4 El Perceptrón Simple

4.1 Introducción

Una de las características más significativas de las redes neuronales es su capacidad para aprender a partir de alguna fuente de información interactuando con su entorno. En 1958 el psicólogo Frank Ronsenblant desarrolló un modelo simple de neurona basado en el modelo de McCulloch y Pitts y en una regla de aprendizaje basada en la corrección del error. A este modelo le llamó Perceptrón. Una de las características que más interés despertó de este modelo fue su capacidad de aprender a reconocer patrones. El Perceptrón está constituido por conjunto de sensores de entrada que reciben los patrones de entrada a reconocer o clasificar y una neurona de salida que se ocupa de clasificar a los patrones de entrada en dos clases, según que la salida de la misma se 1 (activada) o 0 (desactivada). Sin embargo, dicho modelo tenía muchas limitaciones, como por ejemplo, no es capaz de aprender la función lógica XOR. Tuvieron que pasar unos años hasta que se propusiera la regla de aprendizaje de retropropagación del error para demostrarse que el Perceptrón multicapa es un aproximador universal.

4.2 El Perceptrón simple

Supongamos que tenemos una función f de R^n en {-1,1}, que aplica un patrón de entrada $\mathbf{x} = (x_1,x_2,\ldots,x_n)^T \in R^n$ en la salida deseada $z \in \{-1,1\}$, es decir, $f(\mathbf{x}) = z$. La información de que disponemos sobre dicha función viene dada por p pares de patrones de entrenamiento

$$\{\mathbf{x}^1,z^1\}, \{\mathbf{x}^2,z^2\},\ldots,\{\mathbf{x}^p,z^p\}$$

donde $\mathbf{x}^i \in R^n$ y $f(\mathbf{x}^i) = z^i \in \{-1,1\}$, $i=1,2,\ldots,p$. Dicha función realiza una partición en el espacio R^n de patrones de entrada; por una parte estarían los patrones con salida +1 y por otra parte los patrones con salida –1. Por lo tanto, diremos que la función f clasifica a los patrones de entrada en dos clases. Ejemplos de funciones f de este tipo son la función lógica OR o la función par.

Ahora vamos a construir un dispositivo sencillo que aprenda dicha función a partir de un conjunto conocido de patrones (relaciones) de entrenamiento. Para ello vamos a utilizar una **unidad de proceso bipolar** que como vimos es una función matemática con dominio el conjunto n-dimensional $\{-1,1\}^n$ y rango el conjunto {-1,1}, definida por la siguiente expresión:

$$f(x_1,x_2,\ldots,x_n) = \begin{cases} 1 & \text{si} \quad w_1x_1 + w_2x_2 + \ldots + w_nx_n \geq \theta \\ -1 & \text{si} \quad w_1x_1 + w_2x_2 + \ldots + w_nx_n < \theta \end{cases} \qquad (1)$$

donde los parámetros $w_1,w_2,\ldots,w_n$, se llaman **pesos sinápticos** y son los pesos con los que se ponderan los valores de entrada $x_1,x_2,\ldots,x_n$, o argumentos de la función; la suma ponderada $u = w_1x_1 + w_2x_2 + \ldots + w_nx_n$ se llama **potencial sináptico** y el parámetro θ se llama **umbral** o sesgo. También se puede expresar la función f mediante la función signo, es decir,

$$f(x_1, x_2, \ldots, x_n) = \text{sgn}(u - \theta)$$

siendo la función signo,

$$\text{sgn}(x) = \begin{cases} 1 & \text{x} \geq 0 \\ -1 & \text{x} < 0 \end{cases}$$

y diremos que en este caso la función de transferencia es la función signo. Análogamente, se define una unidad de proceso binaria como una función matemática con dominio el conjunto n-dimensional $\{0,1\}^n$ y rango el conjunto $\{0,1\}$, definida por la siguiente expresión:

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{si} \quad w_1x_1 + w_2x_2 + \ldots + w_nx_n \geq \theta \\ 0 & \text{si} \quad w_1x_1 + w_2x_2 + \ldots + w_nx_n < \theta \end{cases} \qquad (2)$$

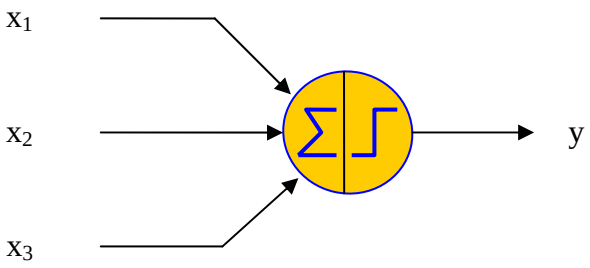
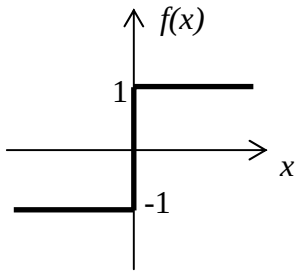


Figura 1. Unidad de proceso bipolar.

Cuando la salida de la unidad de proceso es igual a 1 se dice que dicha unidad de proceso está activada o encendida y presenta el estado 1, mientras que si su salida es igual a cero se dice que está desactivada o apagada, presentando el estado 0.

Función signo



Función paso o De Heaviside

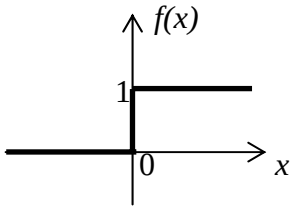


Figura 2. Funciones de transferencia.

Para la determinación de los pesos sinápticos y del umbral vamos a seguir un proceso adaptativo que consiste en comenzar con unos valores iniciales aleatorios e ir modificándolos iterativamente cuando la salida de la unidad no coincide con la salida deseada. La regla que vamos a seguir para modificar los pesos sinápticos se conoce con el nombre de **regla de aprendizaje del Perceptrón simple** y viene dada por la expresión:

$$w_j(k+1) = w_j(k) + \Delta w_j(k), \quad k = 1,2,\ldots$$

siendo

$$\Delta w_j(k) = \eta(k)[z(k) - y(k)]x_j(k) \tag{3}$$

esto nos indica que la variación del peso w_j es proporcional al producto del error $z_i(k) - y_i(k)$ por la componente *j*-ésima del patrón de entrada que hemos introducido en la iteración *k*, es decir, $x_j(k)$. La constante de proporcionalidad $\eta(k)$ es un parámetro positivo que se llama **tasa de aprendizaje** puesto que cuanto mayor es más se modifica el peso sináptico y viceversa. Es decir, es el parámetro que controla el proceso de aprendizaje. Cuando es muy pequeño la red aprende poco a poco. Cuando se toma constante en todas las iteraciones, $\eta(k) = \eta > 0$ tendremos la **regla de adaptación con incremento fijo**.

Cuando la función de transferencia usada es la función signo (valores bipolares) la regla de aprendizaje se puede escribir de la forma:

$$w_j(k+1) = \begin{cases} w_j(k) + 2\eta x_j(k) & \text{si } y(k) = -1 \text{ y } z(k) = 1, \\ w_j(k) & \text{si } y(k) = z(k) \\ w_j(k) - 2\eta x_j(k) & \text{si } y(k) = 1 \text{ y } z(k) = -1 \end{cases}$$

Por lo tanto, esta regla de aprendizaje es un método de detección del error y corrección. Solo aprende, es decir, modifica los pesos, cuando se equivoca. Cuando tenemos un patrón que pertenece a la primera clase (*z(k)*=1) y no es asignado a la misma, entonces corrige el valor del peso sináptico añadiéndole una cantidad proporcional al valor de entrada, es decir lo *refuerza*, mientras que si el patrón de entrada no pertenece a esta clase y el Perceptrón lo asigna a ella, lo que hace es *debilitar* el peso restándole una cantidad proporcional al patrón de entrada . No modificaremos los pesos cuando el valor deseado coincida con la salida de la red.

¿Cómo se modifica el sesgo? De la misma manera, teniendo en cuenta que el sesgo se puede considerar como el peso sináptico correspondiente a un nuevo sensor de entrada que tiene siempre una entrada igual a $x_{n+1} = -1$, y como peso sináptico el valor del umbral, pues

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n \geq \theta \quad \Leftrightarrow \quad w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_{n+1}x_{n+1} \geq 0$$

cuando $w_{n+1} = \theta$ y $x_{n+1} = -1$. Así, la red equivalente tendría *n+1* sensores, su umbral sería siempre cero, los patrones de entrada $(x_1, x_2, \ldots, x_n)$ serán ahora $(x_1, x_2, \ldots, x_n, -1)$,

los pesos asociados ($w_1, w_2, ..., w_n$) serán ($w_1, w_2, ..., w_n, w_{n+1}$) con $w_{n+1} = \theta$ y la regla de aprendizaje:

$$\Delta\theta(k) = -\eta(k)[z(k) - y(k)], \quad k=1,2,\ldots \tag{4}$$

A partir de ahora vamos a considerar el umbral como un peso sináptico más.

Algoritmo de aprendizaje del Perceptrón simple

Paso 0: Inicialización

Inicializar los pesos sinápticos con números aleatorios del intervalo [-1,1]. Ir al paso 1 con *k*=1

Paso 1: (k-ésima iteración)

Calcular

$$y(k) = \operatorname{sgn}\left(\sum_{j=1}^{n+1} w_j x_j(k) \right)$$

Paso 2: Corrección de los pesos sinápticos

Si $z(k) \neq y(k)$ modificar los pesos sinápticos según la expresión:

$$w_j(k+1) = w_j(k) + \eta[z_i(k) - y_i(k)]x_j(k), \quad j = 1,2,...,n+1$$

Paso 3: Parada

Si no se han modificado los pesos en las últimas *p* iteraciones, es decir,

$$w_j(r) = w_j(k), \quad j = 1,2,...,n+1, \; r = k+1,...,k+p,$$

parar. La red se ha estabilizado.

En otro caso, ir al **Paso 1** con *k*=*k*+1.

Ahora surge la cuestión: ¿Dado un conjunto de patrones de entrenamiento, puede el Perceptrón aprender a clasificarlos correctamente? Solamente si los patrones son linealmente separables. Ello reduce considerablemente el campo de aplicaciones del Perceptrón simple puesto que ni siquiera es capaz de implementar la función lógica XOR dada por la siguiente relación:

Entradas	Salidas
(1, 1)	−1
(1, −1)	1
(−1, 1)	1
(−1,−1)	−1

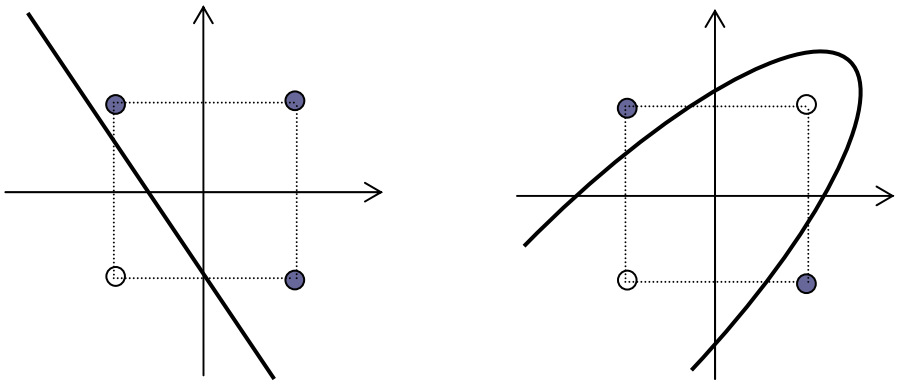


Figura . (a) Patrones separables linealmente. (b) Patrones no separables linealmente.

Decimos que dos conjuntos de puntos *A* y *B* son ***linealmente separables*** en un espacio *n*-dimensional si existen *n+1* números reales $w_1, w_2, ..., w_n, \theta,$ de manera que cada punto $(x_1, x_2, ..., x_n) \in A$ satisface $\sum_{i=1}^{n} w_i x_i \geq \theta$ y cada punto $(x_1, x_2, ..., x_n) \in B$ satisface $\sum_{i=1}^{n} w_i x_i < \theta$. A continuación vamos a estudiar la convergencia del Perceptrón simple, es decir, bajo que condiciones un Perceptrón es capaz de encontrar una solución en un número finito de iteraciones.

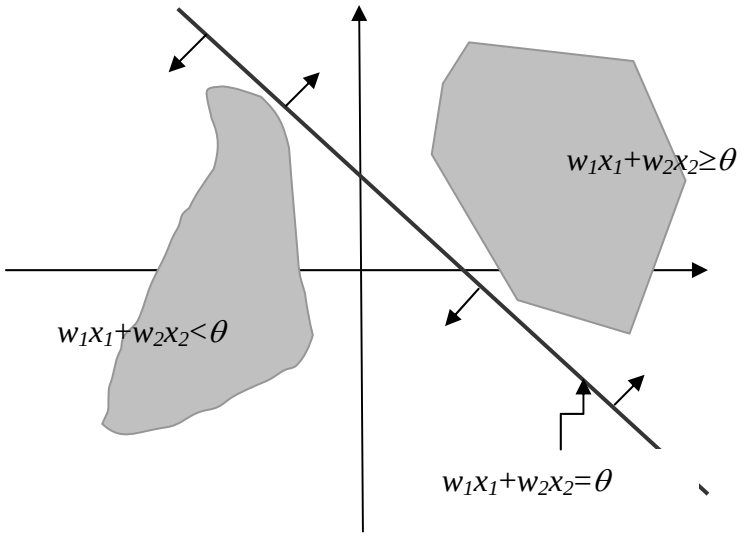


Figura . Semiespacios que define el Perceptrón.

Teorema 1 (**de convergencia del Perceptrón**)
Si el conjunto de patrones de entrenamiento,

$$\{\mathbf{x}^1, z^1\}, \{\mathbf{x}^2, z^2\}, \ldots, \{\mathbf{x}^p, z^p\},$$

es linealmente separable entonces el Perceptrón simple encuentra una solución en un número finito de iteraciones, es decir, consigue que la salida de la red coincida con la salida deseada para cada uno de los patrones de entrenamiento.

Demostración:

En efecto, como los patrones son linealmente separables existirán unos valores $w_1^*, w_2^*, ..., w_{n+1}^*$ tales que

$$\sum_{j=1}^{n} w_j^* x_j > w_{n+1} \quad \text{para los patrones de la clase 1 y}$$

$$\sum_{j=1}^{n} w_j^* x_j < w_{n+1} \quad \text{para los patrones de la clase 2.}$$

Obsérvese que w_{n+1} es el umbral θ.

Supongamos que en la iteración *k* la red tiene que modificar los pesos sinápticos según la regla de aprendizaje, puesto que la salida de la red *y(k)* no coincide con la salida deseada *z(k)*. Tendremos que:

$$\sum_{j=1}^{n+1} \left(w_j(k+1) - w_j^*\right)^2 = \sum_{j=1}^{n+1} \left(w_j(k) + \eta[z(k) - y(k)]x_j(k) - w_j^*\right)^2$$

Desarrollando,

$$= \sum_{j=1}^{n+1} \left(w_j(k) - w_j^*\right)^2 + \eta^2[z(k) - y(k)]^2 \sum_{j=1}^{n+1} x_j(k)^2 +$$

$$+ 2\eta[z(k) - y(k)] \sum_{j=1}^{n+1} \left(w_j(k) - w_j^*\right) x_j(k)$$

(5)

$$= \sum_{j=1}^{n+1} \left(w_j(k) - w_j^*\right)^2 + \eta^2[z(k) - y(k)]^2 \sum_{j=1}^{n+1} x_j(k)^2 +$$

$$+ 2\eta[z(k) - y(k)](\sum_{j=1}^{n+1} w_j(k)x_j(k)) - 2\eta[z(k) - y(k)] \sum_{j=1}^{n+1} w_j^* x_j(k)$$

Obsérvese que $[z(k) - y(k)] \sum_{j=1}^{n+1} w_j(k)x_j(k) < 0$, puesto que si $\sum_{j=1}^{n+1} w_j(k)x_j(k) > 0$ entonces *y(k)*=1 y como la salida es incorrecta tiene que ser *z(k)*=–1, y si $\sum_{j=1}^{n+1} w_j(k)x_j(k) < 0$ entonces *y(k)*=–1 y como la salida es incorrecta, *z(k)*=1. Dicho término se puede escribir de la forma:

$$-2\left|\sum_{j=1}^{n+1} w_j(k)x_j(k)\right|$$

Asimismo, el término

$$[z(k) - y(k)] \sum_{j=1}^{n+1} w_j^* x_j(k) > 0$$

es positivo puesto que si $\sum_{j=1}^{n+1} w_j^* x_j(k) > 0$ entonces la salida deseada es z*(k)*=1 y la salida incorrecta de la red tiene que ser *y(k)*=–1, y si $\sum_{j=1}^{n+1} w_j^* x_j(k) < 0$ entonces *z(k)*=–1 y la

salida incorrecta de la red tiene que ser *y(k)*=1. Así, también se puede escribir de la forma:

$$2\left|\sum_{j=1}^{n+1} w_j^* x_j(k)\right|$$

Por lo tanto tenemos que

$$\sum_{j=1}^{n+1}\left(w_j(k+1)-w_j^*\right)^2 \le \sum_{j=1}^{n+1}\left(w_j(k)-w_j^*\right)^2 + 4\eta^2\sum_{j=1}^{n+1} x_j(k)^2 - 4\eta\left|\sum_{j=1}^{n+1} w_j^* x_j(k)\right|$$

puesto que hemos prescindido de un término negativo en la derecha de la expresión.

Sea

$$D(k+1)=\sum_{j=1}^{n+1}\left(w_j(k+1)-w_j^*\right)^2$$

$$D(k)=\sum_{j=1}^{n+1}\left(w_j(k)-w_j^*\right)^2$$

$$L=\max_{1\le k\le p}\left\{\sum_{j=1}^{n+1} x_j(k)^2\right\}$$

$$T=\min_{1\le k\le p}\left\{\sum_{j=1}^{n+1} w_j^* x_j(k)\right\}$$

entonces la expresión anterior se reduce a

$$D(k+1)\le D(k)+4\eta^2 L-4\eta T$$

$$D(k+1)\le D(k)+4\eta\left[\eta L-T\right]$$

Si tomamos $\eta L-T<0$, es decir,

$$\eta<\frac{T}{L}$$

entonces *D(k+1)<D(k)*. Esto significa que eligiendo un valor de η tal que $0<\eta<\frac{T}{L}$ hace que *D(k)* disminuya al menos en la cantidad constante $\eta(\eta L-2T)$ en cada iteración con corrección. Si el número de iteraciones con corrección fuese infinito entonces llegaríamos al absurdo de alcanzar en una momento determinado un valor negativo del término D(k) que evidentemente es no negativo. ■

La siguiente cuestión que vamos a abordar es estudiar cuál sería el valor mejor que debemos elegir del parámetro de aprendizaje η de forma que se consiga más rápidamente la convergencia de la red. Se trata de elegir η de manera que *D(k+1)* sea lo menor posible y así conseguir un mayor acercamiento de los pesos de la red a la solución. Como *D*(*k*+1) es una función cuadrática del parámetro η solo tenemos que derivar e igualar a cero para encontrar el valor de dicho parámetro que corresponde al mínimo de la expresión *D(k*+1).

$$E(\eta) = D(k+1) = D(k) + 4\eta^2 \sum_{j=1}^{n+1} x_j(k)^2 - 4\eta \left| \sum_{j=1}^{n+1} w_j(k) x_j(k) \right| - 4\eta \left| \sum_{j=1}^{n+1} w_j^* x_j(k) \right|$$

$$\frac{\partial E(\eta)}{\partial \eta} = 8\eta \sum_{j=1}^{n+1} x_j(k)^2 - 4 \left| \sum_{j=1}^{n+1} w_j(k) x_j(k) \right| - 4 \left| \sum_{j=1}^{n+1} w_j^* x_j(k) \right| = 0$$

El valor de η que verifica dicha ecuación es

$$\eta_{opt} = \frac{\left| \sum_{j=1}^{n+1} w_j(k) x_j(k) \right| + \left| \sum_{j=1}^{n+1} w_j^* x_j(k) \right|}{2 \sum_{j=1}^{n+1} x_j(k)^2}$$

Sin embargo el segundo término del numerador no lo conocemos puesto que no conocemos los valores w_j^*. Si aproximamos dicho término por el anterior tenemos que un valor aproximado de la tasa de aprendizaje óptima es el siguiente:

$$\tilde{\eta}_{opt} = \frac{\left| \sum_{j=1}^{n+1} w_j(k) x_j(k) \right|}{\sum_{j=1}^{n+1} x_j(k)^2}$$

Como hemos visto anteriormente,

$$-2 \left| \sum_{j=1}^{n+1} w_j(k) x_j(k) \right| = [z(k) - y(k)] \sum_{j=1}^{n+1} w_j(k) x_j(k)$$

y así

$$\tilde{\eta}_{opt} = \frac{-[z(k) - y(k)] \sum_{j=1}^{n+1} w_j(k) x_j(k)}{2 \sum_{j=1}^{n+1} x_j(k)^2}$$

Sustituyendo este valor del parámetro η en la regla de aprendizaje (cuando *y(k)≠z(k)*) obtenemos,

$$w_j(k+1) = w_j(k) - \frac{[z(k) - y(k)] \sum_{j=1}^{n+1} w_j(k) x_j(k)}{2 \sum_{j=1}^{n+1} x_j(k)^2} [z(k) - y(k)] x_j(k)$$

es decir,

$$w_j(k+1) = w_j(k) - 2\frac{\sum_{j=1}^{n+1} w_j(k)x_j(k)}{\sum_{j=1}^{n+1} x_j(k)^2} x_j(k) \qquad (6)$$

puesto que $[z(k)-y(k)]^2 = 4$ cuando $y(k) \neq z(k)$.

Dicha regla se conoce con el nombre de **regla del Perceptrón normalizada**, pues si partimos de un vector de pesos normalizado, es decir, $\|\mathbf{w}(k)\| = 1$, entonces todos los pesos que se van obteniendo según la regla de aprendizaje se mantienen normalizados. En efecto,

$$\begin{aligned}
\|\mathbf{w}(k+1)\|^2 &= \sum_{j=1}^{n+1} w_j(k+1)^2 \\
&= \sum_{j=1}^{n+1} w_j(k)^2 + \sum_{j=1}^{n+1} x_j(k)^2 \left(2\frac{\sum_{j=1}^{n+1} w_j(k)x_j(k)}{\sum_{j=1}^{n+1} x_j(k)^2} \right)^2 - 4\frac{\sum_{j=1}^{n+1} w_j(k)x_j(k)}{\sum_{j=1}^{n+1} x_j(k)^2} \sum_{i=1}^{n+1} w_j(k)x_j(k) \\
&= \sum_{j=1}^{n+1} w_j(k)^2 = 1
\end{aligned}$$