El Perceptrón, Entrenamiento del Perceptrón y Backpropagation

The Perceptron, Perceptron Training, and Backpropagation

Autor: Juan Pablo López Ramírez
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: juanpablo.lopez1@utp.edu.co

***Resumen*— Este documento presenta un resumen acerca del tema del perceptrón, de su entrenamiento y de muchos otros aspectos importantes este, además también vamos a hablar sobre el tema del backpropagation ampliando un poco todo lo que tiene que ver con estos temas de una forma concreta y fácil de entender para todos, haciendo uso de imágenes para una mayor comprensión.**

***Palabras clave*— Perceptrón, entrenamiento , inteligencia artificial, programación, backpropagation, redes neuronales .**

***Abstract*— This document presents a summary about the subject of the perceptron, its training and many other important aspects of this, in addition we are also going to talk about the subject of backpropagation expanding a little everything that has to do with these subjects in a concrete and easy way to understand for everyone, making use of images for greater understanding.**

***Key Word*— Perceptron, training, artificial intelligence, programming, backpropagation, neural networks.**

INTRODUCCION

En este documento vamos a realizar un análisis sobre los temas del perceptrón y el backpropagation, donde trataremos de aprender más sobre estos temas que no son tan conocidos pero que son muy importantes, además de hablar un poco de lo que antecede a estos temas que es el tema de las redes neuronales.

Haciendo uso de los recursos que tenemos a la mano, buscaremos la forma de abordar de la mejor manera estos temas, buscando comprender de buena manera cada uno de los aspectos que hace al perceptrón y al backpropation importantes, además de que esto nos ayuda a profundizar más en el área de la computación y la inteligencia artificial.

1.1 REDES NEURONALES

¿QUE ES UNA RED NEURONAL?

Las redes neuronales son un modelo inspirado en el funcionamiento del cerebro humano. Está formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida [1].

Otra definición seria la siguiente; las redes neuronales son modelos simples del funcionamiento del sistema nervioso. Las unidades básicas son las neuronas, que generalmente se organizan en capas. Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas [2].

¿CUAL ES EL OBJETIVO DE UNA RED NEURONAL?

El objetivo principal de este modelo es aprender modificándose automáticamente a si mismo de forma que puede llegar a realizar tareas complejas que no podrían ser realizadas mediante la clásica programación basada en reglas. De esta forma se pueden automatizar funciones que en un principio solo podrían ser realizadas por personas.

¿COMO FUNCIONAN?

Como se ha mencionado el funcionamiento de las redes se asemeja al del cerebro humano. Las redes reciben una serie de valores de entrada y cada una de estas entradas llega a un nodo llamado neurona. Las neuronas de la red están a su vez agrupadas en capas que forman la red neuronal. Cada una de las neuronas de la red posee a su vez un peso, un valor numérico, con el que modifica la entrada recibida. Los nuevos valores obtenidos salen de las neuronas y continúan su camino por la red. Este funcionamiento puede observarse de forma esquemática en la siguiente imagen.

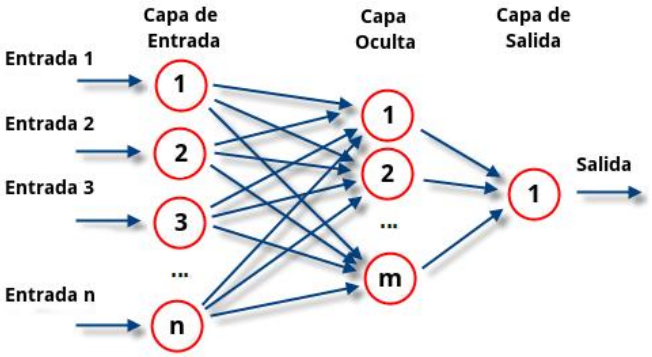

Figura 1. Red Neuronal.

TIPOS DE ENTRENAMIENTO DE UNA RED NEURONAL

Hay tres grandes paradigmas de aprendizaje, cada uno correspondiente a una tarea de aprendizaje abstracto en particular. Estos son:

- El entrenamiento o aprendizaje supervisado.
- El entrenamiento o aprendizaje no supervisado.
- Aprendizaje por refuerzo.

VENTAJAS DE UNA RED NEURONAL

Debido a su constitución y a sus fundamentos, las redes neuronales artificiales presentan un gran número de características semejantes a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia [3]. Esto hace que ofrezcan numerosas ventajas y que este tipo de tecnología se esté aplicando en múltiples áreas. Entre las ventajas se incluyen:

- Aprendizaje Adaptativo. Capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.
- Auto-organización. Una red neuronal puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.
- Tolerancia a fallos. La destrucción parcial de una red conduce a una degradación de su estructura; sin embargo, algunas capacidades de la red se pueden retener, incluso sufriendo un gran daño.
- Operación en tiempo real. Los cómputos neuronales pueden ser realizados en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.

ALGUNOS USOS DE LAS REDES NEURONALES

Estos sistemas de algoritmia que nos ayudan a resolver problemas tienen múltiples aplicaciones [4]. Estas las podemos englobar en :

- Predicción de sucesos y simulaciones: Producción de los valores de salida esperados en función de los datos entrantes.
- Reconocimiento y clasificación: Asociación de patrones y organización de conjuntos de datos en clases predefinidas. Incluso identificando características únicas sin datos previos.
- Procesamiento de datos y modelización: Validación, agregación y análisis de datos. Diseño y búsqueda de fallos en sistemas de software complejos.
- Ingeniería de control: Monitorización de sistemas informáticos y manipulación de robots. Incluida la creación de sistemas y robots autónomos.
- Inteligencia Artificial: Formando parte de las tecnologías de Deep learning y machine learning que son partes fundamentales de la inteligencia artificial.

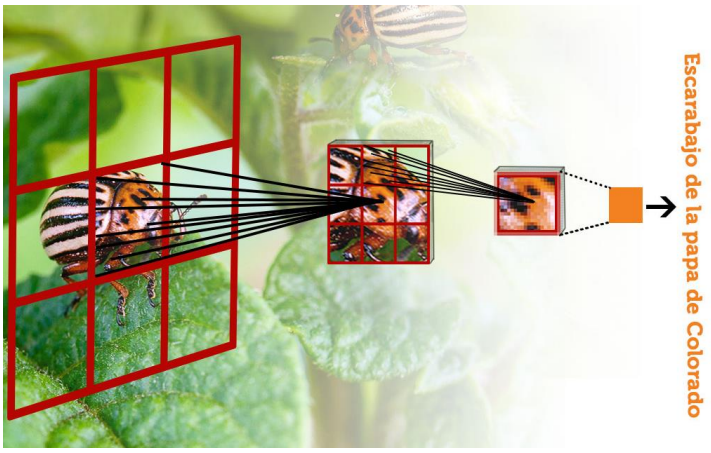

Figura 2. Uso de redes neuronales para la detección de plagas [5].

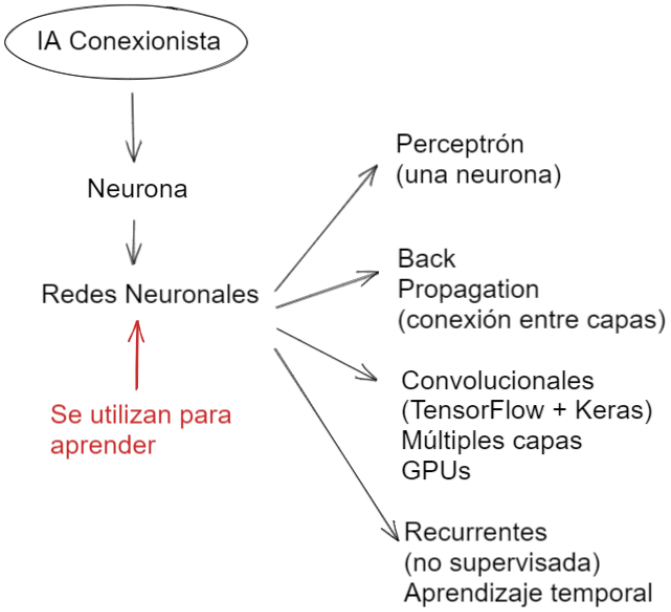

Figura 3. Resumen.

1.2 EL PERCEPTRON

INTRODUCCION

El perceptrón es la forma más simple de una red neuronal usada para la clasificación de un tipo especial de patrones, los linealmente separables (es decir, patrones que se encuentran a ambos lados de un hiperplano). Básicamente, consiste en una neurona con pesos sinápticos y umbral ajustables, como se muestra en la figura (4). El algoritmo usado para ajustar los parámetros libres de esta red neuronal apareció por primera vez en un procedimiento de aprendizaje desarrollado por Rosenblatt (1958) para su modelo de perceptrón del cerebro. En realidad, Rosenblatt demostró que, si los patrones usados para entrenar el perceptrón son sacados de dos clases linealmente separables, entonces el algoritmo del perceptrón converge y toma como superficie de decisión un hiperplano entre estas dos clases. La prueba de convergencia del algoritmo es conocida como el teorema de convergencia del perceptrón [6].

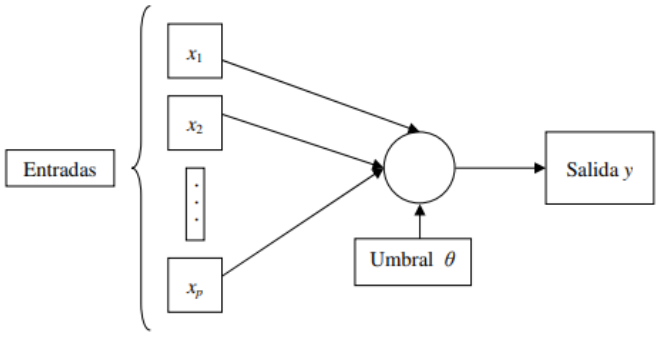

Figura 4. Representación gráfica de un perceptrón de una sola neurona.

El perceptrón de una capa descrito en la figura (4) tiene sólo una neurona. Dicho perceptrón está limitado a realizar clasificación de patrones con sólo dos clases. Expandiendo la capa de salida del perceptrón para incluir más que una neurona, podemos realizar dicha clasificación con más de dos clases. Sin embargo, las clases tendrían que ser linealmente separables para que el perceptrón trabaje correctamente.

CONSIDERACIONES BASICAS

Como ya hemos visto, el modelo de una neurona consiste en un combinador lineal seguido de un limitador. Este modelo se puede ver en la figura (5). El nodo suma del mismo mide una combinación lineal de las entradas aplicadas a sus sinapsis y, además, representa un umbral aplicado externamente. La suma resultante es aplicada a un limitador. Así, de acuerdo con esto, la neurona produce una salida igual a +1 si la entrada del limitador es positiva, y -1 si es negativa.

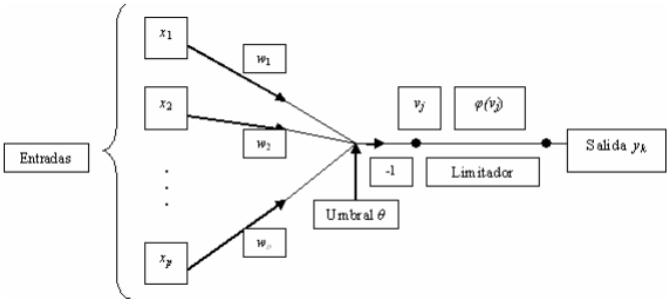

Figura 5. Modelo de un Perceptron Simple.

¿CUÁNDO USAR EL PERCEPTRON DE UNA CAPA?

Cuando los elementos de diferentes categorías pueden ser separados por una sola línea sin que estos elementos se mezclen entre sí. El perceptrón puede ser visto como una red neuronal artificial de una sola neurona (por eso se dice que el perceptrón es unicapa).

A continuación, te presentamos la separación lineal del grupo azul y del grupo rojo, como se dijo arriba, el perceptrón solamente puede hacer esta división, por ello no es tan exacto, pues aún cabe la posibilidad de que elementos de un grupo crucen la línea del grupo contrario [7].

Figura 6. Separación.

Anteriormente hemos hablado sobre el perceptrón, pero más específicamente sobre el perceptrón simple, a continuación, vamos a profundizar un poco más sobre el perceptrón simple.

PROFUNDIZACION EN EL PERCEPTRON SIMPLE

El perceptrón simple fue introducido por Rosenblatt (1962) y es un modelo unidireccional compuesto por dos capas de neuronas, una de entrada y otra de salida. La operación en un perceptrón simple que consta de n neuronas de entrada y m neuronas de salida se puede expresar como:

$$y_i = f\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right)$$

con i = 1, . . . , m.

Las neuronas de entrada son discretas y la función de activación de las neuronas de la capa de salida es de tipo escalón. Véase la Figura 7.

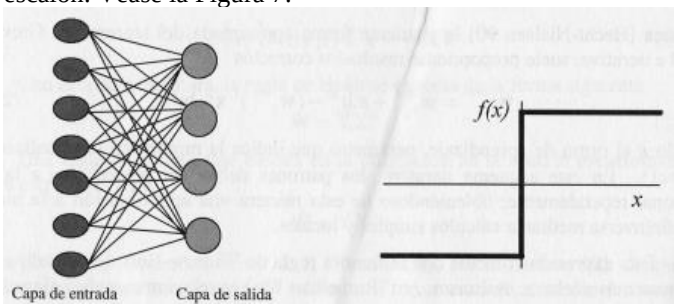

Figura 7. Arquitectura (izquierda) y función de transferencia (derecha) de un perceptrón simple.

El perceptrón simple puede utilizarse como clasificador, radicando su importancia histórica en su carácter de dispositivo entrenable, ya que el algoritmo de aprendizaje del modelo introducido por Rosenblatt (1962) permite determinar automáticamente los pesos sinápticos que clasifican un conjunto de patrones a partir de un conjunto de ejemplos etiquetados [8].

Veamos con un ejemplo sencillo, que contiene dos neuronas de entrada, que el perceptrón simple tan sólo puede discriminar entre dos clases linealmente separables, es decir, clases cuyas regiones de decisión pueden ser separadas mediante una única condición lineal o hiperplano.

Si denotamos por x1 y x2 a las dos neuronas de entrada, la operación efectuada por el perceptrón simple consiste en:

$$y = \begin{cases} 1 & \text{si } w_1 x_1 + w_2 x_2 \geq \theta \\ 0 & \text{si } w_1 x_1 + w_2 x_2 < \theta \end{cases}$$

Si consideramos x1 y x2 situadas sobre los ejes de abscisas y ordenadas respectivamente, la condición

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

es equivalente a

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$$

y representa una recta que define la región de decisión determinada por el perceptrón simple. Es por ello que dicho perceptrón simple representa un discriminador lineal, al implementar una condición lineal que separa dos regiones en el espacio que representan dos clases diferentes de patrones. Véase la Figura 8.

Por tanto, el perceptrón simple presenta grandes limitaciones, ya que tan sólo es capaz de representar funciones linealmente separables. Basándose en este hecho, Minsky y Papert (1969) publicaron un trabajo exponiendo las limitaciones del perceptrón simple, como consecuencia del cual muchos de los recursos que se venían dedicando a las redes neuronales se desviaron a otros campos de la inteligencia artificial.

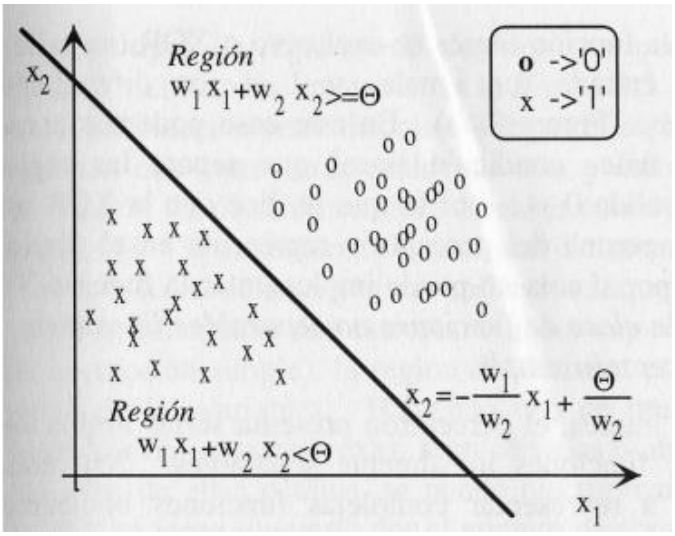

Figura 8. Región de decisión correspondiente a un perceptrón simple con dos neuronas de entrada.

Tal y como se ha comentado previamente, la importancia del perceptrón simple radica en el hecho de su carácter entrenable, ya que el algoritmo introducido por Rosenblatt (1962) permite que el perceptrón simple determine automáticamente los pesos sinápticos que clasifican un conjunto de patrones etiquetados.

El algoritmo de aprendizaje del perceptrón simple pertenece al grupo de los algoritmos que se fundamentan en la corrección de errores. Los algoritmos de este tipo ajustan los pesos de manera proporcional a la diferencia existente entre la salida actual de la red neuronal y la salida deseada, con el objetivo de minimizar el error actual de la red.

1.3 ENTRENAMIENTO DEL PERCEPTRON

EL PERCEPTRON (Conceptos generales)

El perceptrón es la representación de una neurona del cuerpo humano [9].

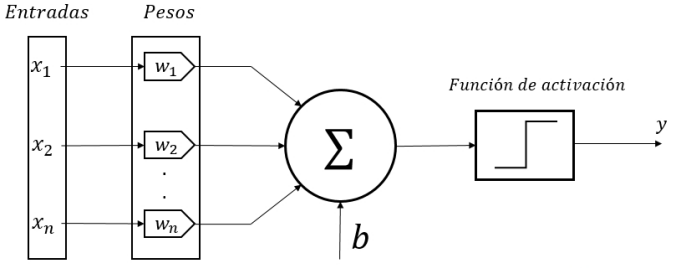

Figura 9. Perceptrón.

El perceptrón se compone por:

- Entradas: Es la información que recibe el perceptrón. Un ejemplo de esto se da en el aprendizaje de colores. Cuando el niño observa un color, sus ojos capturan una imagen con características específicas que le permiten identificar un color especifico. Cada

una de estas características se considera una entrada en el modelo.

- Pesos: Son valores numéricos que se encargan de establecer la influencia de una entrada en la salida deseada. Por ejemplo, al determinar si una persona tiene la posibilidad de sufrir un infarto se evalúan valores de entrada como obesidad, falta de ejercicio, diabetes, entre otros. Estas características podrían indicar si la persona puede sufrir un ataque, sin embargo, características como la obesidad y la diabetes incrementan mucho más el riesgo que la falta de ejercicio, por lo que se consideran más significativas al momento de predecir el infarto.

- Bias: Es un parámetro que tienen algunos modelos de redes neuronales el cual permite encontrar fácilmente la separación entre posibilidades de salida de una red neuronal.

- Función de activación: Es una función matemática que se encarga de determinar un valor de salida una vez se han procesado cada una de las entradas. En el aprendizaje de colores se aplica cuando se debe clasificar el color. Dependiendo de las características, el niño decide cual es el color que visualizó.

PASOS PARA ENTRENAR EL PERCEPTRON

Teniendo en cuenta el ejemplo de la compuerta AND del curso, realizaremos el entrenamiento del perceptrón con diferentes pesos y bias a los que allí se indican. Esto con la finalidad de mostrar que sucede cuando el perceptrón no se entrena en la primera iteración.

Compuerta AND		
Entradas		Salidas
x_1	x_2	$y = x_1 * x_2$
0	0	0
0	1	0
1	0	0
1	1	1

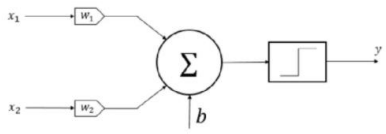

Figura 10. Compuerta AND.

PASO 1: INICIAR LOS PESOS Y LAS BIAS

Cada entrada del perceptrón debe tener un peso. Estos valores pueden ser aleatorios, sin embargo, es mejor empezar con valores pequeños.

$$W_1 = 10 \qquad W_2 = 10 \qquad b = -8$$

PASO 2 : CALCULAR LAS SALIDAS (net)

En este paso se calcula cada salida que teniendo en cuenta los posibles valores pueden tomar las entradas del perceptrón. Para este caso se realiza 4 veces debido a que la compuerta AND solo tiene ese número de posibilidades.

Es importante resaltar que la salida que se obtiene en este paso es la que se da después de la sumatoria. Por convención se nombrará en adelante como net así como se ve en la imagen.

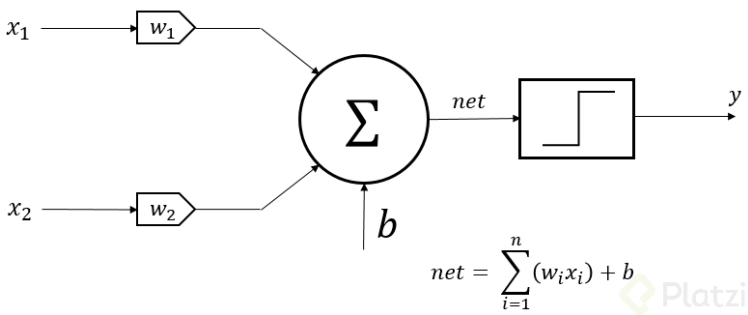

Figura 11.Calculo de las salidas.

Para calcular las posibles salidas se utilizará la fórmula de la imagen teniendo en cuenta cada una de las entradas (w_i). Dichas entradas se multiplican con sus respectivos pesos y se suman. Por último, se adiciona el bias.

Para este ejemplo con dos entradas, el net es igual a:

$$net_n = w_1 x_1 + w_2 x_2 + b$$

En donde el subíndice n hace referencia al caso con el que se está probando. Para este ejemplo, como se mencionó anteriormente, tenemos 4 posibilidades.

Compuerta AND			
Entradas		Salidas	n
x_1	x_2	$y = x_1 * x_2$	
0	0	0	1
0	1	0	2
1	0	0	3
1	1	1	4

Figura 12. Cuatro posibilidades.

Posibilidad 1 **x1 =0 ; x2 = 0**

$$net_1 = 10 * (0) + 10 * (0) - 8 \rightarrow net_1 = -8$$

PASO 3: OBTENER LA SALIDA UTILIZANDO LA FUNCION DE ACTIVACION Y CALCULAR CADA VALOR DE ERROR

En este ejercicio se utiliza la siguiente función de activación.

$$y = 1 \qquad net > 0$$
$$y = 0 \qquad net \leq 0$$

Figura 13. Paso 3.

Pasamos los valores net obtenidos en la función de activación.

$$net_1 = -8 \rightarrow y_1 = 0$$

Ahora calculamos el error con la siguiente ecuación:

$$e = y' - y$$

En donde:

$e \rightarrow$ Valor del error

$y' \rightarrow$ Salida deseada

$y \rightarrow$ Salida obtenida

$$e_1 = 0 - 0 \rightarrow e_1 = 0$$

POSIBLES VALORES DEL ERROR

Al calcular el error se debe tener en cuenta que se puede presentar alguno de los siguientes casos:

Caso 1: Si $e_n = 0$, repita el paso 2 con las siguientes entradas.

Caso 2: Si $e_n \neq 0$, continúe con el paso 4.

Caso 3: Si todos los valores calculados del error son iguales a 0 para todas las entradas. el entrenamiento finaliza

Figura 14. Casos para calcular el error.

PASO 4: CORREGIR EL BIAS Y LOS PESOS

Para realizar las correcciones se utilizan las siguientes ecuaciones:

Corrección de los pesos	$w_i(k+1) = w(k) + e * x_i$
Corrección del bias	$b(k+1) = b(k) + e$

Figura 15. Ecuaciones.

En donde:

$w_n(k+1) \rightarrow$ *Nuevo peso* $\qquad b(k+1) \rightarrow$ *nuevo bias*

$w(k) \rightarrow$ *Peso actual* $\qquad b(k) \rightarrow$ *bias actual*

$e \rightarrow$ *error* $\qquad x_i \rightarrow$ *entrada*

Figura 16. Representación.

Es importante tener en cuenta que siempre que el error sea diferente de 0 se deben corregir los pesos y el bias como se observa en los casos del paso 3.

ENTRENAMIENTO DEL PERCEPTRON

Compuerta AND		
Entradas		**Salidas**
x_1	x_2	$y = x_1 * x_2$
0	0	0
0	1	0
1	0	0
1	1	1

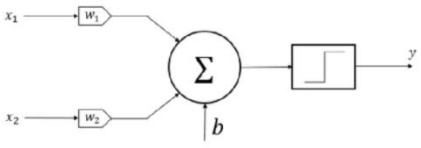

Figura 17. Ejemplo 1.

Tomamos los valores inicializados en la explicación del paso 1.

$$W_1 = 10 \qquad W_2 = 10 \qquad b = -8 \rightarrow \textbf{Paso 1}$$

Iteración 1:

$$net_1 = 10 * (0) + 10 * (0) - 8 \rightarrow net_1 = -8 \rightarrow \textbf{Paso 2}$$

$$net_1 = -8 \rightarrow y_1 = 0$$

$$e_1 = 0 - 0 \rightarrow e_1 = 0 \rightarrow \textbf{Paso 3}$$

Figura 18. Iteración 1.

Como el error es igual a 0, continuamos con las siguientes entradas.

$$net_2 = 10 * (0) + 10 * (1) - 8 \rightarrow net_2 = 2 \rightarrow \boldsymbol{Paso\ 2}$$

$$net_2 = 2 \rightarrow y_2 = 1$$

$$e_2 = 0 - 1 \rightarrow e_2 = -1 \rightarrow \boldsymbol{Paso\ 3}$$

En este caso como el error es diferente de 0 se deben corregir los pesos como se menciona en el paso 4.

CORRECION DE PESOS

$$\boldsymbol{w_i(k+1) = w_i(k) + e * x_i}$$

$$w_1(k+1) = 10 + (-1 * 0) \rightarrow w_1(k+1) = 10$$

$$w_2(k+1) = 10 + (-1 * 1) \rightarrow w_2(k+1) = 9$$

Figura 19. Corrección de erros.

CORRECCION DEL BIAS

$$\boldsymbol{b(k+1) = b(k) + e}$$

$$b(k+1) = -8 - 1 \rightarrow b(k+1) = -9$$

Figura 20. Corrección del Bias.

Una vez se realizan las correcciones, continuamos con las entradas que siguen.

$$net_3 = 10 * (1) + 9 * (0) - 9 \rightarrow net_3 = 1 \rightarrow \boldsymbol{Paso\ 2}$$

$$net_3 = 1 \rightarrow y_3 = 1$$

$$e_3 = 0 - 1 \rightarrow e_3 = -1 \rightarrow \boldsymbol{Paso\ 3}$$

Figura 21. Entrada 3

Corregimos nuevamente los pesos y el bias.

CORRECCIÓN DE PESOS

$$\boldsymbol{w_i(k+1) = w_i(k) + e * x_i}$$

$$w_1(k+1) = 10 + (-1 * 1) \rightarrow w_1(k+1) = 9$$

$$w_2(k+1) = 9 + (-1 * 0) \rightarrow w_2(k+1) = 9$$

Figura 22. Corrección de los pesos.

CORRECION DEL BIAS

$$\boldsymbol{b(k+1) = b(k) + e}$$

$$b(k+1) = -9 - 1 \rightarrow b(k+1) = -10$$

Figura 23. Corrección del bias.

Modificamos los pesos y el bias y continuamos con la ultima posibilidad de entrada.

$$net_4 = 9 * (1) + 9 * (1) - 10 \rightarrow net_4 = 8$$

$$net_4 = 8 \rightarrow y_4 = 1$$

$$e_4 = 1 - 1 \rightarrow e_4 = 0 \rightarrow \boldsymbol{Paso\ 3}$$

Figura 24. Entrada 4.

Por último, se realiza otra iteración para comprobar que las salidas de las otras entradas no se han afectado con las ultimas correcciones realizadas.

Iteración 2:

$$net_1 = 9 * (0) + 9 * (0) - 10 \rightarrow net_1 = -10 \qquad y_1 = 0 \rightarrow e_1 = 0$$

$$net_2 = 9 * (0) + 9 * (1) - 10 \rightarrow net_2 = -1 \qquad y_2 = 0 \rightarrow e_2 = 0$$

$$net_3 = 9 * (1) + 9 * (0) - 10 \rightarrow net_3 = -1 \qquad y_3 = 0 \rightarrow e_3 = 0$$

$$net_4 = 9 * (1) + 9 * (1) - 10 \rightarrow net_4 = 8 \qquad y_4 = 1 \rightarrow e_4 = 0$$

Figura 25. Iteración 2.

1.4 BACKPROPAGATION

QUE ES BACKPROPAGATION

La propagación hacia atrás o retro propagación (del inglés backpropagation) es un método de cálculo del gradiente utilizado en algoritmos de aprendizaje supervisado utilizados para entrenar redes neuronales artificiales. El método emplea un ciclo propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo, las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total [10].

IMPORTANCIA

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento.

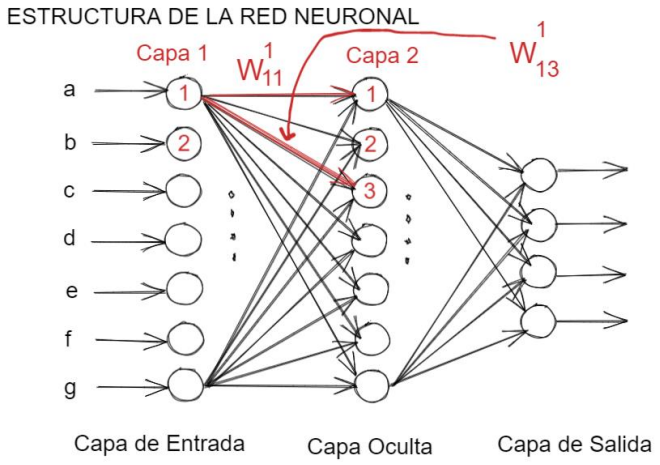

Figura 26. Red neuronal.

REGLA DE APRENDISAJE

La red retropropagada trabaja bajo aprendizaje supervisado y por tanto necesita un conjunto de entrenamiento que le describa cada salida y su valor de salida esperado de la siguiente forma:

$$(p_1, t_1), (p_2, t_2), \ldots, (p_q, t_q)(1)$$

Donde Pq es una entrada a la red y tq es la correspondiente salida deseada para el patrón q-ésimo. El algoritmo debe ajustar los parámetros de la red para minimizar el error cuadrático medio. Es importante recalcar que no existe una técnica para determinar el número de capas ocultas, ni el número de neuronas que debe contener cada una de ellas para un problema específico, esta elección es determinada por la experiencia del diseñador, el cual debe cumplir con las limitaciones de tipo computacional. Cada patrón de entrenamiento se propaga a través de la red y sus parámetros para producir una respuesta en la capa de salida, la cual se compara con los patrones objetivo o salidas deseadas para calcular el error en el aprendizaje, este error marca el camino más adecuado para la actualización de los pesos y ganancias que al final del entrenamiento producirán una respuesta satisfactoria a todos los patrones de entrenamiento, esto se logra minimizando el error cuadrático medio en cada iteración del proceso de aprendizaje.

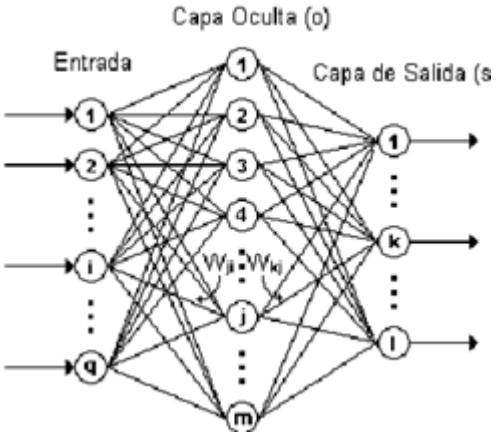

Figura 27. Imagen de ejemplo.

q: Número de componentes del vector de entrada.
m: Número de neuronas de la capa oculta.
l: Número de neuronas de la capa de salida.

Para iniciar el entrenamiento se le presenta a la red un patrón de entrenamiento, el cual tiene q componentes como se describe en la ecuación (2):

$$p = \begin{bmatrix} p_1 \\ p_2 \\ \cdot \\ \cdot \\ p_i \\ \cdot \\ \cdot \\ p_q \end{bmatrix} (2)$$

Figura 28. Ecuacion (2).

Cuando se le presenta a la red un patrón de entrenamiento, este se propaga a través de las conexiones existentes produciendo una entrada neta n en cada una las neuronas de la siguiente capa, la entrada neta a la neurona j de la siguiente capa debido a la presencia de un patrón de entrenamiento en la entrada está dada por la ecuación (3), nótese que la entrada

neta es el valor justo antes de pasar por la función de transferencia:

$$n_j^0 = \sum_{i=1}^{q}(w_{ji}^0 * p_i) + b_j^0 (3)$$

Figura 29. Ecuación (3).

w_{ji}^0: Peso que une la componente i de la entrada con la neurona j de la capa oculta.

p_i: Componente i del vector p que contiene el patrón de entrenamiento de q componentes.

b_j^0 Ganancia de la neurona j de la capa oculta.

Donde el 0 representa la capa oculta a la que pertenece cada parámetro. Cada una de las neuronas de la capa oculta tiene como salida a_j^0 dada por:

$$a_j^0 = f^0(\sum_{i=1}^{q}(w_{ji}^0 * p_i) + b_j^0)(4)$$

Figura 30. Ecuación (4).

f^0 Función de transferencia de las neuronas de la capa oculta.

Las salidas a_j^0 de las neuronas de la capa oculta (de m componentes) son las entradas a los pesos de conexión de la capa de salida, este comportamiento está descrito por:

$$n_k^s = \sum_{j=1}^{m}(w_{kj}^s * a_j^0) + b_k^s (5)$$

Figura 31. Ecuación (5).

w_{kj}^s: Peso que une la neurona j de la capa oculta con la neurona k de la capa de salida, la cual cuenta con l neuronas.

a_j^0: Salida de la neurona j de la capa oculta, la cual cuenta con m neuronas.

b_k^s: Ganancia de la neurona k de la capa de salida.

n_k^s: Entrada neta a la neurona k de la capa de salida.

La red produce una salida final:

$$a_k^s = f^s(n_k^s)(6)$$

Figura 32. Ecuación (6).

f^s:Función de transferencia de las neuronas de la capa de salida.

La salida de la red de cada neurona a_k^s se compara con la salida deseada t_k para calcular el error en cada unidad:

$$\delta_k = (t_k - a_k^s)$$

El error debido a cada patrón p propagado está dado por:

$$ep^2 = 1/2\sum_{k=1}^{l}(\delta_k^2)$$

Figura 33. Fórmula.

ep^2: Error cuadrático medio para cada patrón de entrada p.

δ_k: Error en la neurona k de la capa de salida con l neuronas.

Este proceso se repite para el número total de patrones de entrenamiento (r), para un proceso de aprendizaje exitoso el objetivo del algoritmo es actualizar todos los pesos y ganancias de la red minimizando el error cuadrático medio total descrito en:

$$e^2 = \sum_{p=1}^{r}(ep^2)$$

Figura 34. Formula 2.

e^2 Error total en el proceso de aprendizaje en una iteración después de haber presentado a la red los r patrones de entrenamiento. El error que genera una red neuronal en función de sus pesos, genera un espacio de n dimensiones, donde n es el número de pesos de conexión de la red, al evaluar el gradiente del error en un punto de esta superficie se obtendrá la dirección en la cual la función del error tendrá un mayor crecimiento, como el objetivo del proceso de aprendizaje es minimizar el error debe tomarse la dirección negativa del gradiente para obtener el mayor decremento del error y de esta forma su minimización, condición requerida para realizar la actualización de la matriz de pesos en el algoritmo Backpropagation.

REFERENCIAS

Referencias en la Web:

[1] https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

[2] https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

[3] https://www.frro.utn.edu.ar/repositorio/catedras/quim

ica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

[4] https://www.datacentric.es/blog/insight/red-neuronal-artificial-aplicaciones/

[5] http://ciencia.unam.mx/leer/773/redes-neuronales-artificiales-contribuyen-en-el-desarrollo-de-aplicaciones-en-beneficio-humano

[6] http://bibing.us.es/proyectos/abreproy/11084/fichero/Memoria+por+capítulos+%252FCapítulo+4.pdf+#:~:text=El%20perceptrón%20es%20la%20forma,ambos%20lados%20de%20un%20hiperplano).&text=La%20prueba%20de%20convergencia%20del,teorema%20de%20convergencia%20del%20perceptrón.

[7] https://www.crehana.com/co/blog/desarrollo-web/que-es-perceptron-algoritmo/

[8] http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t14-neuronales.pdf

[9] https://platzi.com/contributions/entrenamiento-del-perceptron/

[10] https://es.wikipedia.org/wiki/Propagación_hacia_atrás