap P) \quad y \quad \nu^-(A) = -\nu(A \setminus P)$$

*son dos medidas y $\nu = \nu^+ - \nu^-$.*

OBSERVACIÓN 9.3.11. Al conjunto $P$ del Teorema de descomposición de Hahn se lo llama *conjunto de positividad* de $\nu$.

OBSERVACIÓN 9.3.12. La descomposición de la medida signada $\nu = \nu^+ - \nu^-$ se la llama *descomposición de Jordan*. A la medida $|\nu| = \nu^+ + \nu^-$ se la conoce como el *valor absoluto* de la medida $\nu$.

Con estas definiciones podemos enunciar el siguiente teorema de representación de funcionales lineales y continuos sobre $C(K)$.

TEOREMA 9.3.13. *Sea $L\colon C(K) \to \mathbb{R}$ un funcional lineal y continuo. Entonces existe una única medida signada $\nu$ sobre los conjuntos de Borel de $K$ tal que*

$$L(f) = \int_K f(x)\, d\nu(x), \quad \forall f \in C(K).$$

*Más aún, $\|L\| = |\nu|(K)$.*

Combinando el Corolario 9.3.4 junto con el Teorema 9.3.13, obtenemos

LEMA 9.3.14. *Sea $U \subset C(I_n)$ un subespacio lineal. Entonces $U$ es denso en $C(K)$ si y sólo si la única medida signada $\nu$ definida en la $\sigma-$álgebra de Borel de $K$ que verifica*

$$\int_K u(x)\, d\nu(x) = 0, \quad \forall u \in U$$

*es la medida nula, $\nu = 0$.*

Necesitamos finalmente un lema técnico sobre medidas signadas. La demostración de este lema es compleja y precisa sobre el final del argumento conocimientos sobre la transformada de Fourier. Si no está familiarizado con el tema, puede saltear la prueba y sólo usar el resultado.

LEMA 9.3.15. *Sea $\nu$ una medida de Borel signada sobre $I_n$ tal que $\nu(H^+_{\mathbf{w},\theta}) = 0$ y $\nu(P_{\mathbf{w},\theta}) = 0$ para todo $\mathbf{w} \in \mathbb{R}^n$ y $\theta \in \mathbb{R}$, donde*

$$H^+_{\mathbf{w},\theta} = \{x \in I_n\colon \mathbf{w} \cdot x + \theta > 0\} \quad y \quad P_{\mathbf{w},\theta} = \{x \in I_n\colon \mathbf{w} \cdot x + \theta = 0\}.$$

*Entonces $\nu = 0$.*

DEMOSTRACIÓN. Consideremos entonces la aplicación $F\colon L^\infty(\mathbb{R}, d|\nu|) \to \mathbb{R}$ definida por

$$F(h) = \int_{I_n} h(\mathbf{w}\cdot x)\, d\nu(x).$$

El funcional $F$ es lineal y acotado, dado que

$$|F(h)| \leq |\nu|(I_n)\|h\|_\infty,$$

donde la norma $\|\cdot\|_\infty$ es tomada con respecto a la medida $|\nu|$. De esta acotación se deduce que $F$ es continua.

Consideremos ahora $h(x) = \mathbf{1}_{[\theta,\infty)}(x)$, luego $F(h) = \nu(H^+_{\mathbf{w},-\theta}) + \nu(P_{\mathbf{w},-\theta}) = 0$. Razonando de manera análoga, $F(\mathbf{1}_{(\theta,\infty)}) = 0$ y de acá se deduce que

$$F(\mathbf{1}_I) = 0, \quad \forall I \text{ intervalo de } \mathbb{R}.$$

Usando la linealidad de $F$ obtenemos entonces que si $s(x)$ es una función escalón

$$s(x) = \sum_{i=1}^{N} \alpha_i \mathbf{1}_{I_i}(x),$$

entonces $F(s) = 0$.

Si ahora consideramos $g \in C_b(\mathbb{R})$ (continua y acotada), existe $\{s_k\}_{k\in\mathbb{N}}$ sucesión de funciones escalón, tal que $\|s_k - g\|_\infty \to 0$ cuando $k \to 0$ y por la continuidad de $F$ obtenemos que

$$F(g) = \lim_{k\to\infty} F(s_k) = 0.$$

Con esto podemos calcular la transformada de Fourier de la medida $\nu$:

$$\begin{aligned}\hat{\nu}(\mathbf{w}) &= \int_{I_n} e^{-i\mathbf{w}\cdot x}\, d\nu(x) = \int_{I_n} \cos(\mathbf{w}\cdot x)\, d\nu(x) - i\int_{I_n} \sin(\mathbf{w}\cdot x)\, d\nu(x)\\ &= F(\cos(\cdot)) - iF(\sin(\cdot)) = 0, \quad \forall \mathbf{w} \in \mathbb{R}^n.\end{aligned}$$

Como la transformada de Fourier es inyectiva, concluimos que $\nu = 0$. □

Bueno, con estos preliminares podemos dar ahora una extensión del Teorema 8.3.1 al caso $n-$dimensional.

TEOREMA 9.3.16 (Cybenko, 1989). *Sea $\sigma$ una función continua de tipo sigmoide (es decir, verifica las hipótesis de la Proposición 8.1.1), entonces el conjunto $U$ definido por*

$$U = \left\{ \sum_{j=1}^{N} \alpha_j \sigma(\mathbf{w}_j \cdot x + \theta_j)\colon \mathbf{w}_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}, N \geq 1 \right\}$$

*es denso en $C(I_n)$.*

DEMOSTRACIÓN. Esta demostración es algo técnica y usa a la transformada de Fourier sobre el final. Si no está familiarizado con este concepto puede saltear la prueba.

Usando el Lema 9.3.14, alcanza con ver que si

$$\int_{I_n} \sigma(\mathbf{w}\cdot x + \theta)\, d\nu(x) = 0, \quad \forall \mathbf{w} \in \mathbb{R}^n, \theta \in \mathbb{R},$$

entonces $\nu = 0$ donde $\nu$ es una medida signada de Borel en $I_n$.

Fijemos $\mathbf{w} \in \mathbb{R}^n$ y $\theta \in \mathbb{R}$ y para $\lambda, \phi \in \mathbb{R}$ definimos

$$\sigma_\lambda(x) = \sigma(\lambda(\mathbf{w} \cdot x + \theta) + \phi).$$

Usando las propiedades de $\sigma$ (ver (ii)-(iii) en la Proposición 8.1.1), tenemos que

$$\lim_{\lambda \to \infty} \sigma_\lambda(x) = \begin{cases} 1, & \text{si } \mathbf{w} \cdot x + \theta > 0 \\ 0, & \text{si } \mathbf{w} \cdot x + \theta < 0 \\ \sigma(\phi), & \text{si } \mathbf{w} \cdot x + \theta = 0. \end{cases}$$

Introducimos la notación

$$\begin{aligned} H^+_{\mathbf{w},\theta} &= \{x \in I_n \colon \mathbf{w} \cdot x + \theta > 0\}, \\ H^-_{\mathbf{w},\theta} &= \{x \in I_n \colon \mathbf{w} \cdot x + \theta < 0\}, \\ P_{\mathbf{w},\theta} &= \{x \in I_n \colon \mathbf{w} \cdot x + \theta = 0\}, \end{aligned}$$

y si definimos a la función

$$\gamma(x) = \begin{cases} 1, & \text{si } x \in H^+_{\mathbf{w},\theta} \\ 0, & \text{si } x \in H^-_{\mathbf{w},\theta} \\ \sigma(\phi), & \text{si } x \in P_{\mathbf{w},\theta}, \end{cases}$$

tenemos que $\sigma_\lambda(x) \to \gamma(x)$ puntualmente cuando $\lambda \to \infty$. Ahora usamos el teorema de convergencia mayorada y obtenemos

$$\begin{aligned} 0 &= \lim_{\lambda \to \infty} \int_{I_n} \sigma_\lambda(x)\, d\nu(x) = \int_{I_n} \gamma(x)\, d\nu(x) \\ &= \nu(H^+_{\mathbf{w},\theta}) + \sigma(\phi)\nu(P_{\mathbf{w},\theta}). \end{aligned}$$

Tomando ahora límite $\phi \to -\infty$, concluimos que

$$\nu(H^+_{\mathbf{w},\theta}) = 0, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R},$$

y por ende

$$\nu(P_{\mathbf{w},\theta}) = 0, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R}.$$

Usando finalmente el Lema 9.3.15 queda demostrado el Teorema □

Observemos que la clase $U$ definida en el Teorema previo es la clase de funciones generada por una red neuronal con una capa oculta, función de activación sigmoide en esa capa y función de activación lineal en la capa de salida. El resultado de Cybenko dice que es posible aproximar con un error arbitrariamente pequeño cualquier función continua $f \in C(I_n)$ con una red neuronal de estas características si no limitamos el número de neuronas en la capa oculta, $N$, el tamaño de los pesos $\mathbf{w}_j$, los sesgos $\theta_j$ y de los pesos de la capa de salida $\alpha_j$.

**9.3.2. Aprendiendo señales de energía finita.** Recordemos que una señal $f$ se dice de energía finita si $f \in L^2(I_n)$, es decir

$$\int_{I_n} |f(x)|^2\, dx < \infty.$$

Este espacio no sólo es un espacio vectorial normado, sino que posee un *producto interno*:

$$(f,g) = \int_{I_n} f(x)g(x)\,dx.$$

La norma, entonces viene *inducida* por este producto interno:

$$\|f\| = (f,f)^{\frac{1}{2}} = \left(\int_{I_n} |f(x)|^2\,dx\right)^{\frac{1}{2}}.$$

La principal propiedad que tienen los espacios con producto interno es el concepto de *ortogonalidad*

DEFINICIÓN 9.3.17. Dos señales $f, g \in L^2(I_n)$ se dicen ortogonales, o que no interactúan entre sí, si se verifica

$$(f,g) = \int_{I_n} f(x)g(x)\,dx = 0,$$

y notamos esto como $f \perp g$.

Observemos que si consideramos una función de activación $\sigma$ acotada, entonces $h_{\mathbf{w},\theta}(x) = \sigma(\mathbf{w}\cdot x + \theta)$ verifica que $h_{\mathbf{w},\theta} \in L^2(I_n)$ para todo $\mathbf{w} \in \mathbb{R}^n$ y $\theta \in \mathbb{R}$.

En esta sección veremos qué se precisa de la función de activación $\sigma$ para garantizar que el conjunto

$$U = \left\{ \sum_{j=1}^{N} \alpha_j \sigma(\mathbf{w}_j \cdot x + \theta_j) \colon \mathbf{w}_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}, N \geq 1 \right\}$$

sea denso en $L^2(I_n)$. Notemos que $U$ es un subespacio lineal de $L^2(I_n)$.

Observemos primero que si $U$ es denso y tenemos que $g \in L^2(I_n)$ verifica que $g \perp U$ (es decir, $(g,u) = 0$ para toda $u \in U$), entonces por la continuidad del producto interno obtenemos que $(g,f) = 0$ para toda $f \in L^2(I_n)$ de donde, tomando $f = g$, sigue que $0 = (g,g) = \|g\|^2$, es decir, $g = 0$. En resumen, si $U$ es denso, la única función $g \in L^2(I_n)$ que es ortogonal a $U$ es la función 0.

Observemos también que la condición $g \perp u$ para toda $u \in U$ es equivalente a

$$\int_{I_n} \sigma(\mathbf{w}\cdot x + \theta) g(x)\,dx, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R}.$$

El recíproco de la observación es cierto. Es decir, si $U$ no es denso en $L^2(I_n)$, entonces existe $g \in L^2(I_n)$, $g \neq 0$, tal que $g \perp U$.

La demostración de esta afirmación es una combinación del lema de separación, Lema 9.3.1 junto con el Teorema de representación de Riesz:

TEOREMA 9.3.18. *Dada $L\colon L^2(I_n) \to \mathbb{R}$ lineal y continua, existe una única $g \in L^2(I_n)$ tal que*

$$L(f) = \int_{I_n} f(x)g(x)\,dx, \quad \forall f \in L^2(I_n).$$

*Más aún, $\|L\| = \|g\|_2$.*

El Teorema de representación de Riesz, Teorema 9.3.18, es más general que el enunciado acá y su demostración se estudia en el curso de Análisis Funcional.

Con estos preliminares, se tiene el siguiente resultado.

PROPOSICIÓN 9.3.19. *Sea $U \subset L^2(I_n)$ un subespacio lineal. Entonces $U$ es denso en $L^2(I_n)$ si y sólo si verifica la propiedad*

$$(g, u) = 0, \quad \forall u \in U \quad \Longrightarrow \quad g = 0.$$

DEMOSTRACIÓN. Como observamos antes, la demostración utiliza el Teorema 9.3.18 y el Lema 9.3.1 y queda de ejercicio al lector. □

Toda esta discusión motiva la siguiente definición.

DEFINICIÓN 9.3.20. Una función de activación $\sigma$ se dice *discriminatoria en el sentido $L^2$* si

(i) $0 \leq \sigma \leq 1$;
(ii) Si $g \in L^2(I_n)$ es tal que

$$\int_{I_n} \sigma(\mathbf{w} \cdot x + \theta) g(x)\, dx, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R},$$

entonces $g = 0$.

Tenemos entonces el siguiente corolario

COROLARIO 9.3.21. *Sea $\sigma$ una función discriminatoria en sentido $L^2$. Entonces el subespacio lineal*

$$U = \left\{ \sum_{j=1}^{N} \alpha_j \sigma(\mathbf{w}_j \cdot x + \theta_j) \colon \mathbf{w}_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}, N \geq 1 \right\}$$

*es denso en $L^2(I_n)$.*

Lo que nos dice el Corolario 9.3.21 es que si $\sigma$ es discriminatoria en sentido $L^2$, entonces dada $h \in L^2(I_n)$ y $\varepsilon > 0$, existe $G \in U$ tal que

$$\int_{I_n} |G(x) - h(x)|^2\, dx < \varepsilon.$$

Equivalentemente, toda función de $L^2$ con entradas en el cubo $I_n$ puede ser aprendida por una red neuronal de una capa oculta con función de activación discriminatoria en sentido $L^2$ y con un error cuadrático medio tan pequeño como queramos.

Lo que queda por determinar es qué funciones de activación $\sigma$ son discriminatorias en sentido $L^2$. Veremos que tanto la función de Heaviside $H(x)$ como la función logística $\sigma(x)$ ambas son discriminatorias en sentido $L^2$.

Empecemos con el siguiente lema

LEMA 9.3.22. *Sea $g \in L^2(I_n)$ tal que $\int_{H_{\mathbf{w},\theta}} g(x)\, dx = 0$ para todo semiespacio $H_{\mathbf{w},\theta} = \{\mathbf{w} \cdot x + \theta > 0\} \cap I_n$. Entonces $g = 0$.*

DEMOSTRACIÓN. La demostración de este lema es completamente análoga a la del Lema 9.3.15. □

EJEMPLO 9.3.23. La función de Heaviside $H(x) = \begin{cases} 1, & \text{si } x \geq 0 \\ 0, & \text{si } x < 0 \end{cases}$ es discriminatoria en sentido $L^2$.

Es obvio que $0 \leq H(x) \leq 1$. Asumamos ahora que

$$\int_{I_n} H(\mathbf{w} \cdot x + \theta) g(x)\, dx = 0, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R},$$

o, equivalentemente,

$$\int_{H_{\mathbf{w},-\theta}} g(x)\, dx = 0, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R},$$

lo que implica que $g = 0$ por el lema previo.

EJEMPLO 9.3.24. La función logística $\sigma(x) = (1 + e^{-x})^{-1}$ es discriminatoria en sentido $L^2$.

Efectivamente, sea $g \in L^2(I_n)$ y supongamos que verifica

$$\int_{I_n} \sigma(\mathbf{w} \cdot x + \theta) g(x)\, dx = 0, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R}.$$

Sea $\sigma_\lambda(x) = \sigma(\lambda(\mathbf{w} \cdot x + \theta))$, luego obtenemos

$$\int_{I_n} \sigma_\lambda(x) g(x)\, dx = 0, \quad \forall \lambda \in \mathbb{R}.$$

Observemos que $\lim_{\lambda \to \infty} \sigma_\lambda(x) = \mathbf{1}_{H_{\mathbf{w},\theta}}(x)$ en casi todo punto.

Si asumimos por un momento que la convergencia $\sigma_\lambda \to \mathbf{1}_{H_{\mathbf{w},\theta}}$ es en sentido $L^2$, obtenemos que

$$0 = \lim_{\lambda \to \infty} (\sigma_\lambda, g) = (\mathbf{1}_{H_{\mathbf{w},\theta}}, g) = \int_{H_{\mathbf{w},\theta}} g(x)\, dx.$$

Como $\mathbf{w} \in \mathbb{R}^n$ y $\theta \in \mathbb{R}$ son arbitrarios, sigue del Lema 9.3.22 que $g = 0$ lo que prueba que $\sigma$ es discriminatoria en sentido $L^2$.

La demostración de que $\sigma_\lambda \to \mathbf{1}_{H_{\mathbf{w},\theta}}$ en sentido $L^2$ queda de ejercicio al lector.

**9.3.3. Aprendiendo funciones integrables.** La teoría de la sección anterior puede adaptarse fácilmente para aproximar funciones integrables $f \in L^1(I_n)$.

La principal diferencia es que los funcionales definidos sobre $L^1(I_n)$ vienen *representados* por funciones acotadas. Esto significa que si $L\colon L^1(I_n) \to \mathbb{R}$ es lineal y continuo, entonces

$$L(f) = \int_{I_n} f(x) g(x)\, dx, \tag{9.3.1}$$

para alguna función $g \in L^\infty(I_n)$. Esto motiva la siguiente definición, que es la análoga a la Definición 9.3.20 al caso $L^1$

DEFINICIÓN 9.3.25. Una función de activación $\sigma$ se dice *discriminatoria en el sentido* $L^1$ si

(i) $\sigma$ es acotada;

(ii) Si $g \in L^\infty(I_n)$ es tal que

$$\int_{I_n} \sigma(\mathbf{w} \cdot x + \theta) g(x)\, dx, \quad \forall \mathbf{w} \in \mathbb{R}^n,\ \theta \in \mathbb{R},$$

entonces $g = 0$.

Luego tenemos el siguiente resultado que es la versión $L^1$ del Corolario 9.3.21.

TEOREMA 9.3.26. *Sea $\sigma$ una función discriminatoria en sentido $L^1$. Entonces el subespacio lineal*

$$U = \left\{ \sum_{j=1}^{N} \alpha_j \sigma(\mathbf{w}_j \cdot x + \theta_j) \colon \mathbf{w}_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}, N \geq 1 \right\}$$

*es denso en $L^1(I_n)$.*

DEMOSTRACIÓN. Sea $L \colon L^1(I_n) \colon \mathbb{R}$ tal que $L(u) = 0$ para toda $u \in U$. Por el Corolario 9.3.4, si probamos que $L = 0$ concluimos el resultado deseado.

Pero esto es una consecuencia inmediata del hecho de que $\sigma$ es discriminatoria en sentido $L^1$ y de la representación (9.3.1). □

Dado que para $g \in L^\infty(I_n)$ la medida $d\mu(x) = g(x)\, dx$ es una medida de Borel signada, la demostración del Teorema 9.3.16 implica que si $\sigma$ es una función sigmoide acotada, entonces resulta discriminatoria en sentido $L^1$.

En particular, una red neuronal de una capa oculta con función de activación logística y una neurona lineal en la capa de salida puede aprende funciones de $L^1(I_n)$.

## 9.4. Aprendiendo soluciones de EDOs

Una aplicación interesante de las redes neuronales es la capacidad de aprender soluciones de ecuaciones diferenciales ordinarias (EDOs).

Para fijar ideas supongamos que queremos aprender las soluciones de la ecuación

$$\begin{cases} y'(t) & = f(t, y(t)) \\ y(t_0) & = y_0, \quad t \in [t_0, t_0 + T), \end{cases} \tag{9.4.1}$$

para algún $T > 0$.

Sabemos que si $f(t, \cdot)$ es Lipschitz entonces la ecuación (9.4.1) tiene una solución única para un cierto $T > 0$. Por simplicidad asumiremos que $y(t) \in \mathbb{R}$ y que $f \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ es Lipschitz en la segunda variable. Resultados similares se pueden obtener para sistemas de ecuaciones diferenciales que corresponden al caso en que $y(t) \in \mathbb{R}^n$.

Construiremos una red neuronal *feedforward* (o de propagación hacia adelante), cuyas entradas sean la variable continua $t \in [t_0, t_0 + T)$ y la salida sea una función suave $\phi_\theta(t)$, que aproxime a la solución $y(t)$ de la ecuación. Los parámetros de la ecuación los notamos por $\theta$.

Consideremos la función de costo

$$\begin{aligned} C(\theta) &= \|\phi'_\theta - f(\cdot, \phi_\theta)\|_2^2 + (\phi_\theta(t_0) - y_0)^2 \\ &= \int_{t_0}^{t_0+T} (\phi'_\theta(t) - f(t, \phi_\theta(t)))^2 \, dt + (\phi_\theta(t_0) - y_0)^2 \end{aligned}$$

y consideramos $\theta^* = \arg\min C(\theta)$. Entonces la salida de la red será $\phi_{\theta^*}$ que comienza con un valor $\phi_{\theta^*}(t_0)$ cercano a $y_0$ y evoluciona cerca de la solución de (9.4.1) que empieza en $y_0$.

En general, la implementación de $C(\theta)$ es compleja y es mejor considerar versiones discretas de la misma. Para eso consideramos discretización del intervalo temporal $t_0 < t_1 < \cdots < t_n = t_0 + T$ con $\Delta t = T/n$ y asociamos la función de costo empírica

$$C(\theta, \{t_i\}) = \sum_{k=0}^{n-1} \left( \frac{\phi_\theta(t_{k+1}) - \phi_\theta(t_k)}{\Delta t} - f(t_k, \phi_\theta(t_k)) \right)^2 \Delta t + (\phi_\theta(t_0) - y_0)^2.$$

Si utilizamos una red neuronal de una capa oculta con función de activación logística y salida lineal, la salida de la red será

$$\phi_\theta(t) = \sum_{j=1}^{N} \alpha_j \sigma(w_j t + b_j), \quad \theta = (\mathbf{w}, \mathbf{b}, \alpha) \in \mathbb{R}^N \times \mathbb{R}^N \times \mathbb{R}^N. \tag{9.4.2}$$

El uso de la función de activación logística $\sigma$ asegura la diferenciabilidad de $\phi_\theta$. Luego, calculamos el gradiente de la función de costo $C(\theta)$ y el método de descenso de gradiente para hallar el valor de $\theta^*$. Ver el Ejercicio 9.6.10.

## 9.5. Resumen

Este capítulo responde a la pregunta sobre qué tipo de funciones pueden ser aproximadas mediante redes neuronales con múltiples capas y se basa en los resultados de la teoría de aproximación desarrollados por Funahashi, Hornik, Stinchcombe, White y Cybenko a finales de la década de 1980.

Los resultados demostrados en este capítulo muestran que una red neuronal con una capa oculta, utilizando neuronas con función de activación sigmoide en la capa oculta y activación lineal en la capa de salida, puede aprender funciones continuas, funciones integrables y funciones de energía finita. La capacidad de las redes neuronales para ser aproximadores universales no depende de la elección específica de la función de activación, sino de la arquitectura de la red feedforward en sí misma.

El precio a pagar es que el número de neuronas en la capa oculta no tiene un límite a priori. Sin embargo, existe un resultado que establece que, por cada dígito adicional de precisión en la aproximación del objetivo, el número de neuronas ocultas debe aumentar en un factor de 100 (ver [**1**]).

Los resultados de aproximación también son válidos si la función de activación es una ReLU o una Softplus. Sin embargo, la demostración no es una modificación directa de las pruebas presentadas en este capítulo. El compromiso entre profundidad, ancho y error de aproximación sigue siendo un tema de investigación activo.

Las redes neuronales también pueden utilizarse para resolver numéricamente ecuaciones diferenciales de primer orden con condiciones iniciales.

## 9.6. Ejercicios

EJERCICIO 9.6.1. Se sabe que el conjunto de los números racionales, $\mathbb{Q}$, es denso en el conjunto de los números reales, $\mathbb{R}$. Formula este resultado en términos de la terminología del aprendizaje automático.

EJERCICIO 9.6.2. Un resultado de aproximación bien conocido es el Teorema de Aproximación de Weierstrass: Sea $f$ una función continua con valores reales definida en el intervalo real $[a, b]$. Entonces, existe una sucesión de polinomios $P_n$ tal que

$$\sup_{x\in[a,b]} |f(x) - P_n(x)| \to 0, \quad \text{cuando } n \to \infty.$$

Formula este resultado en términos de la terminología del aprendizaje automático.

EJERCICIO 9.6.3 (separación de puntos). Sean $x_0, x_1$ dos vectores distintos y no colineales en el espacio normado lineal $X$. Demuestra que existe un funcional lineal acotado $L$ en $X$ tal que $L(x_0) = 1$ y $L(x_1) = 0$.

EJERCICIO 9.6.4. Sean $x_0, x_1$ dos vectores distintos y no colineales en el espacio normado lineal $X$. Demuestra que existe un funcional lineal acotado $L$ en $X$ tal que $L(x_0) = L(x_1) = \frac{1}{2}$, con

$$\|L\| \leq \frac{\delta_0 + \delta_1}{2\delta_0\delta_1},$$

donde $\delta_i$ es la distancia de $x_i$ a la recta generada por el otro vector.

EJERCICIO 9.6.5. Dados dos números finitos $a$ y $b$, demuestra que existe una medida de Borel finita con signo no nula en $[a, b]$ tal que

$$\int_a^b \sin t \, d\mu(t) = \int_a^b \cos t \, d\mu(t).$$

EJERCICIO 9.6.6. Sea $P([0, 1])$ el espacio de polinomios en $[0, 1]$. Para cualquier $P \in P([0, 1])$, define el funcional

$$L(P) = a_0 + a_1 + \cdots + a_n,$$

donde $P(x) = a_0 + a_1x + \cdots + a_nx^n$, con $a_i \in \mathbb{R}$.

(a) Demuestra que $L$ es un funcional lineal y acotado en $P([0, 1])$.
(b) Prueba que existe una medida de Borel finita con signo, $\mu$ en $[0, 1]$, tal que

$$\int_0^1 P(x) \, d\mu(x) = a_0 + a_1 + \cdots + a_n, \quad \forall P \in P([0, 1]).$$

EJERCICIO 9.6.7. (a) ¿La función tangente hiperbólica es discriminatoria en el sentido $L^2$? ¿Y en el sentido $L^1$?
(b) Demuestra que la función $\varphi(x) = e^{-x^2}$ es discriminatoria en el sentido $L^1$ en $\mathbb{R}$.

EJERCICIO 9.6.8. (a) Escribe una fórmula para la salida de una red neuronal de alimentación hacia adelante (FNN) con dos capas ocultas que tienen $N_1$ y $N_2$ neuronas, respectivamente.
(b) Demuestra que las salidas de todas las posibles FNNs con dos capas ocultas y la misma entrada forman un espacio lineal de funciones.

EJERCICIO 9.6.9. Una función de activación $\sigma$ se llama fuertemente discriminatoria para la medida $\mu$ si

$$\int_{I_n} (\sigma \circ f)(x)\, d\mu(x) = 0, \quad \forall f \in C(I_n) \Rightarrow \mu = 0.$$

(a) Demuestra que si $\sigma$ es fuertemente discriminatoria, entonces es discriminatoria en el sentido de las funciones continuas.
(b) Suponer que las funciones de activación de una FNN de dos capas son continuas y fuertemente discriminatorias con respecto a cualquier medida con signo. Demuestra que esta FNN puede aprender cualquier función continua en $C(I_n)$.

EJERCICIO 9.6.10. Encuentra $\varphi'_\theta(t)$, $\nabla_w \varphi_\theta(t)$ y $\nabla_w \varphi'_\theta(t)$ para la expresión de $\varphi_\theta(t)$ dada por (9.4.2).

Capítulo 10

# Clasificación

En problemas de clasificación, una red neuronal tiene que ser capaz de clasificar clusters, es decir, asignar etiquetas distintas a cada cluster. Estas etiquetas pueden ser números naturales, o puntos en el espacio, o vectores que pertenecen al espacio de etiquetas. El procedimiento de clasificación es equivalente al de ser capaz de aprender una *función de división de clusters* o un mapa de decisión. El conjunto de entrenamiento provee etiquetas para cada cluster y esta asignación define un mapa de decisión. La red debe ser capaz de clasificar los datos aprendiendo este mapa de decisión.

## 10.1. Relaciones de equivalencia

Consideremos un conjunto no vacío $\Omega \neq \emptyset$. Un conjunto $S \subset \Omega \times \Omega$ se lo llama una *relación* en $\Omega$. Recordemos algunas definiciones.

(i) $S$ se llama *reflexiva* si contiene a la diagonal $\{(x, x) \colon x \in \Omega\}$.
(ii) $S$ se llama *simétrica* si $(x, y) \in S$ implica que $(y, x) \in S$.
(iii) $S$ se llama *transitiva* si $(x, y), (y, z) \in S$ implica que $(x, z) \in S$.

Una relación $S$ que es reflexiva, simétrica y transitiva es llamada una *relación de equivalencia.*

Dada una relación de equivalencia $S$, decimos que dos puntos $x, y \in \Omega$ son *equivalentes* bajo $S$ si $(x, y) \in S$ y notamos esta equivalencia como $x \sim y$. Dado $x \in I_n$, notamos por $C_x = \{y \in \Omega \colon x \sim y\}$ y se denomina la *clase de equivalencia* de $x$. El conjunto formado por todas las clases de equivalencia, se lo denota por $\Omega|_\sim$ y es llamado el conjunto cociente.

Una *partición* del conjunto $I_n$ es una colección de subconjuntos $\{A_i\}_{i \in I}$ con las siguientes propiedades

(i) $A_i \neq \emptyset$ para todo $i \in I$;
(ii) $A_i \cap A_j = \emptyset$, para todo $i, j \in I$, $i \neq j$;
(iii) $\bigcup_{i \in I} A_i = \Omega$.

Si el conjunto de índices $I$ es finito, decimos que $\{A_i\}_{i \in I}$ es una *partición finita.* Estas particiones finitas pueden utilizarse para clasificar los puntos de $\Omega$ en $N$ clases diferentes.

El siguiente resultado, de demostración elemental y que suele verse en el curso de Álgebra 1, muestra la relación existente entre particiones y relaciones de equivalencia.

Proposición 10.1.1. *Sea $\sim$ una relación de equivalencia en $\Omega$. Entonces existe una partición $\{A_i\}_{i \in I}$ de $\Omega$ tal que*

(i) *para todo $i \in I$ y para todo $xy \in A_i$ tenemos que $x \sim y$;*
(ii) *para todo $x, y \in \Omega$ con $x \sim y$, existe $i \in I$ tal que $x, y \in A_i$.*

*Recíprocamente, dada una partición* $\{A_i\}_{i\in I}$ *de* $I_n$*, la relación* $x \sim y$ *si y sólo si* $x, y \in A_i$ *para algún* $i \in I$ *define una relación de equivalencia en* $\Omega$*.*

DEMOSTRACIÓN. La demostración queda de ejercicio. □

## 10.2. Entropía de una partición

Consideremos ahora dentro de $\Omega$ una $\sigma-$álgebra $\mathcal{F}$ y una medida de probabilidad $\mu$ de manera tal que $(\Omega, \mathcal{F}, \mu)$ es un espacio de probabilidad. En este espacio se considera una partición finita $\mathcal{A} = \{A_j\}_{j=1,\dots,N}$ de $\Omega$ en donde los conjuntos son medibles (i.e. $A_j \in \mathcal{F}$). La *entropía de la partición* $\mathcal{A}$ con respecto a la medida de probabilidad $\mu$ se define como

$$H(\mathcal{A}, \mu) = -\sum_{j=1}^{N} \mu(A_j) \ln \mu(A_j).$$

Como $\mu(A_j) \in (0, 1)$, la entropía es positiva, $H(\mathcal{A}, \mu) > 0$. Es fácil probar que la entropía es máxima cuando todos los conjuntos de la partición tienen la misma medida $\mu(A_1) = \cdots = \mu(A_N) = \frac{1}{N}$.

EJEMPLO 10.2.1. Sea $\Omega \subset \mathbb{R}^n$ un conjunto acotado y definimos la medida de probabilidad $\mu$ en $\Omega$ sobre la $\sigma-$álgebra de Borel, como

$$\mu(A) = \frac{|A|}{|\Omega|}$$

($|\cdot|$ representa el volumen del conjunto o medida de Lebesgue).

En este caso, la entropía representa la incertidumbre de dividir el conjunto $\Omega$ en subconjuntos de diferentes volúmenes.

EJEMPLO 10.2.2. Supongamos que tenemos dos medidas de probabilidad definidas en el espacio medible $(\Omega, \mathcal{F})$, $\mu$ y $\nu$ y asumamos que existe una función $p\colon \Omega \to \mathbb{R}$ tal que

$$\mu(A) = \int_A p(x)\, d\nu(x),$$

para todo conjunto $A \in \mathcal{F}$. Es decir, $p$ es la densidad de probabilidad de $\mu$ con respecto a $\nu$ (el Teorema de Radon-Nikodym da condiciones necesarias y suficientes en las medidas $\mu$ y $\nu$ para garantizar la existencia de esta densidad $p$, este teorema se estudia en el curso de Análisis Real).

Queremos analizar como varía la entropía de una partición al calcularla con respecto a la medida $\mu$ y a la medida $\nu$. Entonces,

$$H(\mathcal{A}, \mu) = -\sum_{j=1}^{N} \int_{A_j} p(x)\, d\nu(x) \ln \left( \int_{A_j} p(x)\, d\nu(x) \right).$$

## 10.3. Funciones de decisión

Consideremos ahora el conjunto $\Omega = I_n = [0, 1] \times \cdots [0, 1] \subset \mathbb{R}^n$ el cubo unitario $n-$dimensional y sea $\{A_1, \dots, A_N\}$ una partición finita de $I_n$ formada por conjuntos de la $\sigma-$álgebra de Borel. Una *función de decisión* es una función que asocia a cada conjunto de la partición un número natural distinto, es decir $f\colon I_n \to \mathbb{N}$, $f(x) = j$

si $x \in A_j$. De manera equivalente, $f(x) = \sum_{j=1}^{N} j\mathbf{1}_{A_j}(x)$. Observemos también que $A_j = f^{-1}(j)$.

El conjunto $\{1, 2, \ldots, N\}$ se llama el *conjunto de etiquetas* y $\mathbb{R}$ se lo llama el *espacio de etiquetas*.

EJEMPLO 10.3.1 (Caso $N = 2$). Consideremos dos clusters de puntos en $\mathbb{R}^n$ que están separados por un hiperplano $\{\mathbf{w} \cdot x + \theta = 0\}$. Luego, un perceptrón puede decidir sobre cada punto a qué cluster pertenece, usando la función de decisión $f(x) = 1 + H(\mathbf{w} \cdot x + \theta)$, donde $H(x)$ es la función de Heaviside. El conjunto de etiquetas es $\{1, 2\}$ y el espacio de etiquetas es $\mathbb{R}$.

Veamos ahora un resultado teórico general que nos dice que si estamos dispuestos a pagar el precio de dejar un conjunto arbitrariamente pequeño sin clasificar entonces una red neuronal con una capa oculta puede implementar cualquier función de decisión.

PROPOSICIÓN 10.3.2. *Sea $f$ una función de decisión asociada con una partición finita y medible $\{A_1, \ldots, A_N\}$ de $I_n$. Sea $\sigma$ una función sigmoide que verifica las hipótesis de la Proposición 8.1.1.*

*Entonces, para todo $\varepsilon > 0$, existe una función $G(x)$ de la forma*

$$G(x) = \sum_{j=1}^{k} \alpha_j \sigma(\mathbf{w}_j \cdot x + \theta_j), \quad \mathbf{w}_j \in \mathbb{R}^n \ \alpha_j, \theta_j \in \mathbb{R}$$

*y un conjunto $D \subset I_n$ tal que $|I_n \setminus D| > 1 - \varepsilon$ y*

$$|G(x) - f(x)| < \varepsilon, \quad \forall x \in D.$$

DEMOSTRACIÓN. Esta proposición es inmediata del Teorema 9.3.16 y del Teorema de Lusin. En efecto, el Teorema de Lusin, dice que dado $\varepsilon > 0$ existe un conjunto $D \subset I_n$ medible con $|I_n \setminus D| > 1 - \varepsilon$ y una función continua $g\colon I_n \to \mathbb{R}$ tal que $f(x) = g(x)$ para todo $x \in D$. La demostración de este teorema se ve en el curso de Análisis Real.

Ahora, el Teorema 9.3.16 nos dice que existe una función $G(x)$ con la forma dada por el enunciado de la proposición, tal que $|G(x) - g(x)| < \varepsilon$ para todo $x \in I_n$. Luego,

$$|G(x) - f(x)| = |G(x) - g(x)| < \varepsilon, \quad \forall x \in D.$$

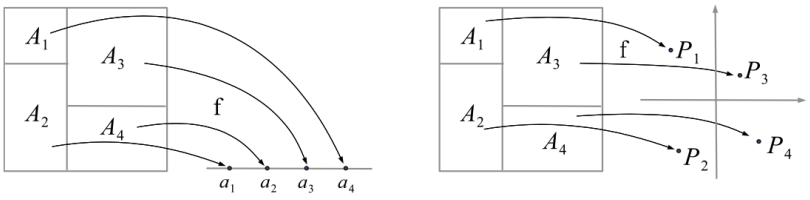
Queda entonces demostrada la proposición. □

Notemos que este es un resultado de existencia. No da ninguna información sobre cómo construir la función $G(x)$. Esto es un problema completamente diferente.

OBSERVACIÓN 10.3.3. A cada función de decisión $f$ se le puede asociar una entropía de la siguiente manera. Dada una partición finita $\mathcal{A} = \{A_1, \ldots, A_N\}$ y una función de decisión $f$, se define una medida de probabilidad $\mu$ como

$$\mu(A_i) = \frac{f(A_i)}{\sum_{i=1}^{N} f(A_i)}$$

y consideramos entonces la entropía $H(\mathcal{A}, \mu)$ como la entropía asociada a la partición $\mathcal{A}$ y la función de decisión $f$.




**a** **b**

FIGURA 10.1. Mapa de decisión asociado a una partición: **a**. El espacio de etiquetas es $\mathbb{R}$. **b**. El espacio de etiquetas es $\mathbb{R}^2$.

## 10.4. Mapas de decisión de vectores *one-hot*

Hay situaciones en donde resulta más conveniente reemplazar las etiquetas de números enteros por vectores unitarios, o vectores *one hot*. Es decir, en lugar de usar como etiquetas $1, 2, \ldots$, podemos asociar los clusters con vectores $\mathbf{e}_1 = (1, 0, \ldots, 0)$, $\mathbf{e}_2 = (0, 1, 0, \ldots, 0)$, etc. El conjunto de etiquetas queda formado por $\{\mathbf{e}_1, \ldots, \mathbf{e}_N\}$ y el espacio de etiquetas resulta $\mathbb{R}^N$.

Sea $\{A_1, \ldots, A_N\}$ una partición medible finita de $I_n$. Un *mapa de decisión de vectores one-hot* es una función medible $f\colon I_n \to \mathbb{R}^N$, que asocia un vector one-hot con cada clase de la partición. Es decir, $f(x) = \mathbf{e}_j$ para todo $x \in A_j$. En este caso, la etiqueta asociada con la clase $A_j$ es un vector unitario $N-$dimensional y estas etiquetas forman una base en $\mathbb{R}^N$.

La ventaja de usar vectores one-hot como etiquetas en lugar de números enteros, es que al tener como espacio de etiquetas un espacio de dimensión mayor, $\mathbb{R}^N$, esto proporciona más espacio para que los conjuntos de prueba se agrupen en direcciones linealmente independientes, lo que lleva a una mejor separación de las clases.

Elegir a los vectores one-hot $\mathbf{e}_j$ como etiquetas es sólo por conveniencia. Se podrían elegir como etiquetas otros $N$ vectores linealmente independientes en $\mathbb{R}^N$, eventualmente de manera que formen una base ortonormal.

El espacio de etiquetas puede tener dimensión menor que $N$, el número de clases en $I_n$. Por ejemplo se puede considerar a $N = 4$ clases y asociar a cada clase como etiqueta un punto en $\mathbb{R}$ o en $\mathbb{R}^2$ respectivamente. Ver Figura 10.4.

En situaciones, en vez de considerar $N$ vectores linealmente independientes en $\mathbb{R}^N$ se consideran $N$ puntos en $\mathbb{R}^N$ notados por $\mathbf{p}_1, \ldots, \mathbf{p}_N$. Al usar puntos como etiquetas necesitamos que los mismos sean linealmente independientes, pero no necesariamente vistos como vectores con origen en el origen de coordenadas. Para entender este concepto damos una definición y un resultado asociado.

DEFINICIÓN 10.4.1. Decimos que $E \subset \mathbb{R}^N$ es un *espacio afín*, si existe $\mathbf{p}_0 \in \mathbb{R}^N$ y $\mathbb{S} \subset \mathbb{R}^N$ subespacio, tal que $E = \mathbb{S} + \mathbf{p}_0$.

PROPOSICIÓN 10.4.2. *Sea $E$ un espacio afín. Asumamos que $E = \mathbb{S} + \mathbf{p} = \mathbb{T} + \mathbf{q}$ donde $\mathbb{S}$ y $\mathbb{T}$ son dos subespacios de $\mathbb{R}^N$ y $\mathbf{p}$, $\mathbf{q}$ son puntos de $\mathbb{R}^N$. Entonces $\mathbb{S} = \mathbb{T}$ y $\mathbf{p}, \mathbf{q} \in E$.*

Es decir, el subespacio que caracteriza al espacio afín es único.

DEMOSTRACIÓN. Si $E = \mathbb{S}+\mathbf{p}$ con $\mathbb{S} \subset \mathbb{R}^N$ un subespacio, entonces $\mathbf{p} = 0+\mathbf{p} \in E$ dado que $0 \in \mathbb{S}$.

Ahora, si tenemos que $\mathbb{S} + \mathbf{p} = \mathbb{T} + \mathbf{q}$, se tiene que $0 + \mathbf{p} = t + \mathbf{q}$ para algún $t \in \mathbb{T}$. De donde sigue que $\mathbf{p} - \mathbf{q} = t \in \mathbb{T}$. Análogamente, $\mathbf{p} - \mathbf{q} \in \mathbb{S}$.

Finalmente, si $s \in \mathbb{S}$, entonces existe $t \in \mathbb{T}$ tal que $s + \mathbf{p} = t + \mathbf{q}$, de donde $s = t+(\mathbf{p}-\mathbf{q}) \in \mathbb{T}$. Es decir $\mathbb{S} \subset \mathbb{T}$. Por simetría, $\mathbb{T} \subset \mathbb{S}$ lo que finaliza la demostración. □

DEFINICIÓN 10.4.3. Sea $E$ un espacio afín, $E = \mathbb{S} + \mathbf{p}$ con $\mathbb{S} \subset \mathbb{R}^N$ un subespacio. Se define la dimensión del espacio afín $E$ como la dimensión del subespacio $\mathbb{S}$. Es decir

$$\dim E = \dim \mathbb{S}.$$

OBSERVACIÓN 10.4.4. Observemos que dados $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \subset \mathbb{R}^N$ si definimos el subespacio $\mathbb{S}$ como

$$\mathbb{S} = \operatorname{gen}\{\mathbf{p}_2 - \mathbf{p}_1, \ldots, \mathbf{p}_N - \mathbf{p}_1\},$$

entonces $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \subset E = \mathbb{S} + \mathbf{p}_1$, y tenemos que

$$\dim E = \dim \mathbb{S} \leq N - 1.$$

DEFINICIÓN 10.4.5. Sean $\mathbf{p}_1, \ldots, \mathbf{p}_N \in \mathbb{R}^N$. Decimos que $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ está en *posición general* si no existe un subespacio afín de dimensión menor que $N-1$ que los contenga.

Es fácil ver que si $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ está en posición general, el subespacio afín que los contiene (que tiene dimensión $N-1$) es único (ver el Ejercicio **??**).

La siguiente proposición caracteriza a puntos en posición general.

PROPOSICIÓN 10.4.6. *Sean $\mathbf{p}_1, \ldots, \mathbf{p}_N \in \mathbb{R}^N$. Entonces $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ está en posición general si y sólo si existe $\mathbf{p}_0 \in \mathbb{R}^N$ tal que el conjunto $\{\mathbf{p}_1 - \mathbf{p}_0, \ldots, \mathbf{p}_N - \mathbf{p}_0\}$ es linealmente independiente.*

DEMOSTRACIÓN. Asumamos que $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ está en posición general y sea $E$ el subespacio afín de dimensión $N-1$ que los contiene, $E = \mathbb{S}+\mathbf{q}$ para algún $\mathbf{q} \in \mathbb{R}^N$ y $\mathbb{S}$ un hiperplano. Sea $\mathbf{p}_0 \in \mathbb{R}^N \setminus E$. Veamos entonces que el conjunto $\{\mathbf{p}_1-\mathbf{p}_0, \ldots, \mathbf{p}_N-\mathbf{p}_0\}$ es linealmente independiente.

Sean $c_1, \ldots, c_N \in \mathbb{R}$ tales que

$$\sum_{j=1}^{N} c_j(\mathbf{p}_j - \mathbf{p}_0) = 0.$$

Debemos ver que $c_1 = \cdots = c_N = 0$.

Ahora,

$$0 = \sum_{j=1}^{N} c_j(\mathbf{p}_j - \mathbf{p}_0) = \left(\sum_{j=1}^{N} c_j\right)(\mathbf{p}_1 - \mathbf{p}_0) + \sum_{j=2}^{N} c_j(\mathbf{p}_j - \mathbf{p}_1).$$

Por la Observación 10.4.4 sabemos que el conjunto $\{\mathbf{p}_2 - \mathbf{p}_1, \ldots, \mathbf{p}_N - \mathbf{p}_1\}$ es linealmente independiente y es un sistema de generadores de $\mathbb{S}$.

Como $\mathbf{p}_0 \notin E$, sigue que $\mathbf{p}_1 - \mathbf{p}_0 \notin \mathbb{S}$. Luego $\mathbf{p}_1 - \mathbf{p}_0$ es linealmente independiente del conjunto $\{\mathbf{p}_2 - \mathbf{p}_1, \dots, \mathbf{p}_N - \mathbf{p}_1\}$ de donde obtenemos que

$$\left(\sum_{j=1}^{N} c_j\right) = c_2 = \cdots = c_N = 0$$

de donde se deduce lo deseado.

Supongamos ahora que existe $\mathbf{p}_0 \in \mathbb{R}^N$ tal que $\{\mathbf{p}_1 - \mathbf{p}_0, \dots, \mathbf{p}_N - \mathbf{p}_0\}$ es linealmente independiente. Debemos ver que el conjunto $\{\mathbf{p}_1, \dots, \mathbf{p}_N\}$ está en posición general. Sea $\mathbb{S}$ el subespacio generado por $\{\mathbf{p}_2 - \mathbf{p}_1, \dots, \mathbf{p}_N - \mathbf{p}_1\}$, entonces $\{\mathbf{p}_1, \dots, \mathbf{p}_N\} \subset E = \mathbb{S} + \mathbf{p}_1$. Veamos que $\dim \mathbb{S} = N - 1$ o, equivalentemente, que $\{\mathbf{p}_2 - \mathbf{p}_1, \dots, \mathbf{p}_N - \mathbf{p}_1\}$ es linealmente independiente. Sean $c_2, \dots, c_N \in \mathbb{R}$ tales que

$$\sum_{j=2}^{N} c_j(\mathbf{p}_j - \mathbf{p}_1) = 0.$$

Pero de acá obtenemos que

$$0 = \sum_{j=2}^{N} c_j(\mathbf{p}_j - \mathbf{p}_1) = \sum_{j=2}^{N} c_j(\mathbf{p}_j - \mathbf{p}_0) - \left(\sum_{j=2}^{N} c_j\right)(\mathbf{p}_1 - \mathbf{p}_0).$$

Como el conjunto $\{\mathbf{p}_1 - \mathbf{p}_0, \dots, \mathbf{p}_N - \mathbf{p}_0\}$ es linealmente independiente, deducimos que

$$c_2 = \cdots = c_N = \left(\sum_{j=2}^{N} c_j\right) = 0,$$

de donde $c_2 = \cdots = c_N = 0$ como queríamos demostrar. □

Así como los espacios afines definen traslaciones de subespacios, las transformaciones afines son traslaciones de transformaciones lineales.

Definición 10.4.7. Decimos que $f\colon \mathbb{R}^N \to \mathbb{R}^N$ es una transformación afín si existe $g\colon \mathbb{R}^N \to \mathbb{R}^N$ transformación lineal y $\mathbf{p} \in \mathbb{R}^N$ tal que

$$f(\mathbf{x}) = g(\mathbf{x}) + \mathbf{p}, \qquad \forall \mathbf{x} \in \mathbb{R}^N.$$

Con la ayuda de la Proposición 10.4.6 podemos definir una transformación afín biyectiva entre vectores one-hot y puntos en posición general.

Proposición 10.4.8. *Sea $\{\mathbf{p}_1, \dots, \mathbf{p}_N\} \subset \mathbb{R}^N$ en posición general. Entonces existe una transformación afín $f\colon \mathbb{R}^N \to \mathbb{R}^N$ biyectiva tal que $f(\mathbf{p}_j) = \mathbf{e}_j$ para $j = 1, \dots, N$.*

Demostración. Sea $\mathbf{p}_0 \in \mathbb{R}^N$ tal que el conjunto $\{\mathbf{p}_1 - \mathbf{p}_0, \dots, \mathbf{p}_N - \mathbf{p}_0\}$ es linealmente independiente. Luego, existe una única transformación lineal $g\colon \mathbb{R}^N \to \mathbb{R}^N$ tal que $g(\mathbf{p}_j - \mathbf{p}_0) = \mathbf{e}_j$. Esta transformación lineal resulta inversible y la inversa viene dada por la única transformación lineal que verifica $g^{-1}(\mathbf{e}_j) = \mathbf{p}_j - \mathbf{p}_0$ para $j = 1, \dots, N$.

Ahora, definimos $f(\mathbf{x}) = g(\mathbf{x} - \mathbf{p}_0) = g(\mathbf{x}) - g(\mathbf{p}_0) = g(\mathbf{x}) + \mathbf{p}$ (donde $\mathbf{p} = -g(\mathbf{p}_0)$). Luego $f$ es una transformación afín, $f(\mathbf{p}_j) = g(\mathbf{p}_j - \mathbf{p}_0) = \mathbf{e}_j$ y $f$ es inversible con

$$f^{-1}(\mathbf{x}) = g^{-1}(\mathbf{x}) + \mathbf{p}_0,$$

lo que finaliza la demostración. □

## 10.5. Separabilidad lineal

Un cluster de puntos en $\mathbb{R}^n$ es un conjunto $\mathcal{G} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ que suponemos posee cierta característica que lo distingue. Dos clusters en $\mathbb{R}^n$, $\mathcal{G}_1$ y $\mathcal{G}_2$, se dicen *linealmente separables* si existe un hiperplano $\mathcal{H}$ que los separa. Esto significa

(i) El hiperplano $\mathcal{H}$ (espacio afín de dimensión $n-1$) divide el espacio $\mathbb{R}^n$ en dos semi-espacios $S_1$ y $S_2$.
(ii) Cada cluster está contenido en uno de estos semi-espacios: $\mathcal{G}_1 \subset S_1$ y $\mathcal{G}_2 \subset S_2$.

Asumamos que el hiperplano está definido implícitamente por la ecuación

$$h(\mathbf{x}) = a_1 x_1 + \cdots + a_n x_n + d = 0,$$

entonces la separabilidad de $\mathcal{G}_1$ y $\mathcal{G}_2$ se puede escribir como $h(g_1)h(g_2) < 0$ para todo par de puntos $g_i \in \mathcal{G}_i$, $i = 1, 2$.

Al estudiar el problema de la separación lineal de clusters es conveniente entender el conjunto que encierra un cluster dado. Para eso necesitaremos algunas definiciones.

Definición 10.5.1. Un conjunto $K \subset \mathbb{R}^n$ se dice convexo si para todo par de puntos $\mathbf{x}, \mathbf{y} \in K$, el segmento de puntos que une $\mathbf{x}$ con $\mathbf{y}$ y que denotamos por

$$[\mathbf{x}, \mathbf{y}] = \{t\mathbf{x} + (1-t)\mathbf{y} \colon t \in [0,1]\},$$

está contenido en $K$.

En general un cluster no es un conjunto convexo. Entonces necesitamos considerar al menor conjunto convexo que lo contiene.

Definición 10.5.2. Sea $A \subset \mathbb{R}^n$ un conjunto no vacío. Definimos la cápsula convexa de $A$, y la notamos como $\text{hull}(A)$ al menor conjunto convexo que contiene a $A$, es decir

$$\text{hull}(A) = \bigcap_{\substack{A \subset K \\ K \text{ convexo}}} K$$

Observamos que, como $\mathbb{R}^n$ es convexo, siempre existe un convexo conteniendo a cualquier conjunto y que la intersección de un número arbitrario de convexos es convexo (verificar esto). Luego $\text{hull}(A)$ está efectivamente bien definido, es convexo y es el convexo más chico (en el sentido de la inclusión) que contiene a $A$.

Cuando el conjunto $A$ es un cluster formado por finitos elementos, $A = \mathcal{G} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, se tiene una caracterización de su cápsula convexa $\text{hull}(\mathcal{G})$.

Proposición 10.5.3. *Sea $\mathcal{G} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \mathbb{R}^n$. Entonces*

$$\text{hull}(\mathcal{G}) = \left\{ \sum_{i=1}^{k} \lambda_i \mathbf{x}_i \colon \sum_{i=1}^{k} \lambda_i = 1,\ \lambda_i \geq 0 \right\}.$$

Es decir, la cápsula convexa de $\mathcal{G}$ es el conjunto de todas las *combinaciones convexas* de elementos del $\mathcal{G}$. La demostración de esta proposición está contenida en el Ejercicio 10.10.5.

Tenemos el siguiente resultado.

Proposición 10.5.4. *Sean $\mathcal{G}_1, \mathcal{G}_2 \subset \mathbb{R}^n$ dos clusters. Entonces $\mathcal{G}_1$ y $\mathcal{G}_2$ son linealmente separables si y sólo si sus cápsulas convexas* $\text{hull}(\mathcal{G}_1)$ *y* $\text{hull}(\mathcal{G}_2)$ *lo son.*

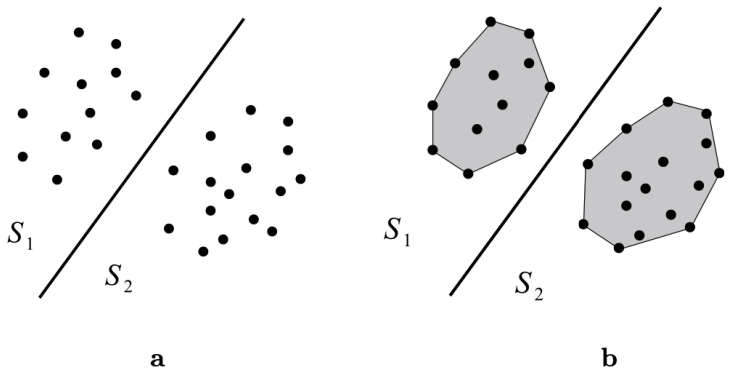



FIGURA 10.2. **a**. Separación lineal de dos clusters. **b**. Separación lineal de las cápsulas convexas de dos clusters.

DEMOSTRACIÓN. Es claro que si $\text{hull}(\mathcal{G}_1)$ y $\text{hull}(\mathcal{G}_2)$ son linealmente separables, entonces $\mathcal{G}_1$ y $\mathcal{G}_2$ también lo son, dado que $\mathcal{G}_i \subset \text{hull}(\mathcal{G}_i)$, $i = 1, 2$. Ver la Figura 10.5

Supongamos ahora que $\mathcal{G}_1$ y $\mathcal{G}_2$ son linealmente separables. Entonces tenemos dos semi-espacios complementarios $S_1$ y $S_2$ tales que $\mathcal{G}_i \subset S_i$, $i = 1, 2$. Pero cada semi-espacio es un conjunto convexo que contiene al cluster, y como la cápsula convexa es el menor conjunto convexo que contiene al cluster deducimos que $\text{hull}(\mathcal{G}_i) \subset S_i$, $i = 1, 2$. □

En el Ejercicio 10.10.9 se ve que $\mathcal{G}_1$ y $\mathcal{G}_2$ son linealmente separables si y sólo si $\text{hull}(\mathcal{G}_1) \cap \text{hull}(\mathcal{G}_2) = \emptyset$.

Si dos clusters no son linealmente separables, entonces no es posible *separarlos* usando transformaciones lineales.

PROPOSICIÓN 10.5.5. *Sean* $\mathcal{G}_1, \mathcal{G}_2 \subset \mathbb{R}^n$ *dos clusters tales que* $\text{hull}(\mathcal{G}_1) \cap \text{hull}(\mathcal{G}_2) \neq \emptyset$. *Entonces si* $F\colon \mathbb{R}^n \to \mathbb{R}^p$ *es una transformación lineal, tenemos que* $F(\mathcal{G}_1)$ *y* $F(\mathcal{G}_2)$ *no son linealmente separables.*

DEMOSTRACIÓN. Asumamos, por el absurdo, que existe $F\colon \mathbb{R}^n \to \mathbb{R}^p$ transformación lineal tal que $F(\mathcal{G}_1)$ y $F(\mathcal{G}_2)$ son linealmente separables.

Sea $h(\mathbf{x}) = \sum_{i=1}^{p} a_i x_i + d = 0$ la ecuación implícita del hiperplano $\mathcal{H}$ que separa $F(\mathcal{G}_1)$ y $F(\mathcal{G}_2)$. Luego, si $g_i \in \mathcal{G}_i$, $i = 1, 2$, tenemos que

$$h(F(g_1))h(F(g_2)) < 0.$$

Sea ahora $g \in \text{hull}(\mathcal{G}_1) \cap \text{hull}(\mathcal{G}_2)$, luego, $g$ admite las representaciones

$$g = \sum_{i=1}^{k_1} \lambda_i^1 g_i^1 = \sum_{j=1}^{k_2} \lambda_j^2 g_j^2,$$

donde $g_i^1 \in \mathcal{G}_1$, $g_j^2 \in \mathcal{G}_2$, $\lambda_i^1, \lambda_j^2 \geq 0$ y $\sum_{i=1}^{k_1} \lambda_i^1 = \sum_{j=1}^{k_2} \lambda_j^2 = 1$.

Si usamos ahora la linealidad de $F$ y observando que $h$ distribuye las combinaciones convexas, obtenemos

$$\begin{aligned} 0 \leq [h(F(g))]^2 &= h(F(g))h(F(g)) \\ &= h\left(F\left(\sum_{i=1}^{k_1} \lambda_i^1 g_i^1\right)\right) h\left(F\left(\sum_{j=1}^{k_2} \lambda_j^2 g_j^2\right)\right) \\ &= \left(\sum_{i=1}^{k_1} \lambda_i^1 h(F(g_i^1))\right)\left(\sum_{j=1}^{k_2} \lambda_j^2 h(F(g_j^2))\right) \\ &= \sum_{i=1}^{k_1}\sum_{j=1}^{k_2} \lambda_i^1 \lambda_j^2 h(F(g_i^1))h(F(g_j^2)) < 0. \end{aligned}$$

Esto es una contradicción y la demostración está completa. □

Observación 10.5.6. (i) Vale la pena notar que tampoco existe una función de separación que sea afín, es decir, de la forma $F(x) = Wx + b$, donde $W$ es una matriz $n \times n$ y $b \in \mathbb{R}^n$. Esto se debe al hecho de que la separabilidad es invariante por traslaciones. En consecuencia, una neurona lineal no puede separar dos clusters cuyas cápsulas convexas se intersecan. Para esta tarea, debemos emplear redes neuronales con funciones de activación no lineales.

(ii) Dos clusters en $\mathbb{R}^n$, $\mathcal{G}_1$ y $\mathcal{G}_2$, se llaman *$F-$separables* si existe una aplicación bi-continua e invertible $F\colon \mathbb{R}^n \to \mathbb{R}^n$ tal que las imágenes de los clusters, $F(\mathcal{G}_1)$ y $F(\mathcal{G}_2)$, son separables linealmente en $\mathbb{R}^n$. Dicha función $F$ se llama un *homeomorfismo de* $\mathbb{R}^n$.

Resultados estándar de la teoría de redes neuronales muestran que una *red neuronal feedforward* (con suficientes capas ocultas) puede aprender la función no lineal y continua $F$. Si además agregamos un perceptrón, podemos realizar la clasificación lineal final. De esta manera, un problema de clasificación se reduce al aprendizaje de la función continua no lineal $F$.

(iii) El papel de la función $F$ es *separar los clusters*, de modo que puedan clasificarse linealmente. Sin embargo, hay casos en los que los clusters no pueden separarse mediante un homeomorfismo de $\mathbb{R}^n$. En este caso, necesitamos una dimensión extra para separar los clusters, y la función continua debe ser $F\colon \mathbb{R}^n \to \mathbb{R}^p$, con $p > n$.

Por ejemplo, los clusters

$$\mathcal{G}_1 = \{x \in \mathbb{R}^2 \colon \|x\| < 1\}, \quad \mathcal{G}_2 = \{x \in \mathbb{R}^2 \colon \|x\| > 2\}$$

no pueden separarse en $\mathbb{R}^2$ de forma continua, pero sí pueden separarse en $\mathbb{R}^3$ si elevamos una de ellas en la dirección vertical.

Observación 10.5.7. Cuando se tienen $k$ clusters, $\mathcal{G}_1, \ldots, \mathcal{G}_k$, decimos que son separables linealmente si lo son dos a dos. Es decir, si $\mathcal{G}_i$ y $\mathcal{G}_j$ son linealmente separables para todo $i \neq j$, $i, j = 1, \ldots, k$.

## 10.6. Separabilidad convexa

La separabilidad puede ser considerada en una manera levemente diferente, pero equivalente. Una familia de clusters $\mathfrak{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$ en $\mathbb{R}^n$ se dice *separable convexamente* si existen $k$ bolas cerradas $B_1, \ldots, B_k$ en $\mathbb{R}^n$ tales que

(i) $B_i \cap B_j = \emptyset$ para todo $i \neq j$;
(ii) $\mathcal{G}_j \subset B_j$, $j = 1, \ldots, k$.

Veamos que los conceptos de separabilidad convexa y lineal son equivalentes. Para eso necesitamos un lema.

LEMA 10.6.1. *Sea $S \subset \mathbb{R}^n$ el semiespacio definido como $S = \{\mathbf{x} \in \mathbb{R}^n \colon x_1 > 0\}$. Entonces*

$$S = \bigcup_{n=1}^{\infty} B_n(n\mathbf{e}_1),$$

*donde $B_r(\mathbf{x}_0)$ denota la bola abierta de radio $r > 0$ centrada en $\mathbf{x}_0 \in \mathbb{R}^n$.*

DEMOSTRACIÓN. Observemos primero que $B_n(n\mathbf{e}_1) \subset S$ para todo $n \in \mathbb{N}$. Ver Figura 10.6.

Sea ahora $\mathbf{x} \in S$. Escribimos $\mathbf{x} = (x_1, \mathbf{x}')$ con $x_1 > 0$ y $\mathbf{x}' \in \mathbb{R}^{n-1}$. Veamos que $\mathbf{x} \in B_n(n\mathbf{e}_1)$ si $n$ es suficientemente grande. Pero

$$\begin{aligned} \|\mathbf{x} - n\mathbf{e}_1\|^2 &= (x_1 - n)^2 + |\mathbf{x}'|^2 \\ &= n^2 \left( \left(1 - \frac{x_1}{n}\right)^2 + \left|\frac{\mathbf{x}'}{n}\right|^2 \right) \\ &= n^2 \left( 1 - \frac{1}{n} \left( 2x_1 - \frac{x_1^2 + |\mathbf{x}'|^2}{n} \right) \right) < n^2, \end{aligned}$$

si

$$2x_1 - \frac{x_1^2 + |\mathbf{x}'|^2}{n} > 0 \quad \text{o, equivalentemente, si} \quad n > \frac{x_1^2 + |\mathbf{x}'|^2}{2x_1}.$$

Esto termina la demostración. □

COROLARIO 10.6.2. *Sea $K \subset \mathbb{R}^n$ un conjunto compacto y sea $S \subset \mathbb{R}^n$ un semiespacio tal que $K \subset S$. Entonces, existe una bola $B \subset S$ tal que $K \subset B$.*

DEMOSTRACIÓN. Aplicando una rotación y una traslación, podemos suponer sin pérdida de generalidad que $S = \{\mathbf{x} \in \mathbb{R}^n \colon x_1 > 0\}$.

Luego, por el Lema 10.6.1, tenemos que

$$K \subset S = \bigcup_{n=1}^{\infty} B_n(n\mathbf{e}_1).$$

Es fácil ver (y queda de ejercicio al lector), que la sucesión de bolas es creciente (ver Figura 10.6). Es decir

$$B_n(n\mathbf{e}_1) \subset B_{n+1}((n+1)\mathbf{e}_1).$$

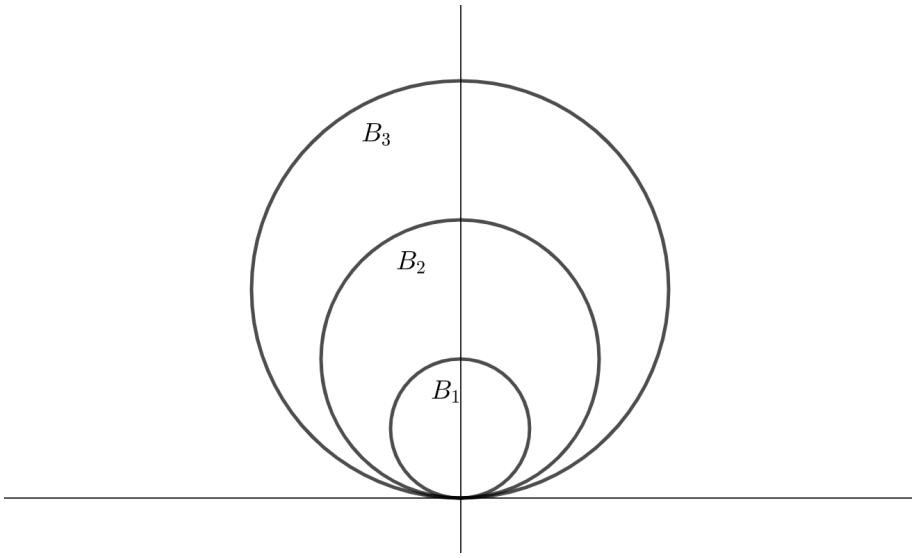



FIGURA 10.3. Las bolas de la sucesión en la demostración del Lema 10.6.1

Ahora bien, como $\{B_n(n\mathbf{e}_1)\}_{n\in\mathbb{N}}$ es un cubrimientos por abiertos de $K$ que es compacto, existe $n_1 < \cdots < n_k$ finitos, tales que

$$S \subset \bigcup_{j=1}^{k} B_{n_j}(n_j\mathbf{e}_1).$$

Pero como la sucesión de bolas es creciente, $S \subset B_{n_k}(n_k\mathbf{e}_1)$. □

Estamos ahora en condiciones de demostrar la equivalencia entre las nociones de separabilidad.

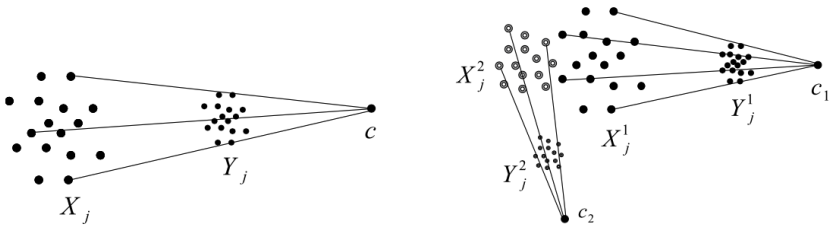
PROPOSICIÓN 10.6.3. *Dos clusters $\mathcal{G}_1$ y $\mathcal{G}_2$ en $\mathbb{R}^n$ son convexamente separables si y sólo si son linealmente separables.*

DEMOSTRACIÓN. Supongamos que $\mathcal{G}_1$ y $\mathcal{G}_2$ son separados convexamente. Entonces tenemos $B_1$ y $B_2$ dos bolas tales que $\mathcal{G}_i \subset B_i$, $i = 1, 2$, con $B_1 \cap B_2 = \emptyset$. Luego, consideramos el hiperplano $\mathcal{H}$ que tiene como normal al vector que une los centros de esas bolas y que pasa por un punto del segmento que une esos centros que esté fuera de ambas bolas. Claramente $\mathcal{H}$ separa $\mathcal{G}_1$ y $\mathcal{G}_2$.

El recíproco es una consecuencia inmediata del Corolario 10.6.2. En efecto, si $\mathcal{G}_1$ y $\mathcal{G}_2$ son linealmente separables, entonces tenemos un hiperplano $\mathcal{H}$ que separa hull($\mathcal{G}_1$) y hull($\mathcal{G}_2$). Por el Corolario 10.6.2 existen $\tilde{B}_i$, $i = 1, 2$ en cada semi-espacio determinado por $\mathcal{H}$ tal que hull($\mathcal{G}_i$) $\subset \tilde{B}_i$, $i = 1, 2$. Ahora, tomando una bola cerrada de mismo centro y radio levemente inferior, $B_i \subset \tilde{B}_i$, $i = 1, 2$ se obtiene lo pedido. □

OBSERVACIÓN 10.6.4. El concepto de separabiliad convexa es en apariencia más robusto y fácil de interpretar geométricamente, pero el de separabilidad lineal es el más sencillo de atacar usando redes neuronales. La Proposición 10.6.3 nos garantiza la equivalencia.

Con estos resultados podemos asegurar la existencia de un mapa de decisión para una familia de clusters.




**a** **b**

FIGURA 10.4. **a**. Contracción de un cluster hacia un punto. **b**. Contracción de dos clusters hacia dos puntos distintos.

PROPOSICIÓN 10.6.5. *Sea* $\mathfrak{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$ *una familia de* $k$ *clusters de* $\mathbb{R}^n$ *que son linealmente separables. Entonces existe un mapa de decisión* $F\colon \mathbb{R}^n \to \mathbb{R}^k$ *tal que* $F(\mathcal{G}_j) = \mathbf{e}_j$, $j = 1, \ldots, k$.

DEMOSTRACIÓN. Como $\mathfrak{G}$ es linealmente separable, es convexamente separable. Entonces existen $B_1, \ldots, B_k$ bolas disjuntas en $\mathbb{R}^n$ tales que $\mathcal{G}_j \subset B_j$ para todo $j = 1 \ldots, k$. Podemos entonces definir el mapa de decisión $F\colon \mathbb{R}^n \to \mathbb{R}^k$ como $F(\mathbf{x}) = \mathbf{e}_j$ si $\mathbf{x} \in B_j$ y $F(\mathbf{x}) = 0$ si $\mathbf{x} \notin \bigcup_{j=1}^k B_j$. □

## 10.7. Contracción al centro

Para poder separar clusters es usual tener que aplicar transformaciones que los contraiga hacia ciertos puntos que será luego utilizados como etiquetas.

Empecemos con el caso simple de un único cluster que es llevado hacia un punto llamado *centro*. Consideremos $\mathcal{G} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \mathbb{R}^n$ un cluster y $\mathbf{c} \in \mathbb{R}^n$ otro punto que llamaremos el centro.

Definimos entonces la transformación dada por

$$\mathbf{y}_j = \lambda \mathbf{x}_j + (1 - \lambda)\mathbf{c}, \quad \lambda \in (0, 1).$$

Para cada valor de $\lambda$ fijo, esta transformación contrae el cluster $\mathcal{G}$ hacia otro cluster $\mathcal{G}^\lambda$ situado en una proximidad del centro $\mathbf{c}$. Ver Figura 10.7 **a**. A valores más pequeños de $\lambda$, más cerca del centro $\mathbf{c}$ estrá el cluster $\mathcal{G}^\lambda$.

Consideremos ahora dos clusters $\mathcal{G}_1$ y $\mathcal{G}_2$ y dos centros distintos $\mathbf{c}_1$ y $\mathbf{c}_2$. Notemos $\mathcal{G}_1 = \{\mathbf{x}_1^1, \ldots, \mathbf{x}_{k_1}^1\}$ y $\mathcal{G}_2 = \{\mathbf{x}_1^2, \ldots, \mathbf{x}_{k_2}^2\}$ y definimos las transformaciones

$$\mathbf{y}_i^1 = \lambda_1 \mathbf{x}_i^1 + (1 - \lambda_1)\mathbf{c}_1, \quad i = 1, \ldots, k_1$$
$$\mathbf{y}_j^2 = \lambda_2 \mathbf{x}_j^2 + (1 - \lambda_2)\mathbf{c}_2, \quad j = 1, \ldots, k_2.$$

Estas transformaciones empujan cada cluster hacia su respectivo centro. Ver Figura 10.7 **b**. Los nuevos clusters, $\mathcal{G}_1^{\lambda_1}$ y $\mathcal{G}_2^{\lambda_2}$ están más separados que los originales.

Estas transformaciones están en principio sólo definidas sobre los puntos de los clusters. La dificultad radica en si es posible extender estas transformaciones a funciones definidas globalmente. La siguiente sección trata este problema.

## 10.8. Aprendiendo mapas de decisión

La clasificación de clusters se alcanza aprendiendo mapas de decisión. En esta sección estudiaremos este tema.

**10.8.1. Mapas de decisión lineales.** Consideremos dos clusters en $\mathbb{R}^2$,

$$\mathcal{G} = \{(x_1, y_1), \dots, (x_N, y_N)\} \quad \text{y} \quad \bar{\mathcal{G}} = \{(\bar{x}_1, \bar{y}_1), \dots, (\bar{x}_M, \bar{y}_M)\},$$

y asumamos que los clusters son linealmente separables, equivalentemente (ver el Ejercicio 10.10.9), que $\operatorname{hull}(\mathcal{G}) \cap \operatorname{hull}(\bar{\mathcal{G}}) = \emptyset$. Estudiemos los casos de espacios de etiquetas de dimensión uno y dos.

*Etiquetas unidimensionales.* Asociamos dos etiquetas, $\alpha$ y $\bar{\alpha}$ a cada uno de los clusters $\mathcal{G}$ y $\bar{\mathcal{G}}$ respectivamente. Como los clusters son linealmente separables resulta razonable buscar una función lineal $f\colon \mathbb{R}^2 \to \mathbb{R}$ que lleve el cluster $\mathcal{G}$ a un entorno de $\alpha$ y el cluster $\bar{\mathcal{G}}$ a un entorno de $\bar{\alpha}$. Esperamos también que el punto medio, $\frac{\alpha+\bar{\alpha}}{2}$, separe los conjuntos $f(\mathcal{G})$ y $f(\bar{\mathcal{G}})$.

Luego, buscamos $f$ de la forma

$$f(x, y) = f_{\mathbf{w},b}(x, y) = w_1 x + w_2 y + b.$$

Es decir, $f$ es la función de entrada-salida de una red neuronal feedforward con salida unidimensional sin capas ocultas. Los parámetros $\mathbf{w} = (w_1, w_2)$ y $b$ tienen que ser elegidos de forma tal que las imágenes de los clusters $f(\mathcal{G})$ y $f(\bar{\mathcal{G}})$ estén localizados en un entorno de $\alpha$ y $\bar{\alpha}$ respectivamente. Para lograr eso, se toma

$$(\mathbf{w}, b) = \arg\min F(\mathbf{w}, b),$$

donde $F$ es la función de costo cuadrático dada por

$$\begin{aligned} F(\mathbf{w}, b) &= \frac{1}{2}\sum_{i=1}^{N}(f_{\mathbf{w},b}(x_i, y_i) - \alpha)^2 + \frac{1}{2}\sum_{j=1}^{M}(f_{\mathbf{w},b}(\bar{x}_j, \bar{y}_j) - \bar{\alpha})^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}(w_1 x_i + w_2 y_i + b - \alpha)^2 + \frac{1}{2}\sum_{j=1}^{M}(w_1 \bar{x}_j + w_2 \bar{y}_j + b - \bar{\alpha})^2. \end{aligned}$$

El mínimo del costo $F$, se alcanza cuando $\nabla F = 0$. Calculemos entonces el gradiente $\nabla F = (\partial_{w_1} F, \partial_{w_2} F, \partial_b F)$. Este cálculo se realiza con una aplicación inmediata de la regla de la cadena:

$$\begin{aligned} \partial_{w_1} F = w_1 &\left(\sum_{i=1}^{N} x_i^2 + \sum_{j=1}^{M} \bar{x}_j^2\right) + w_2 \left(\sum_{i=1}^{N} x_i y_i + \sum_{j=1}^{M} \bar{x}_j \bar{y}_j\right) \\ &+ b \left(\sum_{i=1}^{N} x_i + \sum_{j=1}^{M} \bar{x}_j\right) - \alpha \sum_{i=1}^{N} x_i - \bar{\alpha} \sum_{j=1}^{M} \bar{x}_j \end{aligned}$$

$$\begin{aligned}\partial_{w_2}F &= w_1\left(\sum_{i=1}^{N}x_iy_i+\sum_{j=1}^{M}\bar{x}_j\bar{y}_j\right)+w_2\left(\sum_{i=1}^{N}y_i^2+\sum_{j=1}^{M}\bar{y}_j^2\right)\\ &\quad+b\left(\sum_{i=1}^{N}y_i+\sum_{j=1}^{M}\bar{y}_j\right)-\alpha\sum_{i=1}^{N}y_i-\bar{\alpha}\sum_{j=1}^{M}\bar{y}_j\\ \partial_bF &= w_1\left(\sum_{i=1}^{N}x_i+\sum_{j=1}^{M}\bar{x}_j\right)+w_2\left(\sum_{i=1}^{N}y_i+\sum_{j=1}^{M}\bar{y}_j\right)\\ &\quad+(N+M)b-N\alpha-M\bar{\alpha}.\end{aligned}$$

En base a estos cálculos, obtener $\nabla F=0$ es un problema de álgebra lineal. Escribamos las matrices involucradas.

$$A=\begin{pmatrix}\sum x_i^2 & \sum x_iy_i & \sum x_i\\ \sum x_iy_i & \sum y_i^2 & \sum y_i\\ \sum x_i & \sum y_i & N\end{pmatrix},\qquad \bar{A}=\begin{pmatrix}\sum \bar{x}_j^2 & \sum \bar{x}_j\bar{y}_j & \sum \bar{x}_j\\ \sum \bar{x}_j\bar{y}_j & \sum \bar{y}_j^2 & \sum \bar{y}_j\\ \sum \bar{x}_j & \sum \bar{y}_j & M\end{pmatrix}. \tag{10.8.1}$$

Estas matrices $A$ y $\bar{A}$ contienen la información de los clusters $\mathcal{G}$ y $\bar{\mathcal{G}}$ respectivamente. Sus entradas son los primeros y segundos momentos de las variables $x$ e $y$ de los datos y su correlación. Consideremos ahora los vectores

$$\beta=\alpha\begin{pmatrix}\sum x_i\\ \sum y_i\\ N\end{pmatrix},\qquad \bar{\beta}=\bar{\alpha}\begin{pmatrix}\sum \bar{x}_i\\ \sum \bar{y}_i\\ M\end{pmatrix},$$

que dependen de las etiquetas y de los primeros momentos de los datos. Luego, la ecuación $\nabla F=0$ se escribe como

$$(A+\bar{A})X=\beta+\bar{\beta},$$

donde $X=(w_1,w_2,b)^T$.

La resolución de esta ecuación se puede realizar por varios métodos como los que estudiaron en Álgebra Lineal Computacional o en Elementos de Cálculo Numérico (ver también el Apéndice G de [**5**]).

Las etiquetas $\alpha$ y $\bar{\alpha}$ consideradas pueden tomarse, por simplicidad, como $\alpha=1$ y $\bar{\alpha}=0$.

Este método puede extenderse sin dificultades a cualquier número de clusters siempre que sean linealmente separables.

*Etiquetas bidimensionales.* Vamos a realizar ahora una clasificación de los clusters $\mathcal{G}$ y $\bar{\mathcal{G}}$ usando etiquetas bidimensionales, tales como vectores o distintos puntos del plano. Asumamos que la etiqueta del cluster $\mathcal{G}$ es el punto $\mathbf{a}=(a_1,a_2)$ y la etiqueta de $\bar{\mathcal{G}}$ es el punto $\bar{\mathbf{a}}=(\bar{a}_1,\bar{a}_2)$.

Como estamos suponiendo que los clusters son linealmente separables, buscamos una función $f\colon\mathbb{R}^2\to\mathbb{R}^2$ lineal que mande $\mathcal{G}$ en un entorno de $\mathbf{a}$ y $\bar{\mathcal{G}}$ en un entorno de $\bar{\mathbf{a}}$. Esta función lineal $f=(f^1,f^2)$ se construye como la función de entrada-salida de una red neuronal feedforward con una salida de dimensión 2 sin capas ocultas. Esto

significa que $f$ es de la forma

$$f^1(x,y) = f^1_{W,b}(x,y) = w_{11}x + w_{21}y + b_1$$
$$f^2(x,y) = f^2_{W,b}(x,y) = w_{12}x + w_{22}y + b_2.$$

Los pesos $w_{ij}$ y los sesgos $b_j$ tienen que ser ajustados de manera tal que los conjuntos imagen $f(\mathcal{G})$ y $f(\bar{\mathcal{G}})$ estén lo más próximos posibles a sus etiquetas $\mathbf{a}$ y $\bar{\mathbf{a}}$ respectivamente. Este objetivo se consigue tomando

$$(W,b) = \arg\min G(W,b),$$

donde $G$ es la suma de los cuadrados de las distancias a las etiquetas, i.e.

$$\begin{aligned} G(W,b) &= \frac{1}{2}\sum_{i=1}^{N} \|f_{W,b}(x_i,y_i) - \mathbf{a}\|^2 + \frac{1}{2}\sum_{j=1}^{M} \|f_{W,b}(\bar{x}_j,\bar{y}_j) - \bar{\mathbf{a}}\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{N} \left[(f^1(x_i,y_i) - a_1)^2 + (f^2(x_i,y_i) - a_2)^2\right] \\ &\quad + \frac{1}{2}\sum_{j=1}^{M} \left[(f^1(\bar{x}_j,\bar{y}_j) - \bar{a}_1)^2 + (f^2(\bar{x}_j,\bar{y}_j) - \bar{a}_2)^2\right] \\ &= G_1(w_{11},w_{21},b_1) + G_2(w_{12},w_{22},b_2), \end{aligned}$$

donde

$$G_1(w_{11},w_{21},b_1) = \frac{1}{2}\sum_{i=1}^{N}(w_{11}x_i + w_{21}y_i + b_1 - a_1)^2 + \frac{1}{2}\sum_{j=1}^{M}(w_{11}\bar{x}_j + w_{21}\bar{y}_j + b_1 - \bar{a}_1)^2$$

$$G_2(w_{12},w_{22},b_2) = \frac{1}{2}\sum_{i=1}^{N}(w_{12}x_i + w_{22}y_i + b_2 - a_2)^2 + \frac{1}{2}\sum_{j=1}^{M}(w_{12}\bar{x}_j + w_{22}\bar{y}_j + b_2 - \bar{a}_2)^2$$

Observemos que ahora tenemos

$$\nabla G(W,b) = (\nabla G_1(w_{11},w_{21},b_1), \nabla G_2(w_{12},w_{22}.b_2)) \in \mathbb{R}^6$$

y el sistema $\nabla G = 0$ se desacopla en $\nabla G_1 = 0$ y $\nabla G_2 = 0$ donde cada una de esas ecuaciones está en $\mathbb{R}^3$. Estas ecuaciones en $\mathbb{R}^3$ son las mismas que teníamos en la sección anterior, por lo que las mismas se escriben como

$$(A + \bar{A})X_1 = \beta_1 + \bar{\beta}_1$$
$$(A + \bar{A})X_2 = \beta_2 + \bar{\beta}_2,$$

donde $A$ y $\bar{A}$ vienen dadas por (10.8.1), $X_k = (w_{1k}, w_{2k}, b_k)^T$, $k = 1, 2$ y

$$\beta_k = a_k \begin{pmatrix} \sum x_i \\ \sum y_i \\ N \end{pmatrix}, \qquad \bar{\beta}_k = \bar{a}_k \begin{pmatrix} \sum \bar{x}_i \\ \sum \bar{y}_i \\ M \end{pmatrix}, \qquad k = 1, 2.$$

Estas ecuaciones se resuelven de la misma manera que en la sección previa.

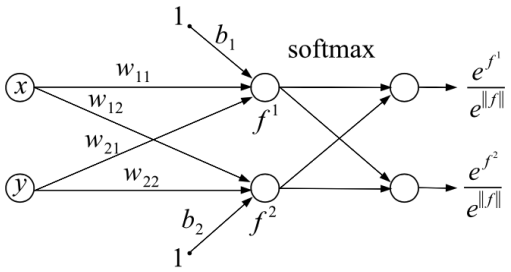



FIGURA 10.5. Red neuronal con una capa extra implementando la función softmax.

*El uso de la función de activación* softmax. En la mayoría de los problemas de clasificación, se utiliza una capa adicional que implementa la función softmax. Esta función se define como

$$\text{softmax}\colon \mathbb{R}^n \to \mathbb{R}^n, \quad \text{softmax}(\mathbf{x}) = \frac{1}{\sum_{i=1}^n e^{x_i}}(e^{x_1}, \dots, e^{x_n}).$$

El objetivo de aplicar esta función es la de normalizar las salidas de la red para entregar un valor entre 0 y 1 para cada coordenada y cumplir el rol de *densidad de probabilidad* de que la salida sea un vector one-hot $\mathbf{e}_i$. Utilicemos esto para el ejemplo de la sección previa tomando como vectores $\mathbf{a} = \mathbf{e}_1 = (1, 0)$ y $\bar{\mathbf{a}} = \mathbf{e}_2 = (0, 1)$.

La salida de la nueva red será entonces $\mathbf{z} = (z_1, z_2)$, donde $\mathbf{z} = \text{softmax}(f)$, es decir

$$z_i = \frac{e^{f^i}}{e^{f^1} + e^{f^2}}, \quad i = 1, 2.$$

Como $z_i > 0$ y $z_1 + z_2 = 1$ sigue que $\mathbf{z}$ pertenece al segmento que une los puntos $\mathbf{a}$ y $\bar{\mathbf{a}}$, con la imagen de $\mathcal{G}$ más próxima a $\mathbf{a}$ y la imagen de $\bar{\mathcal{G}}$ más próxima a $\bar{\mathbf{a}}$.

Asumamos que el punto de separación de los clusters es el punto medio del segmento, $(\frac{1}{2}, \frac{1}{2})$. Luego, dado un punto $(x, y) \in \mathbb{R}^2$ para decidir a qué cluster pertenece corremos el punto a través de la red neuronal (ver Figura 10.8.1) y si la salida pertenece a la parte superior del segmento $[\mathbf{a}, \bar{\mathbf{a}}]$, entonces el punto pertenece al cluster $\mathcal{G}$. Caso contrario, el punto pertenece al cluster $\bar{\mathcal{G}}$.

Este test se puede realizar, por ejemplo, usando la coordenada $z_1$., se tiene

- Si $z_1 < \frac{1}{2}$, entonces $(x, y) \in \mathcal{G}$;
- Si $z_1 > \frac{1}{2}$, entonces $(x, y) \in \bar{\mathcal{G}}$.

El uso de softmax sirve también para problemas de clasificación que involucran más de 2 clusters. Por ejemplo, en el caso de 3 clusters, $\mathcal{G}_1$, $\mathcal{G}_2$ y $\mathcal{G}_3$, se asocian las etiquetas $\mathbf{a}_1 = (1, 0, 0)$, $\mathbf{a}_2 = (0, 1, 0)$ y $\mathbf{a}_3 = (0, 0, 1)$ que forman un triángulo equilátero en $\mathbb{R}^3$. Ver Figura 10.8.1. En este caso, el punto de separación se cambia por un sistema de separación de curvas. Ver Figura 10.8.1.

Para decidir a qué región de decisión pertenece un punto determinado $\mathbf{x} \in \mathbb{R}^3$, es suficiente evaluar la distancia a los vértices $\text{dist}(\mathbf{x}, \mathbf{a}_i)$. Luego, si $\text{dist}(\mathbf{x}, \mathbf{a}_i) < \frac{1}{2}$, entonces $\mathbf{x} \in \mathcal{G}_i$.

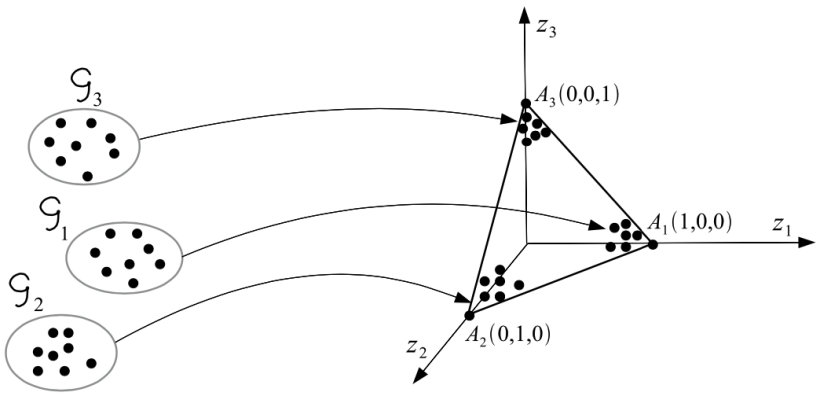



FIGURA 10.6. Los clusters $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$ separables linealmente son mapeados en los vértices de una región triangular.

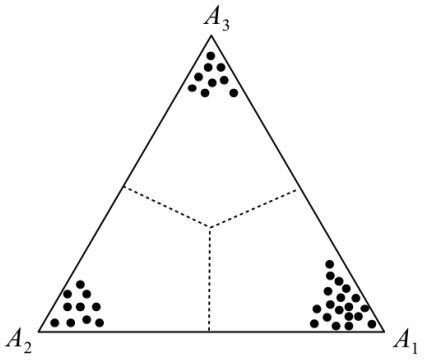



FIGURA 10.7. Las líneas punteadas separan al triángulo en tres regiones de decisión.

**10.8.2. Mapas de decisión no lineales.** Como en el inicio de la sección anterior, consideremos dos clusters en $\mathbb{R}^2$,

$$\mathcal{G} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \quad \text{y} \quad \bar{\mathcal{G}} = \{(\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_M, \bar{y}_M)\},$$

y asumamos que $\text{hull}(\mathcal{G}) \cap \text{hull}(\bar{\mathcal{G}}) \neq \emptyset$.

Asociamos a cada cluster dos números reales $\alpha$ y $\bar{\alpha}$ respectivamente. Como los clusters no son linealmente separables, buscamos una función no lineal $f\colon \mathbb{R}^2 \to \mathbb{R}$ que mapee el cluster $\mathcal{G}$ a un entorno de $\alpha$ y al cluster $\bar{\mathcal{G}}$ a un entorno de $\bar{\alpha}$.

*La neurona sigmoide.* El caso más simple de una función no lineal es producido por una neurona sigmoide, con función de entrada-salida dada por

$$f(x, y) = f_{\mathbf{w},b}(x, y) = \sigma(w_1 x + w_2 y + b),$$

donde $\sigma(x)$ es la función logística, $\sigma(x) = (1 + e^{-x})^{-1}$ (ver Figura 10.8.2). Asumamos que las etiquetas se eligen de forma tal que $\alpha = 0$ y $\bar{\alpha} = 1$, luego el cluster $\mathcal{G}$ se debe mapear cerca de 0 y el cluster $\bar{\mathcal{G}}$ cerca de 1.

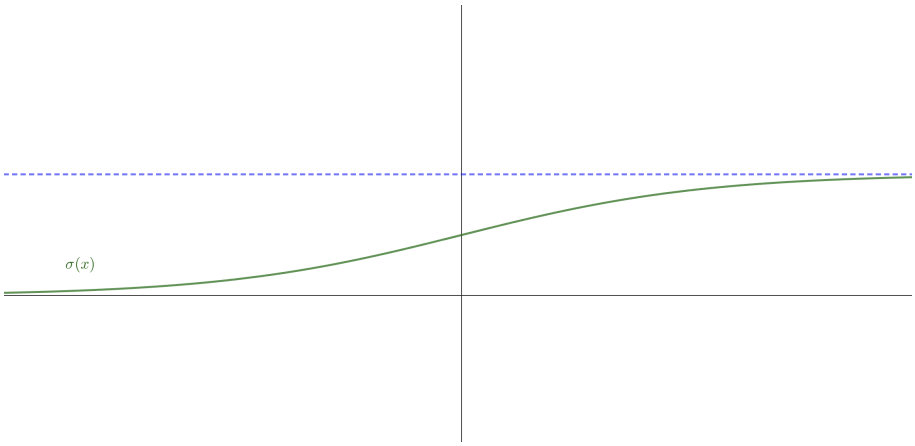



FIGURA 10.8. La función logística $\sigma(x)$.

Para que este mapa localice lo mejor posible, necesitamos minimizar la función de costo

$$F(\mathbf{w}, b) = \frac{1}{2}\sum_{i=1}^{N} \sigma(w_1 x_i + w_2 y_i + b)^2 + \frac{1}{2}\sum_{j=1}^{M} (\sigma(w_1 \bar{x}_j + w_2 \bar{y}_j + b) - 1)^2.$$

Usando la propiedad de la función logística, $\sigma' = \sigma(1-\sigma)$, obtenemos

$$\begin{aligned}
\partial_{w_1} F &= \sum_{i=1}^{N} x_i \sigma^2(1-\sigma)|_{w_1 x_i + w_2 y_i + b} - \sum_{j=1}^{M} \bar{x}_j \sigma(1-\sigma)^2|_{w_1 \bar{x}_j + w_2 \bar{y}_j + b} \\
\partial_{w_2} F &= \sum_{i=1}^{N} y_i \sigma^2(1-\sigma)|_{w_1 x_i + w_2 y_i + b} - \sum_{j=1}^{M} \bar{y}_j \sigma(1-\sigma)^2|_{w_1 \bar{x}_j + w_2 \bar{y}_j + b} \\
\partial_{b} F &= \sum_{i=1}^{N} \sigma^2(1-\sigma)|_{w_1 x_i + w_2 y_i + b} - \sum_{j=1}^{M} \sigma(1-\sigma)^2|_{w_1 \bar{x}_j + w_2 \bar{y}_j + b}.
\end{aligned}$$

Este es un sistema no lineal y no se tiene una forma cerrada para la solución del mismo. La forma de encontrar el mínimo entonces es aplicando el método de descenso de gradiente. La sucesión aproximante es dada de forma iterativa por

$$\begin{aligned}
w_1^{(n+1)} &= w_1^{(n)} - \eta \partial_{w_1} F(w_1^{(n)}, w_2^{(n)}, b^{(n)}) \\
w_2^{(n+1)} &= w_2^{(n)} - \eta \partial_{w_2} F(w_1^{(n)}, w_2^{(n)}, b^{(n)}) \\
b^{(n+1)} &= b^{(n)} - \eta \partial_{b} F(w_1^{(n)}, w_2^{(n)}, b^{(n)}),
\end{aligned}$$

donde $\eta > 0$ es la tasa de aprendizaje. Como valores iniciales se pueden tomar $w_1^{(0)} = w_2^{(0)} = b^{(0)} = 0$.

En principio no es obvio como extender este método a más de dos clusters.

*Una red con una capa escondida.* Asumamos que la función no lineal $f$ es el mapa de entrada-salida de una red neuronal feedforward con tres capas, que tiene una salida unidimensional y una capa oculta. Las neuronas en la capa oculta poseen una función

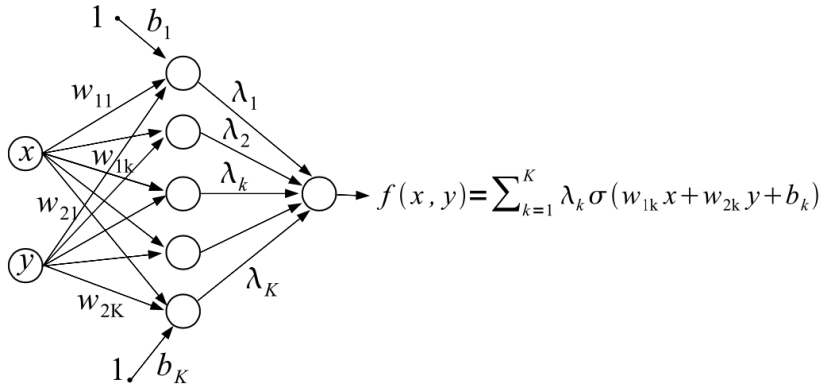



FIGURA 10.9. Una red neuronal con una capa oculta y salida unidimensional.

de activación no lineal. Entonces el mapa resulta

$$f(x,y) = f_{W,\lambda,b}(x,y) = \sum_{k=1}^{K} \lambda_k \sigma(w_{1k}x + w_{2k}y + b_k),$$

donde $K$ es el número de neuronas en la capa oculta, $w_{ij}$ son los pesos de la entrada en la capa oculta, $b_k$ los sesgos de estas neuronas, y $\lambda_k$ son los pesos de la capa oculta hacia la capa de salida, Ver la Figura 10.8.2. La función $f_{W,\lambda,b}$ depende de $4K$ parámetros ($2K$ pesos $w_{ij}$, $K$ sesgos $b_k$, y $K$ pesos $\lambda_k$).

Los parámetros deben ser ajustados de forma de minimizar la función de costo

$$G(W,b,\lambda) = \frac{1}{2}\sum_{i=1}^{N}(f_{W,\lambda,b}(x_i,y_i) - \alpha)^2 + \frac{1}{2}\sum_{j=1}^{M}(f_{W,\lambda,b}(\bar{x}_j,\bar{y}_j) - \bar{\alpha})^2, \tag{10.8.2}$$

donde $\alpha$ y $\bar{\alpha}$ son las etiquetas asociadas con $\mathcal{G}$ y $\bar{\mathcal{G}}$ respectivamente.

Nuevamente, no se tiene una forma cerrada para el minimizante de $G$, por lo que se necesita aplicar un método de descenso de gradiente. Ver el Ejercicio 10.10.10.

En el caso de más de dos clusters es conveniente el uso de vectores one-hot como etiquetas. En este caso el espacio de etiquetas es $\mathbb{R}^p$, donde $p$ es el número de clusters que necesitan ser clasificados. Ver el Ejercicio 10.10.11.

## 10.9. Resumen

Durante el proceso de clasificación de los clusters, las etiquetas deben asignarse con precisión a cada uno. Existen varios tipos de etiquetas que pueden utilizarse: números, vectores one-hot, puntos, etc. La función que asigna etiquetas a los clusters se denomina mapa de decisión, y puede ser aprendida por redes neuronales.

Dos clusters son linealmente separables si sus cápsulas convexas son disjuntas. En este caso, un perceptrón simple puede aprender la función de decisión asociada.

Si las cápsulas convexas de dos clusters no son disjuntas, entonces no son linealmente separables, y en este caso se requiere una red neuronal con función de activación no lineal para aprender la función de decisión correspondiente.

El uso de la función de activación softmax en la última capa de una red neuronal asigna los n clusters a los vértices de un politopo de dimensión $(n-1)$.

## 10.10. Ejercicios

EJERCICIO 10.10.1. Sea $S = (I_1 \times I_1) \cap (\mathbb{Q} \times \mathbb{Q})$ el conjunto de los puntos con coordenadas racionales en $[0,1] \times [0,1]$.

(a) Demuestre que $S$ es una relación transitiva.
(b) ¿Es $S$ una relación de equivalencia?

EJERCICIO 10.10.2. Demuestre que $k$ puntos están en posición general si y solo si existe un único hiperplano de dimensión $(k-1)$ que los contiene.

EJERCICIO 10.10.3. Sean $P_1, \ldots, P_k$ $k$ puntos en $\mathbb{R}^k$ en posición general, y sea $\mathcal{H} = \mathbb{S} + \mathbf{p}$ el único hiperplano de dimensión $(k-1)$ que los contiene. Demuestre que el conjunto $\{P_2 - P_1, \ldots, P_k - P_1\}$ forma un sistema de vectores independientes en $\mathbb{R}^k$ que genera $\mathbb{S}$.

EJERCICIO 10.10.4. Considere dos números reales distintos $x_1, x_2 \in \mathbb{R}$. Demuestre que existe una única función afín $f(x) = ax + b$ tal que $f(x_1) = 1$ y $f(x_2) = 2$.

EJERCICIO 10.10.5. Sea $\mathcal{G}$ un cluster en $\mathbb{R}^n$.

(a) Demuestre que $\text{hull}(\mathcal{G})$ es un conjunto convexo.
(b) Pruebe que $\text{hull}(\mathcal{G})$ es el conjunto convexo más pequeño en $\mathbb{R}^n$ que contiene a $\mathcal{G}$.
(c) Demuestre la Proposición 10.5.3.

EJERCICIO 10.10.6. Sea $\Phi\colon \mathbb{R}^2 \to \mathbb{R}^2$ una función afín no degenerada, es decir, $\Phi(\mathbf{x}) = W\mathbf{x} + \mathbf{b}$ con $\det W \neq 0$. Considere dos clusters linealmente separables en $\mathbb{R}^2$, $\mathcal{G}_1$ y $\mathcal{G}_2$. Demuestre que $\Phi(\mathcal{G}_1)$ y $\Phi(\mathcal{G}_2)$ son linealmente separables. En otras palabras, las funciones afines preservan la separabilidad lineal.

EJERCICIO 10.10.7. Sea $\mathcal{G}$ un cluster finito de puntos y denote por $g$ su centro de gravedad. Demuestre que $g \in \text{hull}(\mathcal{G})$.

EJERCICIO 10.10.8. Demuestre que una familia de clusters $\mathfrak{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$ en $\mathbb{R}^n$ es convexamente separable si y solo si la familia es mutuamente convexamente separable, es decir, cualquier par de clusters $\mathcal{G}_i$ y $\mathcal{G}_j$ son convexamente separables para todo $i \neq j$.

EJERCICIO 10.10.9. (a) Si $A$ y $B$ son dos conjuntos convexos tales que $A \cap B = \emptyset$, entonces $A$ y $B$ son linealmente separables.
(b) Si $\mathcal{G}_1$ y $\mathcal{G}_2$ son dos clusters tales que $\text{hull}(\mathcal{G}_1) \cap \text{hull}(\mathcal{G}_2) = \emptyset$, entonces $\mathcal{G}_1$ y $\mathcal{G}_2$ son linealmente separables.

EJERCICIO 10.10.10. (a) Encuentre el gradiente $\nabla_{W,b,\lambda} G$, donde $G$ está dado por (10.8.2).
(b) Escriba la recursión del descenso de gradiente para la secuencia de aproximaciones del mínimo.

EJERCICIO 10.10.11. Considere $p$ clusters, $\mathcal{G}_1, \ldots, \mathcal{G}_p$, de puntos en $\mathbb{R}^2$. Escriba la función de costo que asocia el vector $\mathbf{e}_j$ como etiqueta para el cluster $\mathcal{G}_j$, para todo $j = 1, \ldots, p$.

# Bibliografía


[1] Andrew R. Barron, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Trans. Inform. Theory **39** (1993), no. 3, 930–945. MR 1237720

[2] Taweh Beysolow II, *Introduction to deep learning using R: A step-by-step guide to learning and implementing deep learning models using R*, Apress, 2017.

[3] Fred Brauer and John A. Nohel, *The qualitative theory of ordinary differential equations: An introduction*, reprint ed., Dover Books on Mathematics, Dover Publications, 1989.

[4] Haim Brezis, *Functional analysis, Sobolev spaces and partial differential equations*, Universitext, Springer, New York, 2011. MR 2759829

[5] Ovidiu Calin, *Deep learning architectures*, Springer, 2020.

[6] Francois Chollet and François Chollet, *Deep learning with python*, Simon and Schuster, 2021.

[7] Albon Chris, *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning*, NY: O'Reilly Media (2018).

[8] Franois Duval, *Deep learning for beginners: Practical guide with python and tensorflow*, 2017.

[9] E. B. Dynkin, *Markov processes. Vols. I, II*, Die Grundlehren der mathematischen Wissenschaften, Band 121, vol. 122, Springer-Verlag, Berlin-Göttingen-Heidelberg; Academic Press, Inc., Publishers, New York, 1965, Translated with the authorization and assistance of the author by J. Fabius, V. Greenberg, A. Maitra, G. Majone. MR 193671

[10] Lawrence C. Evans, *An introduction to stochastic differential equations*, American Mathematical Society, Providence, RI, 2013. MR 3154922

[11] J. Fernández Bonder, *Ecuaciones diferenciales parciales*, Cursos de grado, Departamento de Matemática, FCEN-UBA, 2015.

[12] J. Fernández Bonder and P. Groisman, *Explosiones en ecuaciones diferenciales estocásticas*, Cursos y Seminarios de Matemática - Serie B, Departamento de Matematica, FCEN-UBA, 2007.

[13] Aurélien Géron, *Concepts, tools, and techniques to build intelligent systems*, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. London, United Kingdom: O'Reilly Media (2017).

[14] Ian Goodfellow, *Deep learning*, 2016.

[15] M. L. Minsky and S.A. Papert, *Perceptrons*, MIT Press, Cambridge, 1969.

[16] A Nielsen, *Neural networks and deep learning*, 2015.

[17] Josh Patterson and Adam Gibson, *Deep learning: A practitioner's approach*, "O'Reilly Media, Inc.", 2017.

[18] Sebastian Raschka and Vahid Mirjalili, *Python machine learning: Machine learning and deep learning with python, scikit-learn, and tensorflow 2*, Packt publishing ltd, 2019.

[19] Raúl Rojas, *Neural networks: a systematic introduction*, Springer Science & Business Media, 2013.

[20] H. L. Royden, *Real analysis*, third ed., Macmillan Publishing Company, New York, 1988. MR 1013117

[21] Richard L. Wheeden and Antoni Zygmund, *Measure and integral*, second ed., Pure and Applied Mathematics (Boca Raton), CRC Press, Boca Raton, FL, 2015, An introduction to real analysis. MR 3381284

[22] Norbert Wiener, *A note on Tauberian theorems*, Ann. of Math. (2) **33** (1932), no. 4, 787. MR 1503094