

Introducción a las redes neuronales: Perceptrón

Mtro. Víctor Mijangos

Facultad de Ingeniería

13-17 de enero, 2020



Clasificación

La regresión lineal establece una función $f : \mathbb{R}^d \rightarrow \mathbb{R}$ (estimar un valor a partir de un vector de entrada).

Otro tipo de problemas del aprendizaje requieren de clasificación. En el caso más simple, tenemos una **clasificación binaria**:

$$f : \mathbb{R}^d \rightarrow \{0, 1\}$$

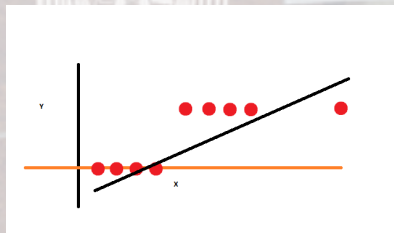
Dado un vector (un objeto caracterizado por d variables) buscamos saber si pertenece a una clase (1) o no (0).



Clasificación

Al igual que en la regresión lineal, se puede tomar el conjunto supervisado y visualizarlo:

$$\mathcal{S} = \{(x, y) : \mathbb{R}^d, y \in \{0, 1\}\}$$



Sin embargo, una línea (o hiperplano) no se ajustarán adecuadamente a los datos.



Clasificación

Necesitamos una función que se ajuste a los datos. Notamos:

- ▶ Existen dos clases. Por tanto, f debe tomar valores 0 ó 1, únicamente.
- ▶ Se espera que exista b tal que:

$$f(x) = \begin{cases} 1 & \text{si } f(x) > b \\ 0 & \text{si } f(x) \leq b \end{cases}$$

Si esta última condición se cumple, se dirá que los datos son **separables**. Cuando f es lineal se dice que son **linealmente separables**.



¿Qué es el perceptrón?

CORNELL AERONAUTICAL LABORATORY, INC.

Report No. 85-460-1

THE PERCEPTRON

A PERCEIVING AND RECOGNIZING AUTOMATON

(PROJECT PARA)

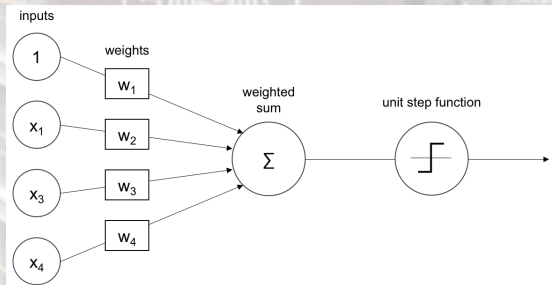
January, 1957

- ▶ El perceptrón es un método para solucionar el problema de la clasificación binaria.
- ▶ La idea básica del perceptrón fue planteada por McCulloch y Pitts [2] y después establecida algorítmicamente por Frank Rosenbluth [3].
- ▶ La idea general del perceptrón es emular el funcionamiento de una neurona.



Perceptrón

- ▶ El perceptrón puede verse como el nodo de una gráfica dirigida que recibe como entrada un conjunto de rasgos (estímulos).
- ▶ Estos rasgos se codifican como un vector en \mathbb{R}^d .
- ▶ Los rasgos son ponderados a través de pesos.
- ▶ Los valores de los rasgos ponderados se suman.
- ▶ La neurona se activa o no a partir de una función de activación.



Formalización del perceptrón

El perceptrón se puede ver como una función $f : \mathbb{R}^d \rightarrow \{0, 1\}$ dada por:

$$f(x) = \begin{cases} 1 & \text{si } \sum_{i=1}^d w_i x_i + b > 0 \\ 0 & \text{si } \sum_{i=1}^d w_i x_i + b \leq 0 \end{cases} \quad (1)$$

donde $w \in \mathbb{R}^d$ es un vector de pesos, $b \in \mathbb{R}$ es un bias o sesgo.

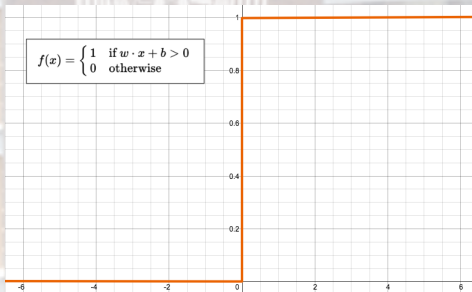
El sesgo b puede verse como un **umbral** que se debe superar para que se active la neurona.



Perceptrón y clasificación binaria

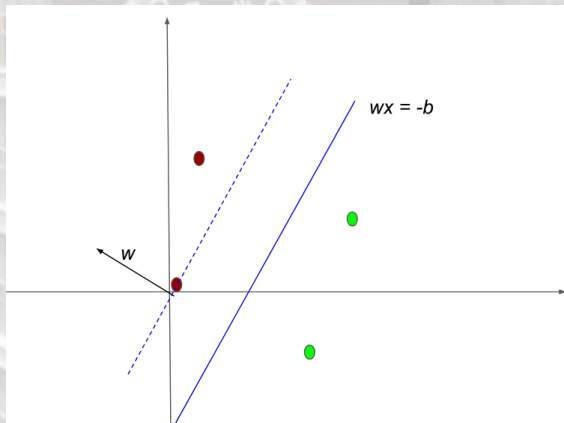
El perceptrón es capaz de clasificar datos en dos clases cuando estos datos son linealmente separables.

El perceptrón tiene semejanzas con la regresión lineal, pero, en tanto clasificación, puede visualizarse como una función escalonada.



Representación geométrica del perceptrón

Otra forma de visualizar el perceptrón es a partir de una línea (o hiperplano) que produce una separación lineal en el espacio de los datos.



Perceptrón: problemas lógicos

El perceptrón es capaz de resolver problemas lógicos del siguiente tipo:

x_1	x_2	OR	AND	NOT (x_1)
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0



Perceptrón: problemas lógicos

La solución del problema OR es

$$w = (1 \ 1), b = -0,5$$

La solución del problema AND es:

$$w = (1 \ 1), b = -1,5$$

La solución del problema NOT (x_1) es:

$$w = (-1 \ 0), b = 0,5$$

Sin embargo, surge la pregunta ¿cómo determinar estos parámetros para cualquier clasificación binaria?



Aprendizaje en el perceptrón

Podemos observar cómo se comporta el perceptrón cuando comete un error. Supóngase w como antes, y b igual a 0.5. En el problema OR, tenemos:

x_1	x_2	$w\mathbf{x} + \mathbf{b}$	$f(\mathbf{x})$	$f(\mathbf{x}) - y$
0	0	0.5	1	1
0	1	1.5	1	0
1	0	1.5	1	0
1	1	2.5	1	0

La diferencia entre $f(\mathbf{x})$ y el valor esperado y es 0 cuando acierta y 1 cuando se equivoca. Si restamos esta diferencia obtendremos el valor de b que buscamos ($-0,5$).



Algoritmo del perceptrón: intuición

La función que define el perceptrón **no es derivable**. Por tanto, no podemos aplicar una derivación para encontrar el error mínimo.

Geométricamente, se establece un ángulo entre w y x :

$$\cos(\angle w, x) \propto w^T x$$

Y podemos interpretar la función del perceptrón como:

$$f(x) = \begin{cases} 1 & \text{si } \angle w, x < \frac{\pi}{2} \\ 0 & \text{si } \angle w, x > \frac{\pi}{2} \end{cases}$$



Algoritmo del perceptrón: intuición

Si el perceptrón comete un error, tenemos dos casos:

- ▶ Asigno $f(x) = 0$, cuando debió asignar $y = 1$ (reducir ángulo o aumentar el coseno):

$$\begin{aligned}\cos(\angle w, x) + x^T x &= w^T x + x^T x \\ &= (w + x)^T x \\ &= (w - (0 - 1)x)^T x\end{aligned}$$

- ▶ Asigno $f(x) = 1$, cuando debió asignar $y = 0$ (aumentar ángulo o reducir el coseno):

$$\begin{aligned}\cos(\angle w, x) - x^T x &= w^T x - x^T x \\ &= (w - x)^T x \\ &= (w - (1 - 0)x)^T x\end{aligned}$$

Por tanto, se actualiza por la regla $w \leftarrow w - (f(x) - y)x$



Algoritmo del perceptrón

Dado un conjunto de datos $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ el objetivo del algoritmo del perceptrón es encontrar los parámetros $w \in \mathbb{R}^d$ y $b \in \mathbb{R}$.

Deben observarse el comportamiento de los errores:

- ▶ Si la neurona debió activarse, pero no lo hizo ($f(x) > 0$), entonces los valores de w deben ascender.
- ▶ Si no debió activarse, pero lo hizo ($f(x) \leq 0$), entonces los valores de w deben descender.

La variación de los pesos debe de variar con respecto a la regla:

$$\Delta w_i = -\eta[f(x^{(k)}) - y^{(k)}]x_i^{(k)} \quad (2)$$

$y^{(k)}$ es la activación esperada para el ejemplo $x^{(k)}$.

η se conoce como el rango de aprendizaje (qué tanto varían los ángulos en cada paso).



Algoritmo del perceptrón

Para el bias, se toma $x^{(k)} = (x_1 \ \cdots \ x_d \ 1)^T$ el Podemos resumir el algoritmo del perceptrón como sigue:

Inicialización: Se escoge aleatoriamente un vector $w = (w_1 \ \cdots \ w_d)^T \in \mathbb{R}^d$

Entrenamiento: Por cada iteración, y para todo $x^{(k)}$ hacer:

1. Calcular

$$f(x) = \begin{cases} 1 & \text{si } w \cdot x^{(k)} > 0 \\ 0 & \text{si } w \cdot x^{(k)} \leq 0 \end{cases}$$

Actualizar los pesos como:

$$w_i \leftarrow w_i - \eta[f(x^{(k)}) - y^{(k)}]x_i^{(k)}$$

Finalización El algoritmo termina cuando el error es 0 o después de T iteraciones.



Ejemplo de aplicación

Supóngase que se tienen los siguientes datos:

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} \quad Y = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Para la inicialización, tomaremos los valores $w = (1 \ 1)^T$ y $b = 1$.



Ejemplo de aplicación

Una primera aplicación muestra que:

$$f(X) = (1 \ 1 \ 1 \ 1)^T$$

Aplicando la función de error, tenemos, por ejemplo que ($\eta = 1$):

$$\begin{aligned} w_1 &\leftarrow w_1 - \eta \sum_{k=1}^4 [f(x^{(k)}) - y^{(k)}] x_1^{(k)} \\ &\leftarrow 1 - [(1 - 1)1 + 0 + 0 + (1 - 1)1] = 1 \end{aligned}$$

$$\begin{aligned} w_2 &\leftarrow w_1 - \eta \sum_{k=1}^4 [f(x^{(k)}) - y^{(k)}] x_2^{(k)} \\ &\leftarrow 1 - [0 + (1 - 0)1 + 0 + (1 - 1)1] = 1 - 1 = 0 \end{aligned}$$



Ejemplo de aplicación

Resumiendo el algoritmo, obtenemos:

Iteración 1 $w = (1 \ 0)^T, b = -1/f(X) = (0 \ 0 \ 0 \ 0)^T$

Iteración 2 $w = (3 \ 1)^T, b = 1/f(X) = (1 \ 1 \ 1 \ 1)^T$

Iteración 3 $w = (3 \ 0)^T, b = -1/f(X) = (1 \ 0 \ 0 \ 1)^T$



Complejidad y convergencia

Proposición

La complejidad del algoritmo es $O(TN)$, donde N es el número de datos y T el número de iteraciones.

Si bien se trata de un algoritmo sencillo, no siempre llegará a converger.

Para que el algoritmo converga, los datos X deben ser **linealmente separables**.

Proposición

Sean X_0 y X_1 dos conjuntos con clases 0 y 1, respectivamente. Estos son linealmente separables si y sólo si son convexos. Esto es si:

$$\forall x, y \in X_i, \lambda x + (1 - \lambda)y \in X_i$$

con $\lambda \in [0, 1]$.



Complejidad y convergencia

De esto, podemos decir que:

Proposición

Sean X_0 y X_1 dos conjuntos de datos con clases 0 y 1, respectivamente. El perceptrón es capaz de clasificar los datos con una solución w^ si y sólo si X_0 y X_1 son convexos.*

Más aún, podemos calcular el tiempo en que el algoritmo del perceptrón converge:

Teorema (Novikoff, 1962)

Supóngase que los datos son convexos por clases. Si para todo x , $\|x\| \leq \rho$ y existe w^ que separa los datos tal que cumple:*

$$\frac{|x \cdot w^*|}{\|w^*\|} \geq \gamma$$

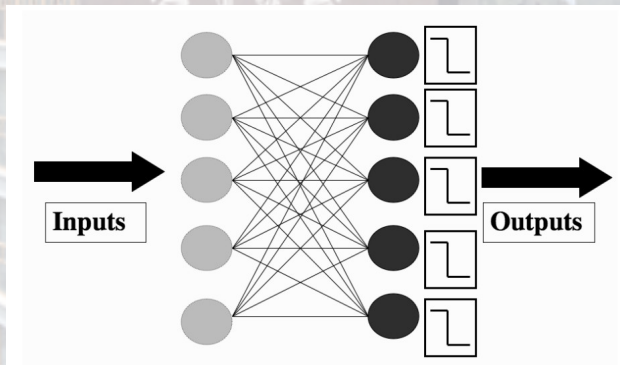
Entonces, el número de iteraciones T para que el algoritmo del perceptrón converja cumple:

$$T \leq \frac{\rho}{\gamma^2}$$



Generalizaciones del perceptrón

La salida del perceptrón puede ser un vector $f(x) \in \{0, 1\}^m$, entonces, los parámetros son $w \in \mathbb{R}^{d \times m}$ y $b \in \mathbb{R}^m$:

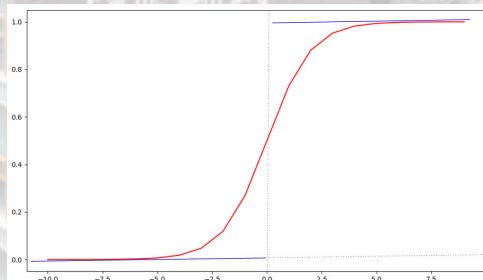


Generalizaciones del perceptrón

Las funciones indicadores ($f : X \rightarrow \{0, 1\}$) pueden ser aproximada por una función $\sigma : X \rightarrow [0, 1]$, llamada sigmoide y definida como:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \quad (3)$$

En una red neuronal a es la preactivación $a = w \cdot x + b$.



Generalizaciones del perceptrón

- ▶ La función sigmoide así definida puede verse como una función de probabilidad, donde $p(X = 1) = \sigma(a)$ y $p(X = 0) = 1 - \sigma(a)$. Por lo que el perceptrón puede verse como un método de inferencia ($X \sim \text{Ber}[\sigma(a)]$).
- ▶ Si la función indicadora toma los valores -1 y 1 , se puede utilizar la aproximación:

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$



El perceptrón como función lógica

El perceptrón surge como un algoritmo para solucionar problemas lógicos [2].

Por tanto, es capaz de solucionar problemas **AND**, **OR** o **NOT**.

Pero no puede resolver un problema de tipo **XOR**.

x_1	x_2	OR	AND	NOT (x_1)	XOR
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	0	1
1	1	1	1	0	0



Problema XOR

Las funciones AND, OR y NOT son **básicas**, en tanto toda función lógica puede expresarse con éstas.




Así, el problema XOR puede expresarse por medio de estas tres funciones [1]:

$$AND(x_1, x_2) = AND[NOT(AND(x_1, x_2)), OR(x_1, x_2)]$$

Esto implica que el problema XOR puede solucionarse por medio de **composición** de funciones lógicas (o perceptrones).



References

-  **Francesco Cicala.**
Perceptrons, logical functions, and the xor problem.
<https://towardsdatascience.com/perceptrons-logical-functions-and-the-xor-problem-37ca5025790a>.
Accessed: 2019-01-05.
-  **Warren S McCulloch and Walter Pitts.**
A logical calculus of the ideas immanent in nervous activity.
The bulletin of mathematical biophysics, 5(4):115–133, 1943.
-  **Frank Rosenblatt.**
The perceptron, a perceiving and recognizing automaton Project Para.
Cornell Aeronautical Laboratory, 1957.



The End

