Capítulo 3

Perceptrón

"Actualmente, las máquinas resuelven problemas de acuerdo con los principios que hemos establecidos en ellas. Dentro de poco quizá aprendamos a ponerlas a trabajar en problemas específicos de mejorar su capacidad de resolver problemas. Una vez traspasado cierto umbral, esto nos llevaría a una espiral de aceleración y sería difícil perfeccionar un regulador fiable para refrenarlo"
Marvin Minsky

3.1. Sinapsis Neuronal

La **sinapsis neuronal** es el intercambio de información que se da entre las neuronas a través de impulsos eléctricos. Este intercambio se da de una neurona *Presináptica* (la que manda la información) a una neurona *Postsináptica* (la que recibe la información).

Dado que las neuronas no se tocan, se tienen que comunicar por medio de sustancias químicas que son el Sodio (Na), Cloro (Cl) y Potasio (K). La manera de enviarse estas sustancias es por medio de neurotransmisores los cuales no llegan a las neuronas Postsinápticas, sino que solamente sirven como una llave para abrir los canales y así puedan pasar las sustancias químicas. Una vez que llegan estas sustancias (la información) a las neuronas Postsinápticas, lo que hacen es sumarla toda (decimos toda porque pueden llegar de distintas neuronas e inclusive la misma neurona puede tener dentro cierta cantidad de sustancias químicas) y si se llega a rebasar un cierto umbral (un límite) la neurona se activa, esto es, dispara potencial de acción hacia otra neurona convirtiéndose así en una nueva neurona Presináptica.

Nota: Recomendamos al lector poner atención a las palabras subrayadas ya que en estas nos basaremos para construir el modelo de la **neurona artificial**.

Las principales partes de las neuronas biológicas (ver Figura 3.1) son: las *dendritas*, el *núcleo* y el *axón*. A continuación, describiremos la función de cada una de ellas en el proceso de la sinapsis:

- Dendritas: son las encargadas de recibir la información que entra a la neurona Postsináptica por medio de las neuronas Presinápticas.
- Núcleo o cuerpo celular: es donde se suma la información que llega de las dendritas y se determina si se rebasa o no el umbral.
- Axón: es por donde se transmite la información ya procesada hacia las ramificaciones terminales que fungirán como nuevas dendritas para las neuronas vecinas mediante el axón.

Luego, el proceso de la sinapsis lo podemos resumir en los siguientes pasos:

1. *Entrada*: será cuando llegue la información de las neuronas Presinápticas, por medio de las dendritas, a las neuronas Postsinápticas. Esta información se pasa al núcleo de la neurona.
2. *Activación*: es el proceso que se da dentro del núcleo de la neurona en el cual se hace la suma de toda la información, y si esa información supera cierto umbral entonces la neurona se activa.
3. *Salida*: es la salida de la información ya procesada a las neuronas vecinas.

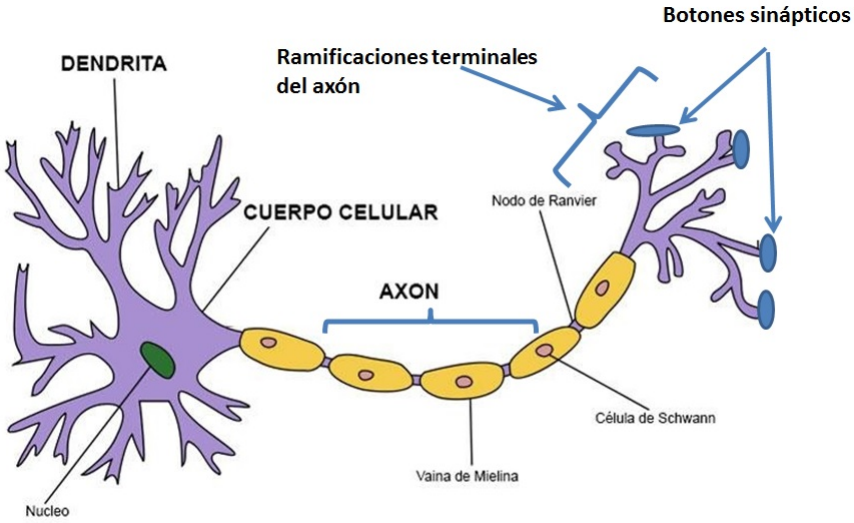


Figura 3.1: Ilustración de una neurona biológica.

3.2. Neurona Artificial

Como vimos anteriormente el proceso de la sinapsis neuronal lo podemos dividir en tres pasos: *entrada*, *activación* y *salida*. A continuación describiremos cómo se da la construcción del modelo de la **neurona artificial** a partir de estos pasos.

3.2.1. Entrada

El proceso de entrada es cuando las *dendritas* reciben información de las neuronas vecinas, por lo que si suponemos que una neurona recibe información de un número n de neuronas (es decir, se comunica con n neuronas vecinas), entonces esta entrada de información la

podemos pensar como un vector en $\mathbb{R}^n$.

Con esto ya tenemos que la entrada de información a la **neurona artificial** es un vector $\mathbf{x} \in \mathbb{R}^n$ al cual daremos el nombre de **vector de entrada**. Entonces el vector de entrada $\mathbf{x}$ se representa de la forma

$$\mathbf{x} = (x_1, x_2, \ldots, x_n).$$

Sin embargo, para que nuestro modelo de la neurona artificial sea más cercano al comportamiento de la neurona biológica, hay que tener en cuenta que cada entrada va a tener una cierta importancia: esto por ejemplo se puede dar en el caso cuando una de las neuronas *presinápticas* se encuentre muy alejada de una neurona *postsináptica*, lo que puede producir que la señal que envíe sea más débil que la de las que se encuentran más cercanas.

Luego, para indicar la importancia que tiene cada entrada del vector $\mathbf{x}$, le asociamos un *peso* a cada una de ellas. Este peso recibe el nombre de **peso sináptico**. Puesto que el vector de entrada es n dimensional, entonces el **vector de pesos sinápticos** también ha de serlo.

Por lo tanto, $\mathbf{w} \in \mathbb{R}^n$ i.e.

$$\mathbf{w} = (w_1, w_2, \ldots, w_n),$$

donde w_i es el peso de la entrada x_i sobre la neurona.

3.2.2. Activación

La activación de la neurona biológica se da dentro del *soma* o también llamado *cuerpo celular*, y como se mencionó más arriba, este proceso consiste en sumar toda la información que llega a la neurona por medio de las dendritas.

Para efectuar esta suma haremos uso del *producto interno* en $\mathbb{R}^n$.

Definición 2. *(Producto Interno). Sea V un conjunto y $\mathbb{F}$ un campo. Un* ***producto interno*** *en V es una función*

$$<,>: V \times V \longrightarrow \mathbb{F}$$

tal que $\forall u, v, w \in V$ y $\lambda \in \mathbb{F}$ cumple las siguientes propiedades:

1. $< u, u > \geq 0$ *y* $< u, u >= 0$ *si, y sólo si* $u = \mathbf{0}$.
2. $< u, v >= \overline{< v, u >}$.
3. $< \lambda u, v >= \lambda < u, v >$.
4. $< u + v, w >=< u, w > + < v, w >$.

Proposición 1. *Sean* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. *Luego, la función* $<,>: \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ *definida como*

$$< \mathbf{x}, \mathbf{y} >= \sum_{i=1}^{n} x_i y_i$$

es un producto interno en $\mathbb{R}^n$.

Demostración
Sean $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ y $\lambda \in \mathbb{R}$.

1. Primero probemos que $< \mathbf{x}, \mathbf{x} >\geq 0$.

$$< \mathbf{x}, \mathbf{x} >= \sum_{i=1}^{n} x_i^2 \geq 0$$

La última desigualdad se obtiene puesto que la suma de números no negativos es un número no negativo, esto es, mayor o igual a cero.

Si $< \mathbf{x}, \mathbf{x} >= 0$, entonces

$$\sum_{i=1}^{n} x_i^2 = 0 \quad \Leftrightarrow \quad x_i^2 = 0, \forall i \quad \Leftrightarrow \quad x_i = 0, \forall i.$$

Por lo tanto, $\mathbf{x} = \mathbf{0}$.

Si $\mathbf{x} = \mathbf{0}$, entonces

$$< \mathbf{x}, \mathbf{x} >= \sum_{i=1}^{n} (0)(0) = 0.$$

2. Recordando que si $u \in \mathbb{R}$ entonces $u = \overline{u}$. Luego,

$$< \mathbf{x}, \mathbf{y} >= \sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{n} y_i x_i =< \mathbf{y}, \mathbf{x} >= \overline{< \mathbf{y}, \mathbf{x} >}.$$

3.

$$< \lambda\mathbf{x}, \mathbf{y} >= \sum_{i=1}^{n} (\lambda x_i) y_i = \lambda \left(\sum_{i=1}^{n} x_i y_i \right) = \lambda < \mathbf{x}, \mathbf{y} > .$$

4.

$$\begin{aligned} < \mathbf{x} + \mathbf{y}, \mathbf{z} >&= \sum_{i=1}^{n} (x_i + y_i) z_i = \sum_{i=1}^{n} x_i z_i + \sum_{i=1}^{n} y_i z_i \\ &=< \mathbf{x}, \mathbf{z} > + < \mathbf{y}, \mathbf{z} > . \end{aligned}$$

Por lo tanto, $<,>$ define un producto interno en $\mathbb{R}$ ■

Nota: El producto interno anterior también es conocido con el nombre de producto punto, ya que existe otra notación como se muestra a continuación:

$$< \mathbf{x}, \mathbf{y} >= \mathbf{x} \cdot \mathbf{y}.$$

En este libro haremos uso indistinto de las dos notaciones.

Con esto la suma de la información del **vector de entrada x** es de la forma

$$< \mathbf{x}, \mathbf{w} >= \sum_{i=1}^{n} x_i w_i$$

ya que así sumaremos todas las entradas con sus respectivos pesos.

Para determinar que una neurona se active, decíamos que entonces la suma de la información que le llegue debe de superar cierto **umbral**[1]. Dicho umbral será un número real al cual denotaremos con el símbolo θ, y para determinar si se supera o no éste valor, veremos si la siguiente expresión

$$< \mathbf{x}, \mathbf{w} > -\theta$$

es no negativa, ya que, si no es negativa esto significa que superó o fue igual que el valor del umbral θ. A manera de analogía piense que cuando va a la tienda a comprar productos y lleva un *capital*, el decir que le alcanzó con su *capital* para todo lo que quería comprar es equivalente a decir que el $capital - gasto \geq 0$ porque el *gasto* no superó al capital, mientras que si $capital - gasto < 0$ significaría que como el *gasto* superó al capital entonces no le alcanzó para comprar todo lo que deseaba.

Puesto de manera explícita, decimos que la suma de la información:

- Rebasó al umbral si $< \mathbf{x}, \mathbf{y} > -\theta > 0$. Se activa la neurona.
- Fue igual a la del umbral si $< \mathbf{x}, \mathbf{y} > -\theta = 0$. Se activa la neurona.
- No rebasó al umbral si $< \mathbf{x}, \mathbf{y} > -\theta < 0$. No se activa la neurona.

Las **funciones de activación** es un tema en el cual nos adentraremos más adelante, por ahora basta mencionar que existen varias funciones y la elección va a depender del problema en cuestión.

3.2.3. Salida

La salida de la neurona es la información ya procesada, es decir, la respuesta a la activación de la neurona. Sin embargo, como la **función de activación** depende del tipo de *problema* que se esté abordando, entonces también será la misma situación para la salida.

Los problemas más comunes son:

1. **Problemas de Clasificación**: aquellos donde se tiene que clasificar (catalogar). Ejemplo: dado el historial crediticio de un cliente en un banco clasificarlo como alguien susceptible de recibir o no un crédito.
2. **Problemas de Regresión**: aquellos donde se pretende pronosticar. Ejemplo: predicción de un índice bursátil con base en datos históricos.

[1] La palabra viene del inglés **threshold** y en algunas bibliografías se traduce también como *sesgo*.

Nota: en ambos problemas se necesita el uso de una base de datos que sirva de *entrenamiento* ya que, como en mencionó en la **Introducción**, este libro está enfocado a modelos de redes neuronales con *aprendizaje supervisado*.

En los *problemas de clasificación* la salida de la neurona artificial es un elemento del conjunto de posibles clasificaciones, siendo

$$Y = \{0, 1\},$$

el conjunto de salida más sencillo porque sólo determina si la neurona se activa o no. Este conjunto recuerda al *sistema binario* ya que tenemos el caso de *Verdadero* o *Falso* (1 o 0) y, por lo tanto, a este problema se le conoce como el problema de la **clasificación binaria**.

En lo que resta del capítulo veremos únicamente el problema de la clasificación binaria.

3.2.4. Definiendo la Neurona Artificial

A continuación definimos la neurona artificial con base en todo lo comentado anteriormente.

Definición 3. *Sean los conjuntos $X \subseteq \mathbb{R}^n$, $Y \subseteq \mathbb{R}$ y la función*

$$\sigma : \mathbb{R} \longrightarrow Y.$$

*Definimos a la **neurona artificial** como la tercia (X, Y, σ), donde X es el **conjunto de entrada**, Y el **conjunto de salida** y σ la **función de activación** de dicha neurona.*

Analicemos la definición:

- La **neurona artificial** se define como una tercia (X, Y, σ) debido a que cada neurona se compone de un **conjunto de entrada**, un **conjunto de salida** y una **función de activación**.
- $\mathbf{x} \in X$ pero $\mathbf{w} \in \mathbb{R}^n$. Esto significa que si bien el **vector de entrada** tiene que pertenecer al conjunto X, el **vector de pesos** no necesariamente. Esto se debe a que el vector $\mathbf{w}$ le asocia un número real a cada entrada del vector $\mathbf{x}$, por lo que no necesariamente se tendrá que $\mathbf{w} \in X$.

  La importancia de ésta aclaración se entenderá más adelante debido a que en el proceso de entrenamiento el conjunto de entrada no cambia mientras que el vector de pesos sí.
- La **función de activación** va de $\mathbb{R}$ al conjunto de salida Y ya que la suma de la información es un número real.
- El **conjunto de salida** es un subconjunto de los números reales porque aunque estemos clasificando otros objetos que no sean números, siempre podemos hallar una asociación con ellos. Por ejemplo, si estamos en un restaurante y queremos que las clasificaciones sean *bueno*, *regular* y *malo*, podemos asociar al número 1 con *bueno*, al número 2 con *regular* y al número 3 con *malo* (en realidad hacemos uso de la notación

one hot encoding, pero esto se hablará en la sección del **Perceptrón Monocapa** del capítulo 7). Para el caso de los problemas de regresión es aun más sencillo ya que los pronósticos suelen ser números.

3.3. Construyendo el Perceptrón

Así como el metro es la unidad básica para medir la distancia en el Sistema Internacional de Medida, el **perceptrón** es la unidad básica en las RNA. El **perceptrón** fue creado por Frank Rosenblatt en 1956 y se le conoce también como la neurona de McCulloch-Pitts en honor a Warren McCulloch y Walter Pitts, quienes con su artículo *A logical calculus of the ideas immanent in nervous activity* publicado en 1943, plantearon la posibilidad de crear las neuronas artificiales.

A continuación vamos a construir el **perceptrón** con dos caracterizaciones las cuales son equivalentes, pero es importante conocer ambas ya que en ciertas ocasiones nos convendrá ocupar más una que la otra.

3.3.1. Primera Caracterización

Si bien es cierto que la definición de neurona artificial contempla todos los puntos importantes del modelo del pase de información entre las neuronas biológicas, aun falta una descripción formal de dicho proceso. Esta descripción vendrá dada involucrando el *producto punto* del **vector de entrada** con el **vector de pesos** para la suma de la información, posteriormente la resta de esta información con el **umbral** y, por último, la aplicación de la **función de activación** a esta diferencia. Todo este proceso de *tranferencia* de la información en una neurona artificial es lo que se conoce como **perceptrón**.

A manera de conveniencia, en una neurona artificial (X, Y, σ) vamos a definir a la función $z : X \times \mathbb{R}^n \longrightarrow \mathbb{R}$ como

$$z = < \mathbf{x}, \mathbf{w} > -\theta$$

donde $<,>$ es el *producto interior* en $\mathbb{R}^n$ anteriormente planteado, $\mathbf{x}$ un vector de entrada, $\mathbf{w}$ un vector de pesos y θ el umbral de la neurona. Esta función lleva el nombre de **preactivación** porque es la operación que hace la neurona artificial antes de aplicar la función de activación.

Definición 4. *Sea (X, Y, σ) una neurona artificial. Si $\mathbf{x}, \mathbf{w} \in X$ son los vectores de entrada y pesos de la neurona, respectivamente, y $\theta \in \mathbb{R}$ el umbral entonces la función $f : X \longrightarrow Y$ definida por*

$$f = \sigma \circ z,$$

*se le conoce con el nombre de **perceptrón** o **neurona de McCulloch-Pitts**.*

En la Figura 3.2 se puede apreciar de manera gráfica la construcción del **perceptrón**. Este tipo de gráficas son muy comunes en la literatura debido a que explican de una manera sencilla el comportamiento de las neuronas artificiales.

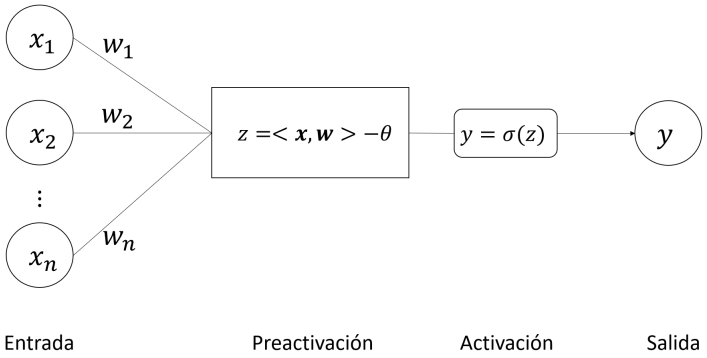


Figura 3.2: Esquema de la **primera caracterización del perceptrón**.

Note que el **perceptrón** lo definimos mediante una función f, la cual toma valores del conjunto de entrada X y devuelve valores del conjunto de salida Y. Esta función f recibe el nombre de **función de transferencia** porque transfiere la información de las neuronas Presinápticas a las Postsinápticas (hay que tener en cuenta que hasta este punto únicamente hemos hecho el modelo de una neurona artificial, pero más adelante hablaremos de las RNA).

3.3.2. Segunda caracterización

En esta segunda caracterización vamos a realizar una pequeña modificación en el *vector de entrada* y en el de *pesos*, esto con el fin de simplificar el proceso de la *preactivación*.

El **perceptrón** hasta hora definido tiene como vectores de entrada y pesos vectores n-dimensionales, es decir, $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$. Definamos $\hat{\mathbf{x}}, \hat{\mathbf{w}} \in \mathbb{R}^{n+1}$ de la siguiente manera:

$$\hat{\mathbf{x}} := (\mathbf{x}, 1) \quad \text{y} \quad \hat{\mathbf{w}} := (\mathbf{w}, -\theta),$$

donde θ es el *umbral* del **perceptrón** en cuestión.

Note que la preactivación ahora tiene un pequeño cambio, ya que

$$z =< \mathbf{x}, \mathbf{w} > -\theta = \sum_{i=1}^{n} x_i w_i - \theta =< \hat{\mathbf{x}}, \hat{\mathbf{w}} > .$$

Es decir, lo que hacemos con este cambio es simplificar la escritura de la preactivación del **perceptrón** debido a que al aumentar una dimensión en los vectores de *entrada* y *pesos* podemos introducir, de manera automática, el valor de *umbral* por medio del *producto punto*.

Vea en la figura 3.3 la diferencia en la construcción del **perceptrón** con la segunda caracterización; la entrada x_{n+1} del vector $\hat{\mathbf{x}}$ toma el valor de 1, y la entrada w_{n+1} del vector $\hat{\mathbf{w}}$ toma el valor del $-\theta$, por lo que al multiplicar la última entrada de ambos vectores nos da por resultado el valor de $-\theta$. Puesto en pocas palabras: añadimos una nueva entrada al vector de entrada con el valor constante de 1 y peso respectivo $-\theta$, y con esto restamos el

valor del umbral de manera automática.

Este tipo de caracterización sobre todo nos servirá para la parte *computacional* porque así nos ahorramos crear la variable *umbral*.

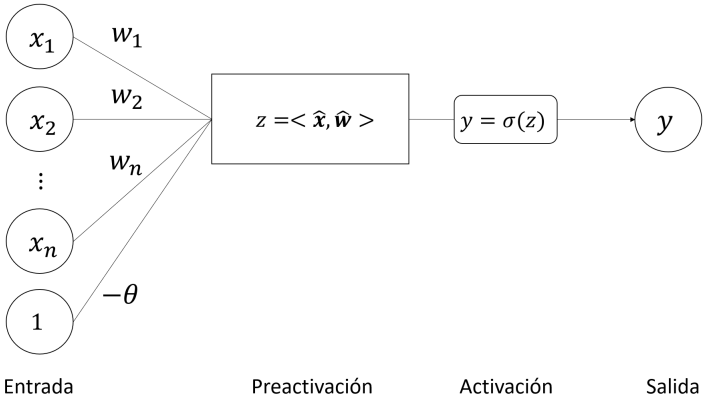


Figura 3.3: Esquema de la **segunda caracterización del perceptrón**.

3.4. Aprendizaje del Perceptrón

El perceptrón que hemos construido modela de forma adecuada la transferencia de información en una neurona biológica, sin embargo, cuenta con un problema: no hemos modelado el proceso de aprendizaje. Este apartado abordará el tema de aprendizaje del perceptrón y lo vamos a dividir en cuatro puntos:

1. Daremos respuesta a la siguiente pregunta: ¿qué significa que un perceptrón aprenda?
2. Contruiremos el modelo del perceptrón para resolver **puertas lógicas**. Esto nos ayudará a entender cuándo un perceptrón clasifica bien y cuándo clasifica mal.
3. Explicaremos el significado matemático de la clasificación por medio del perceptrón. Además veremos la relación del perceptrón con el modelo de la **RLS** y **RLM**.
4. Estudiaremos el **Algoritmo de Aprendizaje del Perceptrón** (AAP) que son los pasos que debe de seguir el perceptrón para que se realice el proceso de aprendizaje de forma automática, y enunciaremos el **Teorema de Convergencia del Perceptrón** que nos garantizará conseguirlo en un tiempo finito.

3.4.1. ¿Qué significa que un perceptrón aprenda?

Primero mencionaremos una metáfora que se suele usar para explicar en qué consiste el aprendizaje supervisado, la cual nos da la pauta para entender el proceso de aprendizaje del perceptrón respecto a sus componentes.

Metáfora del maestro y el estudiante
Supongamos que un maestro experto en el área de la Olimpiada de Matemáticas está

instruyendo a un estudiante interesado en esta área. Para lograr su propósito, el maestro le entrega al estudiante una cantidad n de ejercicios variados. El estudiante al terminar los ejercicios se los entrega al maestro el cual, como es un experto, sabe las respuestas de todos los ejercicios y, por lo tanto, le dice cuáles respuestas tuvo correctas y cuáles incorrectas. Con base en esto, el estudiante procede de la siguiente forma para su estudio: los temas de los ejercicios que tuvo incorrectos les dará más importancia que los temas de los ejercicios que tuvo correctos. Luego del estudio, el estudiante le pide nuevamente los n-ejercicios al maestro y procede como al inicio: se le entregan los ejercicios al estudiante, una vez resueltos son revisados por el maestro el cual señala al estudiante los ejercicios correctos y los incorrectos, por último, el estudiante estudia los temas de los ejercicios dando más importancia a los que tuvo incorrectos. Siguiendo este proceso de **entrenamiento** el estudiante llegará a ser capaz de resolver no solamente los ejercicios que el profesor le entregue, sino, cualquier ejercicio similar a ellos.

En la *metáfora del maestro y el estudiante*, el modelo del perceptrón juega el papel del alumno y las clasificaciones el de las respuestas del maestro (correcto e incorrecto). Note que el proceso de entrenamiento del alumno (perceptrón) es darle más importancia (vector de pesos) a los temas (vector de entrada) en los cuales tiene ejercicios incorrectos, lo que significa que el proceso de aprendizaje del perceptrón es modificar el vector de pesos[2] con base en las clasificaciones que tenemos del conjunto de datos del entrenamiento.

Retomando el ejemplo de la clasificación de imágenes de perros y gatos (ver en **Introducción**), supongamos que el vector de entrada contiene la información de tres características del animal en cuestión; tiene la forma de los ojos como primera entrada, de las orejas como segunda y la de los dientes como tercera. El proceso de entrenamiento será ver cuáles de las tres características tienen más importancia para determinar si la imagen dada es un perro o un gato, de tal suerte que, si la forma de los dientes es más importante que la de los ojos, entonces el peso de la entrada tres será mayor que el de la entrada uno[3]. Puesto en términos matemáticos, si $\mathbf{x} = (x_1, x_2, x_3)$ donde x_1 mide la forma de los ojos, x_2 la de las orejas y x_3 la de los dientes, entonces,

$$w_3 > w_1.$$

Nota: en la desigualdad anterior suponemos que tanto x_1 como x_3 se ponderan de la misma manera.

Finalmente: ¿qué significa que un perceptrón aprenda?

El *proceso de aprendizaje del perceptrón* consiste en modificar el vector pesos con base en las clasificaciones dadas por los datos, o dicho con las palabras de la metáfora, por medio de las correciones del maestro.

[2]Si estamos hablando de la **primera caracterización del perceptrón** entonces es modificar al vector de pesos y al umbral; si es el caso de la **segunda caracterización del perceptrón**, entonces solamente es modificar el vector de pesos debido a que el vector de pesos contempla en la última entrada al umbral.

[3]Este ejemplo sólo es a manera de ilustración. Lo que se aplica para la identificación de imágenes son redes más específicas como las *Redes Convolucionales*.

3.4.2. Puertas Lógicas

Las *puertas lógicas* son el ejemplo por excelencia para explicar la clasificación binaria puesto que, dado un conjunto de entradas, sólo devuelven una salida de tipo *booleano*[4]: 0 o 1.

Como la salida de la neurona debe de pertenecer al conjunto $Y = \{0, 1\}$, entonces la función de activación que tomaremos como referencia será la **función de heaviside** o también llamada la *función salto unitario*.

Definición 5. *La función* $H : \mathbb{R} \longrightarrow \{0, 1\}$ *se define como*

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

Con la función H de heaviside y la definición del perceptrón dadas, en teoría ya somos capaces de modelar el comportamiento de las *puertas lógicas*. Daremos como ejemplos las puertas lógicas AND y OR. Sin embargo, veremos que la puerta XOR presenta un problema.

Puerta AND

Las puertas lógicas se pueden definir por medio de varios enfoques, como por ejemplo las *tablas de verdad* o el álgebra de boole. Nosotros daremos la definición por medio de funciones cuyo dominio será el conjunto $\{0, 1\}^2$ y el rango el conjunto $\{0, 1\}$, esto con el fin de simplificar la notación, pero daremos también la tabla de verdad respectiva para hacer notar la equivalencia.

Definición 6. *Sea* $\mathbf{x} \in \{0, 1\}^2$. *Definimos la puerta lógica* AND *como:*

$$AND(\mathbf{x}) = \begin{cases} 1, & si\ x_1 = x_2 = 1 \\ 0, & en\ otro\ caso. \end{cases}$$

La tabla de verdad respectiva es:

x_1	x_2	$AND(x_1, x_2)$
1	1	1
1	0	0
0	1	0
0	0	0

Solución
Sean $\mathbf{w} \in \mathbb{R}^2$ y $\theta \in \mathbb{R}$.

[4]En honor a Georg Boole quien en su libro *Las leyes del pensamiento* (1854) construyó un sistema algebráico para la lógica. Ochenta y cuatro años después, Claude Shannon retoma el *álgebra de boole* para crear, por medio de transistores, circuitos eléctricos que dan como resultado las primeras computadoras.

Tomemos primero $\mathbf{x} = (1,1)$. Usando la primera caracterización del perceptrón, la preactivación z resulta:

$$z =< \mathbf{x}, \mathbf{w} > -\theta =< (1,1), (w_1, w_2) > -\theta = 1 \cdot w_1 + 1 \cdot w_2 - \theta = w_1 + w_2 - \theta.$$

Por definición tenemos que la salida debe de ser

$$AND(\mathbf{x}) = 1,$$

y por la activación del perceptrón (ver figura 3.4)

$$AND(\mathbf{x}) = \sigma(z) = H(z).$$

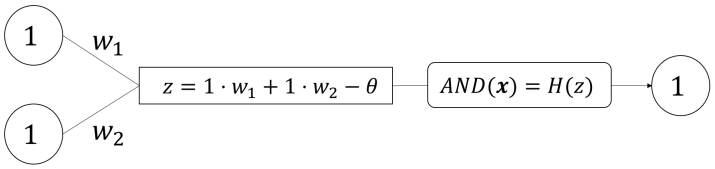


Figura 3.4: Esquema del perceptrón con vector de entrada $\mathbf{x} = (1,1)$.

Luego,

$$H(w_1 + w_2 - \theta) = 1.$$

De manera similar tenemos

$$AND(1,0) = H((w_1)(1) + (w_2)(0) - \theta) = 0,$$

$$AND(0,1) = H((w_1)(0) + (w_2)(1) - \theta) = 0,$$

$$AND(0,0) = H((w_1)(0) + (w_2)(0) - \theta) = 0.$$

Y por lo tanto resulta el siguiente sistema de ecuaciones

$$\begin{aligned} H(w_1 + w_2 - \theta) &= 1, \\ H(w_1 - \theta) &= 0, \\ H(w_2 - \theta) &= 0, \\ H(-\theta) &= 0. \end{aligned}$$

Por la función de heaviside obtenemos el siguiente sistema de desigualdades:

$$\begin{aligned} w_1 + w_2 \geq& \theta, \\ w_1 <& \theta, \\ w_2 <& \theta, \\ -\theta <& 0. \end{aligned}$$

Con lo que concluimos que cualquier vector $\mathbf{w}$ y cualquier escalar θ que cumplan las desigualdades anteriores sirven para modelar el comportamiento de la puerta AND por medio de un perceptrón.

Si por ejemplo, tomamos el vector $\mathbf{w} = \left(\frac{1}{2}, \frac{1}{2}\right)$ y el escalar $\theta = 0.8$, entonces

$$\frac{x_1}{2} + \frac{x_2}{2} = 0.8$$

es un perceptrón que modela de manera correcta el comportamiento de la puerta AND (ver figura 3.5, de color café), porque las desigualdades

$$\begin{aligned} 1 = \frac{1}{2} + \frac{1}{2} &\geq 0.8, \\ \frac{1}{2} &< 0.8, \\ \frac{1}{2} &< 0.8, \\ -0.8 &< 0, \end{aligned}$$

son verdaderas. Si tomamos $\mathbf{w} = (1, 1)$ y $\theta = 1.5$, entonces

$$x_1 + x_2 = 1.5$$

es un perceptrón que modela de manera correcta el comportamiento de la puerta AND (ver figura 3.5, de color verde), ya que las desigualdades

$$\begin{aligned} 2 = 1 + 1 &\geq 1.5, \\ 1 &< 1.5, \\ 1 &< 1.5, \\ -1.5 &< 0, \end{aligned}$$

son verdaderas.

Luego, encontramos una relación que nos da un vector de pesos basándonos en los datos de los resultados de la puerta AND, es decir, construimos un perceptrón que clasifica de manera correcta las entradas de la puerta AND.

Puerta OR

Procedamos de manera similar para la puerta OR.

Definición 7. *Sea* $\mathbf{x} \in \{0, 1\}^2$. *Definimos la puerta lógica OR como:*

$$OR(\mathbf{x}) = \begin{cases} 0, & \text{si } x_1 = x_2 = 0 \\ 1, & \text{en otro caso.} \end{cases}$$

La tabla de verdad respectiva es:

x_1	x_2	$OR(x_1, x_2)$
1	1	1
1	0	1
0	1	1
0	0	0

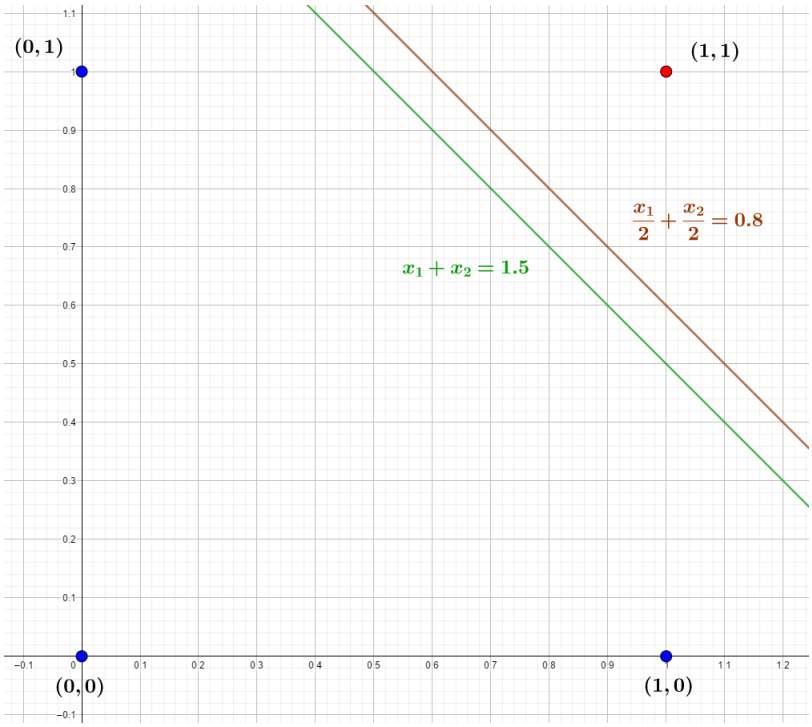


Figura 3.5: Puerta lógica *AND*. Los puntos azules son aquellos cuya salida tiene el valor de 0 y los rojos el valor de 1. Con esto vemos cómo los perceptrones separan de manera adecuada unos puntos de otros, es decir, hacen una correcta clasificación.

Solución
Sean $\mathbf{w} \in \mathbb{R}^2$ y $\theta \in \mathbb{R}$.

Luego,

$$\begin{aligned} OR(1,1) &= H((w_1)(1) + (w_2)(1) - \theta) = 1, \\ OR(1,0) &= H((w_1)(1) + (w_2)(0) - \theta) = 1, \\ OR(0,1) &= H((w_1)(0) + (w_2)(1) - \theta) = 1, \\ OR(0,0) &= H((w_1)(0) + (w_2)(0) - \theta) = 0. \end{aligned}$$

Lo que implica

$$\begin{aligned} H(w_1 + w_2 - \theta) &= 1, \\ H(w_1 - \theta) &= 1, \\ H(w_2 - \theta) &= 1, \\ H(-\theta) &= 0. \end{aligned}$$

Y por lo tanto

$$\begin{aligned} w_1 + w_2 \geq& \theta, \\ w_1 \geq& \theta, \\ w_2 \geq& \theta, \\ -\theta <& 0. \end{aligned}$$

Con lo cual tomando $\mathbf{w} = (1,1)$ y $\theta = 0.7$, resulta que

$$x_1 + x_2 = 0.7$$

es un perceptrón que modela el comportamiento de la puerta OR (ver figura 3.6, color verde) porque

$$\begin{aligned} 2 = 1 + 1 &\geq 0.7, \\ 1 &\geq 0.7, \\ 1 &\geq 0.7, \\ -0.7 &< 0. \end{aligned}$$

Tomando $\mathbf{w} = (1,2)$ y $\theta = 0.9$, resulta que

$$x_1 + 2x_2 = 0.9$$

es un perceptrón que modela el comportamiento de la puerta OR (ver figura 3.6, color naranja) porque

$$\begin{aligned} 3 = 1 + 2 &\geq 0.9, \\ 1 &\geq 0.9, \\ 2 &\geq 0.9, \\ -0.9 &< 0. \end{aligned}$$

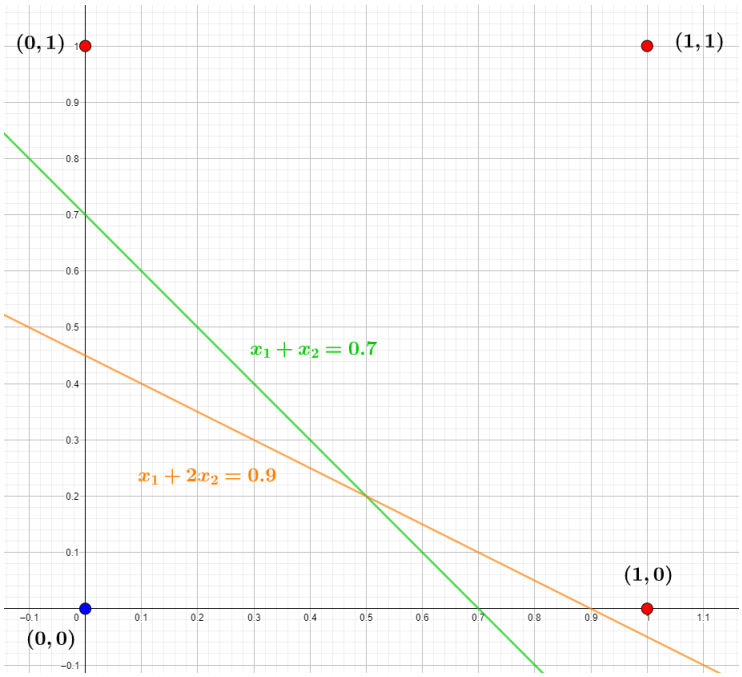


Figura 3.6: Puerta lógica OR. Los puntos azules son aquellos cuya salida tiene el valor de 0 y los rojos el valor de 1.

Los ejemplos de la puerta AND y OR nos muestran cómo el perceptrón puede simular su comportamiento por medio de una *línea* que separa los puntos etiquetados con el valor 1 de los puntos etiquetados con el valor de 0, y ante esto se podría conjeturar que todo perceptrón puede funcionar como un buen modelo para todas las puertas lógicas. Sin embargo vamos a analizar lo que pasa con la puerta XOR.

Puerta XOR

La puerta *XOR* se define como a continuación.

Definición 8. *Sea* $\mathbf{x} \in \{0,1\}^2$. *Definimos la puerta XOR como:*

$$XOR(\mathbf{x}) = \begin{cases} 1, & si\ x_1 \neq x_2 \\ 0, & en\ otro\ caso \end{cases}$$

La tabla de verdad respectiva es:

x_1	x_2	$XOR(x_1, x_2)$
1	1	0
1	0	1
0	1	1
0	0	0

Esta puerta lógica es un caso que no puede ser modelado de manera correcta por medio de un perceptrón, y esto lo enunciamos en la siguiente proposición.

Proposición 2. *No existe ningún perceptrón que pueda modelar a la puerta lógica XOR.*

Demostración
Supongamos que existe un perceptrón con vector de pesos $\mathbf{w} \in \mathbb{R}^2$ y con umbral $\theta \in \mathbb{R}$ que puede modelar a la puerta lógica *XOR*. Luego,

$$\begin{aligned} XOR(1,1) &= H((w_1)(1) + (w_2)(1) - \theta) = 0, \\ XOR(1,0) &= H((w_1)(1) + (w_2)(0) - \theta) = 1, \\ XOR(0,1) &= H((w_1)(0) + (w_2)(1) - \theta) = 1, \\ XOR(0,0) &= H((w_1)(0) + (w_2)(0) - \theta) = 0. \end{aligned}$$

Lo que implica

$$\begin{aligned} H(w_1 + w_2 - \theta) &= 0, \\ H(w_1 - \theta) &= 1, \\ H(w_2 - \theta) &= 1, \\ H(-\theta) &= 0. \end{aligned}$$

Y por lo tanto

$$\begin{aligned} w_1 + w_2 &< \theta, \\ w_1 &\geq \theta, \\ w_2 &\geq \theta, \\ -\theta &< 0. \end{aligned}$$

Este último sistema de desigualdades es equivalente (sumar la primera desigualdad con la última y la segunda con la tercera) a las desigualdades

$$w_1 + w_2 < 2\theta \quad \text{y} \quad w_1 + w_2 \geq 2\theta,$$

lo cual es una contradicción.

La contradicción viene de suponer que existe tal perceptrón y, por lo tanto, no existe ningún perceptrón que pueda modelar la puerta lógica XOR■

Con la demostración anterior aseguramos que no existe un perceptrón que pueda modelar la puerta XOR, lo que es equivalente a decir que no existe un línea que pueda separar los puntos cuya salida es 1 de los puntos cuya salida es 0. Te invitamos a que antes de pasar a la siguiente sección intentes encontrar una línea recta que separe los puntos rojos de los azules en la figura 3.7.

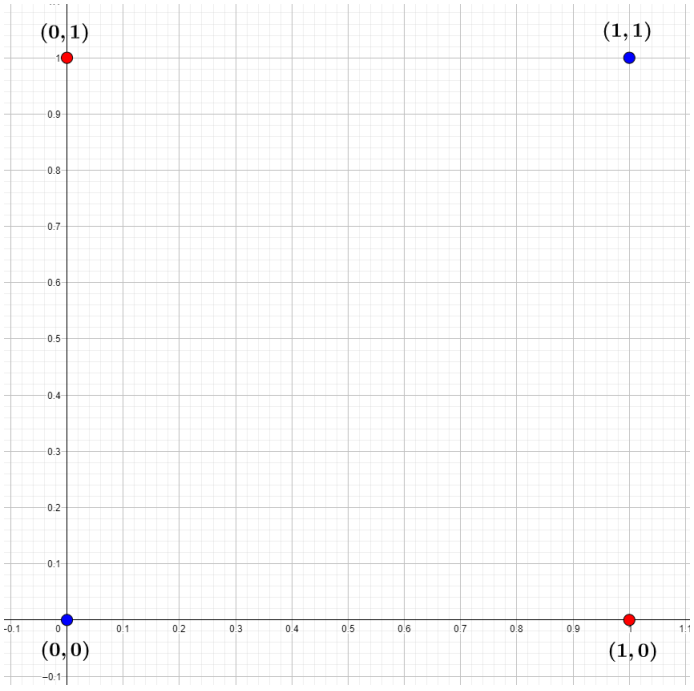


Figura 3.7: Puerta lógica XOR. No existe ninguna línea recta que separe los puntos rojos de los azules.

3.4.3. Clasificación del perceptrón

En la sección anterior vimos que el **perceptrón** puede simular de manera correcta el comportamiento de las puertas lógicas AND y OR, lo que significa que puede clasificar las entradas con respecto de sus etiquetas de la misma forma que lo hacen las funciones AND y OR. También vimos el caso de la puerta XOR en el cual no podía realizar dicha clasificación y dimos la demostración: de manera geométrica decíamos que se debía a que no existía una línea recta que pudiese separar los puntos etiquetados con el número 1 de

los que estaban etiquetados con el número 0.

Lo anterior nos da pauta a concluir que la representación gráfica del perceptrón, en el problema de la clasificación binaria, es una línea recta que separa unos puntos de otros con base a sus clasificaciones. Entonces, decir que un perceptrón puede simular el comportamiento de una puerta lógica es equivalente a decir que *existe al menos una línea recta que separe a unos puntos con una clasificación respecto de los puntos con la otra clasificación*, o dicho en pocas palabras, *que las clasificaciones son linealmente separables*.

Pero ¿qué pasaría si el conjunto de entrada no fuera subconjunto de $\mathbb{R}^2$? Por ejemplo, observe la figura 3.8 y suponga que se quieren separar los puntos rojos de los azules, en este caso podemos apreciar que una línea recta no bastaría porque los puntos ahora pertenecen a $\mathbb{R}^3$ (son puntos en el espacio), pero si en vez de usar una línea recta usáramos un *plano* (vea figura 3.9) entonces lograríamos la clasificación binaria de manera exitosa. La idea de un *plano*[5] que separa puntos en dimensiones superiores (de $\mathbb{R}^4$ en adelante) la tenemos en el concepto **hiperplano afín** que es nuestra siguiente definición.

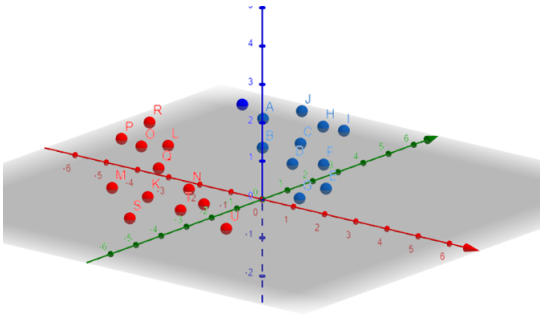


Figura 3.8: Puntos en el espacio.

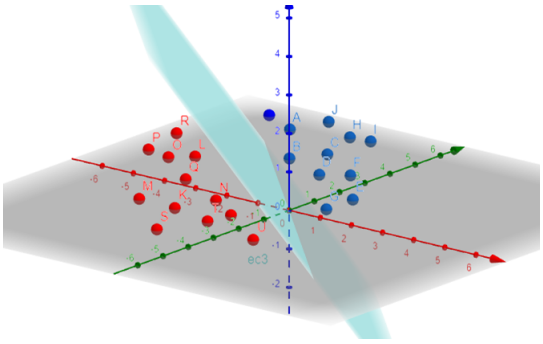


Figura 3.9: Los puntos azules han sido separados de los puntos rojos mediante el plano azul claro.

[5]Los planos de $\mathbb{R}^4, \mathbb{R}^5, \ldots, \mathbb{R}^n$ se les da el nombre de hiperplanos en $\mathbb{R}^4, \mathbb{R}^5, \ldots, \mathbb{R}^n$, respectivamente.

Definición 9. *Un **hiperplano afín** Hp de $\mathbb{R}^n$ puede expresarse de la forma*

$$Hp = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \cdot \mathbf{u} = c\}$$

*donde $\mathbf{u} \neq 0$ (unitario) es el vector normal a Hp y $c \in \mathbb{R}$. Definimos también a los dos **semiespacios cerrados** determinados por Hp como*

$$Hp^+ = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \cdot \mathbf{u} \geq c\}$$

$$Hp^- = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \cdot \mathbf{u} \leq c\}$$

Entonces un **hiperplano afín** en $\mathbb{R}^n$ será aquel hiperplano en $\mathbb{R}^n$ que pueda separar dos conjuntos de puntos Hp^+ y Hp^-; esto significa que el hiperplano Hp divide a $\mathbb{R}^n$ en dos semiespacios, donde en cada semiespacio se encontrará un conjunto de puntos. Con esto ya podemos dar una definición de que dos conjuntos sean separables, es decir, suceptibles de clasificación binaria.

Definición 10. *Sean $A, B \subseteq \mathbb{R}^n$. A es **separable afín** de B si existe un **hiperplano afín** que los separe, esto es si existe un vector $(\mathbf{w}, \theta) \in \mathbb{R}^{n+1}$ tal que*

$$\mathbf{w} \cdot \mathbf{x} - \theta = \begin{cases} \geq 0 & si\ \mathbf{x} \in A \\ < 0 & si\ \mathbf{x} \in B. \end{cases}$$

*Diremos que A es **separable afín de forma estricta** de B si*

$$\mathbf{w} \cdot \mathbf{x} - \theta = \begin{cases} > 0 & si\ \mathbf{x} \in A \\ < 0 & si\ \mathbf{x} \in B. \end{cases}$$

De lo anterior se concluye que una función podrá ser modelada por medio de un **perceptrón** si el conjunto es **separable afín**, y viceversa; es decir, si un conjunto es **separable afín** existe una función que puede ser modelada por medio de un **perceptrón**. Este resultado lo enunciamos en el siguiente Teorema.

Teorema 1. *Sea $X \subseteq \mathbb{R}^n$. Una función de transferencia $f : X \longrightarrow \{0, 1\}$ puede ser representada por un **perceptrón** si y sólo si existe un **hiperplano afín** que separe al conjunto $f^{-1}(1)$ del conjunto $f^{-1}(0)$.*

Demostración
f es una función de transferencia que representa a un **perceptrón** si y sólo si existe un vector $(\mathbf{w}, \theta) \in \mathbb{R}^{n+1}$ tal que

$$f(\mathbf{x}) = H(\mathbf{w} \cdot \mathbf{x} - \theta)$$

Lo que por la función de Heaviside se cumple si y sólo si

$$\mathbf{w} \cdot \mathbf{x} - \theta = \begin{cases} \geq 0 & \text{si } \mathbf{x} \in f^{-1}(1) \\ < 0 & \text{si } \mathbf{x} \in f^{-1}(0) \end{cases}$$

Que es cierto si y sólo si existe un **hiperplano afín** que separe al conjunto $f^{-1}(1)$ del conjunto $f^{-1}(0)$■

La importancia de este Teorema radica en que la separación afín nos da un criterio para determinar si un conjunto puede ser modelado por medio de un perceptrón o no en el problema de la clasificación binaria.

Envolvente Convexa

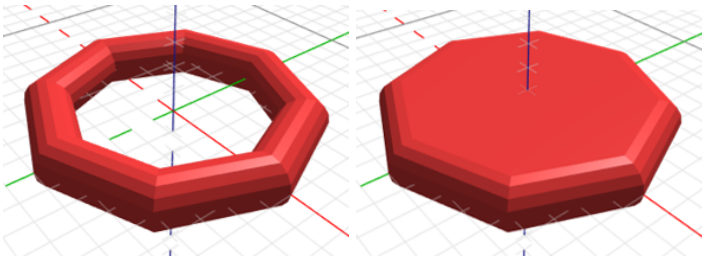
El concepto de **envolvente convexa** de un conjunto nos brinda una manera más sencilla de determinar si un conjunto es separable afín de otro y, por lo tanto, si existe un perceptrón que pueda realizar una correcta clasificación binaria.

Definición 11. *Sea $A \subseteq \mathbb{R}^n$. Se define la **envolvente convexa** de A, y se representa por $convA$, como la intersección de todos los subconjuntos convexos de $\mathbb{R}^n$ que contienen a A.*

Por lo tanto, $convA$ será el conjunto convexo más pequeño de $\mathbb{R}^n$ que contenga a A. Además

$$convA = \left\{ \sum_{i=1}^{k} \lambda_i \mathbf{x}_i : \; k \in \mathbb{N}, \mathbf{x}_i \in A, \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}.$$

A pesar de que la formulación matemática de la **envolvente convexa** de un conjunto sea compleja la idea detrás de ella es bastante simple, ya que se puede pensar como emplayar un conjunto de puntos. Por ejemplo, en la figura 3.10 del lado izquierdo mostramos la representación gráfica de un *toro*[6] en $\mathbb{R}^3$ y si *emplayáramos* con un plástico a este toro obtendríamos algo similar a la figura de la derecha, la cual es su respectiva envolvente convexa.



(a) Toro en $\mathbb{R}^3$ (b) Envolvente convexta del toro.

Figura 3.10: Ejemplo de la envolvente convexa del toro.

Luego, dos conjuntos son separables afín si sus envolventes convexas no coinciden en ningún punto. Este resultado lo enunciaremos como un Corolario y anexaremos su demostración advirtiendo al lector que es un tanto técnica.

[6]Figura geométrica que es una superficie de revolución generada por una circunferencia que gira alrededor de una recta exterior coplanaria.

Corolario 1. *Sean $A, B \in \mathbb{R}^n$ dos conjuntos no vacíos siendo B un conjunto abierto. Entonces, A es **separable afín** de B si y sólo si*

$$conv(A) \cap conv(B) = \emptyset.$$

Demostración
$\Rightarrow$) Supongamos que A es separable afín de B. Entonces existen dos semiespacios

$$Hp^+ = \{x : \mathbf{x} \cdot \mathbf{w} \leq \theta\} \quad \text{y} \quad Hp^- = \{x : \mathbf{x} \cdot \mathbf{w} < \theta\}$$

con $A \subseteq Hp^+$ y $B \subseteq Hp^-$.

Puesto que Hp^+ y Hp^- son convexos entonces se sigue que

$$conv(A) \subseteq Hp^+ \quad \text{y} \quad conv(B) \subseteq Hp^-$$

lo que implica que

$$conv(A) \cap conv(B) = \emptyset \blacksquare$$

$\Leftarrow$) Supongamos que $conv(A) \cap conv(B) = \emptyset$. Puesto que B es abierto entonces $conv(B)$ es abierto, y como supusimos que $conv(A)$ y $conv(B)$ son disjuntos entonces son separables afín.

Debido a que $A \subseteq conv(A)$ y $B \subseteq conv(B)$ entonces A es separable afín de $B \blacksquare$

Con este Corolario ahora resulta mucho más sencillo visualizar que no existe ningún perceptrón capaz de simular el comportamiento de la puerta *XOR* (ver figura 3.11).

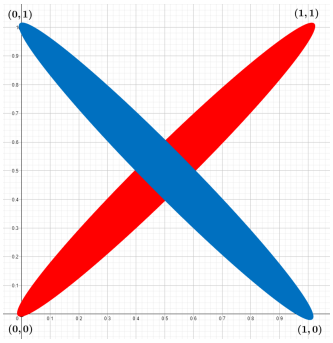


Figura 3.11: La intersección de las envolventes convexas de los conjuntos a clasificar es distinta del vacío.

Relación con la RLS y RLM

El modelo de la **RLS** (ver capítulo de **Regresión Lineal**) consiste en encontrar la línea que mejor se aproxime a describir la tendencia de los datos, de aquí que el modelo del perceptrón sea muy similar en cuanto a formulación, debido a que ambos modelos tienen

como representación geométrica una línea recta. Por ejemplo, más arriba demostramos que el perceptrón

$$x_1 + 2x_2 = 0.9,$$

simula de manera adecuada el comportamiento de la puerta lógica *OR*.

Si a este perceptrón se le compara con la función de *RLS*:

$$r(x) = \beta_0 + \beta_1 x,$$

entonces tendríamos la relación $\beta_0 = \frac{0.9}{2}$ y $\beta_1 = -\frac{1}{2}$, pues

$$x_2 = \frac{0.9}{2} - \frac{1}{2}x_1.$$

De manera similar, la **RLM** tiene como representación geométrica un hiperplano y, por lo tanto, la formulación tiene una equivalencia con el perceptrón con conjunto de entrada en dimensiones superiores.

3.4.4. AAP

El APP es la secuencia de pasos que al seguir nos garantizará una correcta clasificación, es decir, son los pasos que nos permitirán modificar el vector de pesos del perceptrón con el fin de resolver el problema de la clasificación binaria. El que sea un algoritmo nos permite programarlo y que de esta manera sea un proceso de **aprendizaje automático**. Se usa el término *aprendizaje automático* en el sentido de que ya no tendremos que preocuparnos de que el vector de pesos cumpla las desigualdades como en el modelado de las puertas lógicas AND y OR, o que los conjuntos de puntos (según las clasificaciones) sean separables afín ya sea por medio del hiperplano afín o de las envolventes convexas; basta programar el algoritmo para que el perceptrón encuentre el vector de pesos en un número finito de pasos.

Puesto que deseamos modificar el vector de pesos necesitamos una notación para diferenciar los distintos vectores que tendremos. Dicha notación sólo se usará en éste algoritmo y se hará por medio de índices, pues para la programación nos será más útil establecerlo así.

Denotamos al k-ésimo vector de pesos como $\mathbf{w}_k$. Es importante notar la diferenciar entre la notación $\mathbf{w}_k$ y w_k, haciendo la primera referencia al k-ésimo vector de pesos en el proceso del entrenamiento, mientras que la segunda se refiere a la k-ésima entrada de un vector de pesos $\mathbf{w}$. Para ser más precisos tenemos que

$$\mathbf{w}_k = (w_{k_1}, w_{k_2}, \ldots, w_{k_n}, \theta_k).$$

Nota: la variable k es un número entero finito[7] no negativo, es decir, $k \in \mathbb{N} \cup \{0\}$ con $k < \infty$.

Dicho la anterior dejamos el AAP.

[7] Recuerde que el Teorema de Convergencia del Perceptrón nos garantiza conseguir la clasificación en un número finito de pasos.

AAP
Paso 1 . Asignar al vector $\mathbf{w}_0$ entradas de forma aleatoria y $k := 0$.
Paso 2 . Tomar un vector de entrada aleatorio $\mathbf{x}$.
a) Si $\mathbf{x} \in f^{-1}(1)$ y $\mathbf{w}_k \cdot \mathbf{x} > 0$ ir a Paso 2.
b) Si $\mathbf{x} \in f^{-1}(1)$ y $\mathbf{w}_k \cdot \mathbf{x} \leq 0$ ir a Paso 3.
c) Si $\mathbf{x} \in f^{-1}(0)$ y $\mathbf{w}_k \cdot \mathbf{x} < 0$ ir a Paso 2.
d) Si $\mathbf{x} \in f^{-1}(0)$ y $\mathbf{w}_k \cdot \mathbf{x} \geq 0$ ir a Paso 4.
Paso 3 . $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{x}$ e *incrementamos* el valor de k en 1. Ir a Paso 2.
Paso 4 . $\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{x}$ e *incrementamos* el valor de k en 1. Ir a Paso 2.

Veamos los siguientes puntos para analizar a fondo el AAP:

1. $\mathbf{x}, \mathbf{w} \in \mathbb{R}^{n+1}$ pues se está usando la segunda caracterización del perceptrón.

2. $\mathbf{w}_0$ es el primer vector de pesos.

3. La función f puede ser representada por un perceptrón pues es separable afín, esto es, existe un hiperplano afín que separa $f^{-1}(1)$ de $f^{-1}(0)$ (ver **Teorema 1**).

4. En el Paso 2, los incisos a) y c) retornan al Paso 2 porque en ambos casos se hizo una correcta clasificación. Los incisos b) y d) mandan a los Pasos 3 y 4, respectivamente, para crear un nuevo vector de pesos con base en el actual.

5. El índice k indexa a la sucesión de vectores de pesos que se irán creando con el AAP. Dicha sucesión de vectores de pesos recibe el nombre de **sucesión de entrenamiento**. Una sucesión de entrenamiento será adecuada si en el AAP se repite el Paso 2 de manera indefinida (esto de manera teórica, para la parte práctica lea el siguiente punto), ya que esto significa (según el punto anterior) que cada vector de entrada es clasificado de manera correcta.

6. En el Paso 2 se toma un vector aleatorio del conjunto de entrada X, sin embargo, en la programación se le da un orden a los vectores de entrada; es decir, si la cardinalidad del conjunto X es de N entonces,

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}.$$

   Luego, el AAP en el Paso 2 primero tomará al vector $\mathbf{x}_1$, luego al $\mathbf{x}_2$, etc. Si el Paso 2 se repite para todos los vectores del conjunto X (de forma consecutiva), entonces el algoritmo habrá acabado.

La construcción del AAP no la comentamos por el momento debido a que es un caso particular del algoritmo de optimización **Descenso del Gradiente** del que nos ocuparemos más adelante.

Finalmente, dejamos el **Teorema de Convergencia del Perceptrón** que al tener una demostración mucho más técnica preferimos omitirla y dejar en la bibliografía la consulta a la fuente original.

Teorema 2. *Sea f un perceptrón con la segunda caracterización. Si el conjunto de entrada X es separable afín entonces el AAP actualiza el vector de pesos $\mathbf{w}_k$ en un número finito de pasos para un correcta clasificación binaria.*

Ejemplo práctico

Apliquemos el AAP para representar la puerta *OR* por medio de un perceptrón.

Para este ejemplo tenemos que el conjunto de entrada es:

$$X = \{(1,1),(1,0),(0,1),(0,0)\}.$$

Pero, como usamos la segunda caracterización del perceptrón resulta

$$X = \{(1,1,1),(1,0,1),(0,1,1),(0,0,1)\}$$

donde

$$f^{-1}(1) = \{(1,1,1),(1,0,1),(0,1,1)\} \quad \text{y} \quad f^{-1}(0) = \{(0,0,1)\}.$$

Paso 1. Establecemos de manera aleatoria $\mathbf{w}_0$:

$$\mathbf{w}_0 = (1,2,1).$$

Paso 2. Tomamos $\mathbf{x}_1 = (1,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_0 \cdot \mathbf{x}_1 = (1,2,1) \cdot (1,1,1) = 4 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_2$.

Paso 2. Tomamos $\mathbf{x}_2 = (1,0,1) \in f^{-1}(1)$.

$$\mathbf{w}_0 \cdot \mathbf{x}_2 = (1,2,1) \cdot (1,0,1) = 2 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_3$.

Paso 2. Tomamos $\mathbf{x}_3 = (0,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_0 \cdot \mathbf{x}_3 = (1,2,1) \cdot (0,1,1) = 3 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_4$.

Paso 2. Tomamos $\mathbf{x}_4 = (0,0,1) \in f^{-1}(0)$.

$$\mathbf{w}_0 \cdot \mathbf{x}_4 = (1,2,1) \cdot (0,0,1) = 1 > 0.$$

Lo que resulta el inciso d) y, por lo tanto, vamos al Paso 4.

Paso 4. Creamos el vector de pesos $\mathbf{w}_1$:

$$\mathbf{w}_1 = \mathbf{w}_0 - \mathbf{x}_4 = (1,2,1) - (0,0,1) = (1,2,0).$$

El valor de k será $k = 0 + 1 = 1$.

Esto concluye la primera *iteración* pues ya recorrimos todos los vectores de entrada de X. La sucesión de entrenamiento hasta ahora obtenida es:

$$\{(1,2,1),(1,2,0)\}.$$

Vamos al Paso 2.

Paso 2. Tomamos $\mathbf{x}_1 = (1,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_1 \cdot \mathbf{x}_1 = (1,2,0) \cdot (1,1,1) = 3 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_2$.

Paso 2. Tomamos $\mathbf{x}_2 = (1,0,1) \in f^{-1}(1)$.

$$\mathbf{w}_1 \cdot \mathbf{x}_2 = (1,2,0) \cdot (1,0,1) = 1 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_3$.

Paso 2. Tomamos $\mathbf{x}_3 = (0,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_1 \cdot \mathbf{x}_3 = (1,2,0) \cdot (0,1,1) = 2 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_4$.

Paso 2. Tomamos $\mathbf{x}_4 = (0,0,1) \in f^{-1}(0)$.

$$\mathbf{w}_1 \cdot \mathbf{x}_4 = (1,2,0) \cdot (0,0,1) = 0.$$

Lo que resulta el inciso d) y, por lo tanto, vamos al Paso 4.

Paso 4. Creamos el vector de pesos $\mathbf{w}_2$:

$$\mathbf{w}_2 = \mathbf{w}_1 - \mathbf{x}_4 = (1,2,0) - (0,0,1) = (1,2,-1).$$

El valor de k será $k = 1 + 1 = 2$.

La sucesión de entrenamiento es:

$$\{(1,2,1),(1,2,0),(1,2,-1)\}.$$

Vamos al Paso 2.

Paso 2. Tomamos $\mathbf{x}_1 = (1,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_2 \cdot \mathbf{x}_1 = (1,2,-1) \cdot (1,1,1) = 2 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_2$.

Paso 2. Tomamos $\mathbf{x}_2 = (1,0,1) \in f^{-1}(1)$.

$$\mathbf{w}_2 \cdot \mathbf{x}_2 = (1,2,-1) \cdot (1,0,1) = 0.$$

Lo que resulta el inciso b) y, por lo tanto, vamos al Paso 3.

Paso 3. Creamos el vector de pesos $\mathbf{w}_3$:

$$\mathbf{w}_3 = \mathbf{w}_2 + \mathbf{x}_2 = (1,2,-1) + (1,0,1) = (2,2,0).$$

El valor de k será $k = 2 + 1 = 3$.

La sucesión de entrenamiento es:

$$\{(1,2,1),(1,2,0),(1,2,-1),(2,2,0)\}.$$

Vamos al Paso 2 empezando nuevamente con $\mathbf{x}_1$.

Paso 2. Tomamos $\mathbf{x}_1 = (1,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_3 \cdot \mathbf{x}_1 = (2,2,0) \cdot (1,1,1) = 4 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_2$.

Paso 2. Tomamos $\mathbf{x}_2 = (1,0,1) \in f^{-1}(1)$.

$$\mathbf{w}_3 \cdot \mathbf{x}_2 = (2,2,0) \cdot (1,0,1) = 2 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_3$.

Paso 2. Tomamos $\mathbf{x}_3 = (0,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_3 \cdot \mathbf{x}_3 = (2,2,0) \cdot (0,1,1) = 2 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_4$.

Paso 2. Tomamos $\mathbf{x}_4 = (0,0,1) \in f^{-1}(0)$.

$$\mathbf{w}_3 \cdot \mathbf{x}_4 = (2,2,0) \cdot (0,0,1) = 0.$$

Lo que resulta el inciso d) y, por lo tanto, vamos al Paso 4.

Paso 4. Creamos el vector de pesos $\mathbf{w}_4$:

$$\mathbf{w}_4 = \mathbf{w}_3 - \mathbf{x}_4 = (2,2,0) - (0,0,1) = (2,2,-1).$$

El valor de k será $k = 3 + 1 = 4$.

La sucesión de entrenamiento es:

$$\{(1,2,1),(1,2,0),(1,2,-1),(2,2,-1)\}.$$

Vamos al Paso 2.

Paso 2. Tomamos $\mathbf{x}_1 = (1,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_4 \cdot \mathbf{x}_1 = (2,2,-1) \cdot (1,1,1) = 3 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_2$.

Paso 2. Tomamos $\mathbf{x}_2 = (1,0,1) \in f^{-1}(1)$.

$$\mathbf{w}_4 \cdot \mathbf{x}_2 = (2,2,-1) \cdot (1,0,1) = 1 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_3$.

Paso 2. Tomamos $\mathbf{x}_3 = (0,1,1) \in f^{-1}(1)$.

$$\mathbf{w}_4 \cdot \mathbf{x}_3 = (2,2,-1) \cdot (0,1,1) = 1 > 0.$$

Lo que resulta el inciso a) y, por lo tanto, repetimos el Paso 2 pero ahora con $\mathbf{x}_4$.

Paso 2. Tomamos $\mathbf{x}_4 = (0,0,1) \in f^{-1}(0)$.

$$\mathbf{w}_4 \cdot \mathbf{x}_4 = (2,2,-1) \cdot (0,0,1) = -1 < 0.$$

Lo que resulta el inciso c) lo que nos hace repetir el Paso 2 y, por lo tanto, terminamos. Esto es porque el vector de pesos $\mathbf{w} = (2,2,-1)$ clasifica de una manera correcta a todos los elementos del conjunto X.

Entonces el AAP encontró el vector de pesos adecuado con un total de 4 iteraciones y con una sucesión de entrenamiento:

$$\{(1,2,1),(1,2,0),(1,2,-1),(2,2,-1)\}.$$

Por lo tanto, un perceptrón que modela de una manera adecuada el comportamiento de la puerta lógica OR es (ver figura 3.12):

$$2x_1 + 2x_2 = 1.$$

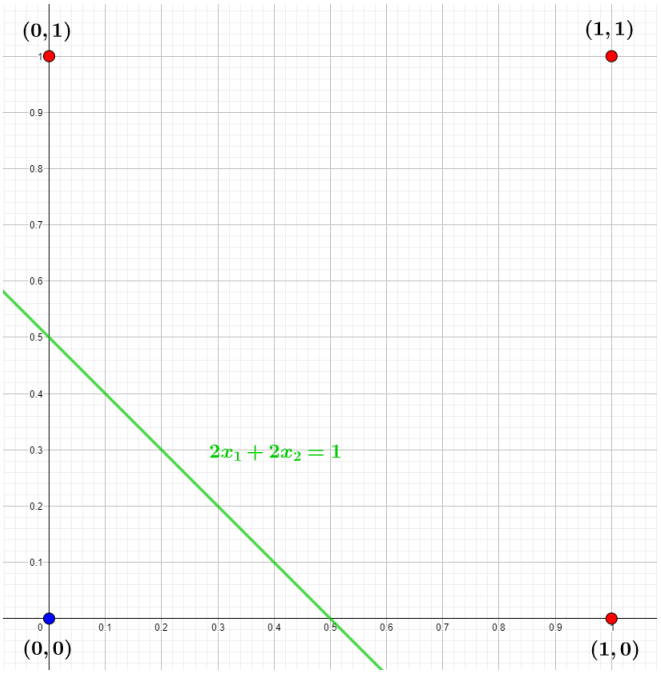


Figura 3.12: Perceptrón $2x_1 + 2x_2 = 1$ obtenido por medio del AAP que modela la puerta lógica *OR*.

3.5. Hiperparámetros

El AAP nos garantiza una correcta clasificación siempre y cuando ésta exista, pero desafortunamente, para otro tipo de neuronas y redes neuronas artificiales no tenemos tal garantía. Por esta razón la mayoría de los algoritmos de DL lo que hacen es hallar una aproximación, por medio de *iteraciones*, para la optimización del problema que se proponen resolver; estas iteraciones reciben el nombre de **épocas**. Por ejemplo, si tenemos un conjunto de entrada X con cardinalidad n,

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\},$$

entonces la primer época del proceso de entrenamiento concluye cuando se termina de aplicar el AAP en la primera iteración, es decir, después de aplicar el AAP a los vectores $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n$ por primera vez.

Las épocas son consideradas un **hiperparámetro** puesto que si bien es un parámetro fundamental para el proceso del aprendizaje del perceptrón (a mayor número de épocas más se aproxima a la solución) no existe una regla o algoritmo específico que nos dé una solución óptima y, por lo tanto, solamente se cuentan con ciertos métodos que se han descubierto por medio de la experimentación.

De igual forma existe otro hiperparámetro llamado la **tasa de aprendizaje**[8] que sirve para modular qué tan rápido queremos que se realice dicho aprendizaje. La letra que representa

[8] También llamado el **ratio de aprendizaje** o **learning rate**.

a la tasa de aprendizaje es la letra griega η y se recomienda que su valor se encuentre entre los valores de 0 y 1, es decir,

$$0 < \eta \leq 1.$$

3.6. AAP computarizado

Como se mencionó anteriormente, a pesar de que el AAP nos garantiza una correcta clasificación para el problema de la clasificación binaria en un número finito de **épocas**, en general no se tiene un algoritmo que funcione de la misma forma para otro tipo de RNA. Por esta razón, vamos a plantear el **AAP computarizado** de tal forma que tenga una misma lógica que otros algoritmos de DL (aproximar la solución por medio de épocas), pero que al mismo tiempo tenga una equivalencia con el AAP.

Nota: el AAP computarizado, al ser equivalente al AAP, rara vez tendrá un impedimento para lograr la clasificación binaria de forma exitosa (salvo cuando el hiperparámetro de la **tasa de aprendizaje** no se ajuste de forma adecuada), porque generalmente se da la convergencia antes de llegar a la última época.

Para describir el AAP computarizado nos basaremos en la *metáfora del maestro y el estudiante* vista en la sección anterior.

El conjunto de entrada X es el conjunto de ejercicios con los que cuenta el maestro, es decir, si el maestro tiene N ejercicios entonces

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}.$$

De aquí tenemos que la primer época del algoritmo es equivalente a que el maestro le da todos los ejercicios, de manera secuencial, por primera vez; es decir, primero $\mathbf{x}_1$, posteriormente $\mathbf{x}_2$, luego $\mathbf{x}_3$ etc. hasta finalmente $\mathbf{x}_N$.

Las respuestas del maestro y el estudiante a los ejercicios se plantearán por medio de las funciones M y E, respectivamente. Dichas funciones son definidas como

$$M, E : X \longrightarrow \{0, 1\}$$

porque dado un ejercicio $\mathbf{x}_i \in X$ éste puede estar bien (1) o mal contestado (0).

Esto nos da ya una equivalencia con el AAP antes planteado:

AAP versión 1

Paso 1 . Asignar al vector $\mathbf{w}_0$ entradas de forma aleatoria, establecemos $k := 0$ y a *contador_epocas* $:= 1$.

Paso 2 . Si *contador_epocas* $<=$ *epocas* entonces establecemos a *contador_ejercicios* $:= 1$ y vamos a Paso 3. En cualquier otro caso termina el algoritmo.

Paso 3 . Si *contador_ejercicios* $<= N$ entonces le asignamos a $\mathbf{x}$ el valor de $\mathbf{x}_{contador\_ejercicios}$. En cualquier otro caso vamos a Paso 2.

a) Si $E(\mathbf{x}) = 1$ y $M(\mathbf{x}) = 0$, entonces $\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{x}$ e incrementamos el valor de k en 1.

b) Si $E(\mathbf{x}) = 0$ y $M(\mathbf{x}) = 1$, entonces $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{x}$ e incrementamos el valor de k en 1.

c) Incrementamos el valor de *contador_ejercicios* en 1 y repetimos Paso 3.

El algoritmo antes planteado a primera instancia se ve más complicado que el AAP, aunque en realidad sea equivalente a éste último sólo que considerando la nueva notación y el hiperparámetro de las épocas, pero la ventaja de utilizarlo es que se puede simplificar al siguiente algoritmo:

AAP versión 2

1. Establecer un vector de pesos aleatorio $\mathbf{w}$.
2. Repetir el Paso 3 tantas veces como épocas haya.
3. Para cada $\mathbf{x}_i \in X$ con $i = 1, 2 \ldots N$, reemplazar el valor de $\mathbf{w}$ por:

$$\mathbf{w} + [M(\mathbf{x}_i) - E(\mathbf{x}_i)]\mathbf{x}_i.$$

A continuación planteamos el AAP mediante pseudocódigo, y para los lectores que no conozcan mucho de programación recomendamos revisar el **Apéndice B** donde se abordan, aunque de manera simplificada, los conceptos clave para entender el algoritmo.

```
Crear w de manera aleatoria.
Para i=1 hasta epocas
    Para j=1 hasta N
        w += lr * [M(x[j])-E(x[j])]x[j]
```

Note que el AAP computarizado tiene una diferencia respecto al algoritmo antes planteado: la multiplicación con *lr*. El término *lr* se refiere al hiperparámetro η pues en inglés tasa de aprendizaje se traduce como *learning rate*, y la razón por lo que ahora sólo lo comentamos es debido a que su explicación vendrá más adelante cuando hablemos del algoritmo **descenso del gradiente**. Por ahora basta darse cuenta que los algoritmos hasta ahora expuestos también venían multiplicados por una tasa de aprendizaje: $\eta = 1$.

3.7. Ejercicios

1. Investiga las siguientes puertas lógicas, defínelas tanto por su tabla de verdad como por medio de una función y da un perceptrón que pueda modelar de manera correcta

su comportamiento.

a) NOT.

b) NOR.

c) $NAND$.

2. Demuestra que no existe ningún perceptrón que puede simular el comportamiento de la puerta lógica $XNOR$.

3. ¿Geométricamente, en qué nos ayuda el umbral θ en el modelo del Perceptrón?

   Tome por ejemplo un conjunto de entrada $X \subset \mathbb{R}^2$ que es separable afín, por lo que existe un vector de pesos $\mathbf{w} = (w_1, w_2, -\theta)$ tal que

   $$x_1 w_1 + x_2 w_2 - \theta = 0,$$

   realiza una clasificación binaria de X.

   Compare con un perceptrón con $\theta = 0$, es decir, de la forma

   $$x_1 w_1 + x_2 w_2 = 0.$$

   Grafique con valores aleatorios ambos perceptrones y note en qué se diferencían uno de otro.

4. (*Programación*) Programar el AAP para la solución de puertas lógicas. Comprobar programa con el ejemplo práctico visto en la sección 3.4.4.

5. Utilizar el AAP para encontrar un perceptrón que separe al conjunto de puntos

   $$A = \{(0.5, 1.5), (0, 2)\},$$

   del conjunto

   $$B = \{(0, 1), (0.5, 0.5), (-0.5, 1)\}.$$

   Si se le da el punto (0.5,2), ¿a cuál conjunto de puntos pertenecerá?