Entrenamiento del Perceptrón

Perceptron Training

**Joan Sebastián Ángel Castaño**

**Universidad Tecnológica de Pereira, Pereira, Colombia**

**Correo: jsangel@utp.edu.co**

***Resumen*—** Al inicio del desarrollo de las redes neuronales se basaron en el funcionamiento de las neuronas bilógicas, por eso se estudia a profundidad el perceptrón que es la unidad básica de una red neuronal, el cual simula matemáticamente una neurona biológica. Podemos encontrar que el perceptrón cumple una función de clasificación, por lo que es un método muy utilizado en diferentes ámbitos. A partir de un algoritmo de entrenamiento el perceptrón comienza a tener más fuerza en cuanto a reconocimiento de patrones de entrada complejos. Veremos el algoritmo que permite dicho aprendizaje a partir de ciertos ejemplos de entrada que serán evaluados al momento de calcular alguna salida de la red trabajada con perceptrones.

En el campo de las redes neuronales, el perceptrón se define como la unidad básica de inferencia en forma de discriminador lineal, en pocas palabras, es una neurona artificial. Es por esto que requiere un tipo de aprendizaje para tener utilidad alguna; en el perceptrón, existen dos tipos de aprendizaje, el primero utiliza una tasa de aprendizaje mientras que el segundo no la utiliza. El algoritmo de aprendizaje es el mismo para todas las neuronas.

***Palabras clave—*** Perceptrón, Inferencia, Red Neuronal, Aprendizaje, Inteligencia Artificial, Algoritmo, Neurona, Entrada, Salida.

*Abstract*—

At the beginning of the development of neural networks were based on the functioning of biological neurons, so the perceptron is studied in depth, which is the basic unit of a neural network, which simulates a biological neuron mathematically. We can find that the perceptron fulfills a classification function, so it is a method widely used in different areas. From a training algorithm the perceptron begins to have more strength in recognition of complex entry patterns. We will see the algorithm that allows this learning from certain input examples that will be evaluated when calculating some output of the network worked with perceptrons.

In the fields of neuronal networks, the perceptron is defined as the most basic unity of inference in form of a lineal discriminator, in few words, it is an artificial neuron. Because of this it requires a special kind of learning to have some utility; in the perceptron, there are two types of learning, the first uses a rate of learning meanwhile the second doesn’t. The learning algorithm is the same for all neurons.

*Key Word* — Perceptron, Inference, Neuronal Network, Learning, Artificial Intelligence, Algorithm, Neuron, Input, Output.

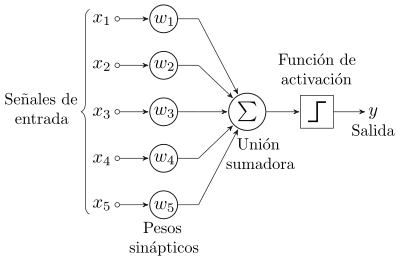
### INTRODUCCIÓN

El perceptrón es una representación matemática de una neurona bilógica, la cual recibe ciertos parámetros de entrada y genera una salida. Es la unidad básica que compone a una red neuronal. Una de las principales características de las redes neuronales es su capacidad para aprender a partir de alguna fuente de información interactuando con su entorno. El psicólogo Frank Rosenblat desarrolló un modelo simple de neurona basado en el modelo de McCulloch y Pitts y en una regla de aprendizaje basada en la corrección del error. A este modelo le llamó Perceptrón en 1958.

Una de las características que más interés despertó de este modelo fue la capacidad de aprender a reconocer patrones. El Perceptrón está constituido por conjunto de sensores de entrada que reciben los patrones de entrada a reconocer o clasificar y una neurona de salida que se ocupa de clasificar a los patrones de entrada en dos clases, según que la salida de la misma es binaria.

Sin embargo, este modelo tiene muchas limitaciones, como, por ejemplo, no es capaz de aprender la función lógica XOR; además tuvieron que pasar varios años hasta que se propusiera la regla de aprendizaje de retro propagación del error para demostrarse que el Perceptrón multicapa es un aproximador universal.

### DESARROLLO DEL TEMA



En la figura podemos ver los componentes del perceptrón que es similar la neurona biológica.

A partir del perceptrón, mediante un algoritmo, podemos clasificar subgrupos de un grupo mayor de las posibles salidas que genera el perceptrón. El problema que podemos identificar en este modelo de un solo perceptrón, es que hay ciertos problemas los cuales no es posible hacer la clasificación, por ejemplo la función XOR.

**Aprendizaje del perceptrón**

A partir de unos ejemplos previamente evaluados podemos entrenar a un perceptrón. El reconocimiento de patrones es una de las principales aplicaciones de los perceptrones, dichos patrones son los linealmente separables (es decir, patrones que se encuentran a ambos lados de un hiperplano), posteriormente clasificados en dos subgrupos los cuales pueden ser 1 o 0.

Inicialmente el perceptrón no fue diseñado para el reconocimiento de patrones de entrada muy complejos, pero se descubrió que mediante un proceso de aprendizaje era capaz de adquirir esta capacidad.

En esencia, el entrenamiento implicaba un proceso de refuerzo mediante el cual la salida de las unidades *A* se incrementaba o se decrementaba dependiendo de si las unidades *A* contribuían o no a las respuestas correctas del perceptrón para una entrada dada.

Como sabemos, un algoritmo de aprendizaje es un método adaptativo por el que una red se auto modifica para implementar el comportamiento deseado. Esto se hace presentando algunos ejemplos de la función entrada-salida a la red. Se presenta un ejemplo y se ejecuta una acción correctiva iterativamente hasta que la red aprende a producir la respuesta deseada. El conjunto de entrenamiento es el conjunto de los ejemplos de los que la red va a aprender.

Usamos la siguiente notación: El vector de entrada al perceptrón es x = (x1,..., xn). Si los pesos son los valores reales w1,..., wn y el umbral es U, diremos que

w = (w1, ..., wn, wn+1) con wn+1 = – U es el vector extendido de pesos del perceptrón y que (x1,..., xn, 1) es el vector extendido de entradas (se añade la entrada de tendencia o bias con valor 1 fijo). La computación de un perceptrón puede expresarse mediante un producto escalar, ya que ƶi wi xi• U es equivalente a w • x • 0 donde w y x son los vectores extendidos de pesos y entradas, respectivamente.

La convergencia del algoritmo de aprendizaje del perceptrón se basa en que cada perceptrón realiza la comprobación

w • x > 0 ó w • x • 0, pero son equivalentes cuando el conjunto de entrenamiento es finito, lo cual siempre es cierto en problemas prácticos. Una forma habitual de comenzar el algoritmo de entrenamiento es inicializando aleatoriamente los pesos de la red y mejorar los parámetros iniciales, comprobando a cada paso, si puede lograrse una separación mejor del conjunto de entrenamiento. Cada vector de pesos w define un hiperplano que separa los puntos con salida

1 (w • x • 0) de los puntos con salida 0 (w • x 0).

**Algoritmo de entrenamiento del perceptrón**

Iniciar: **w0** aleatoriamente; t 0

Repetir Mientras quede vectores mal clasificados

Seleccionar aleatoriamente un vector **x**

Si **x** ε P (salida esperada 1) y **wt • x** < 0 (salida calculada 0) entonces

[Hay un error]

**wt+1** **wt + x** ; t t+1

Fin si

Si **x** ε N (salida esperada 0) y **wt • x** >= 0 (salida calculada 1) entonces

[Hay un error]

**wt+1** **wt – x** ; t t+1

fin si

fin repite

También se encuentra otra forma de escribir el algoritmo, en forma esquemática, para varias salidas.

Notamos los ejemplos de entrenamiento (**X**, **d**), con **X** la entrada y **d** la salida esperada.

0.- Fijar los pesos para las entradas Wij i = 1..p, j = 1..q

1.- Repetir pasos 2-4 Mientras existan ejemplos mal clasificados

2.- Seleccionar otro elemento de entrenamiento (X, d) mientras haya.

3.- Introducir X en red, calcular la salida Y: Yj = 1 si Σi

Xi Wij > U y 0 en otro caso

4.- [Comparar Y con d (salida esperada)]

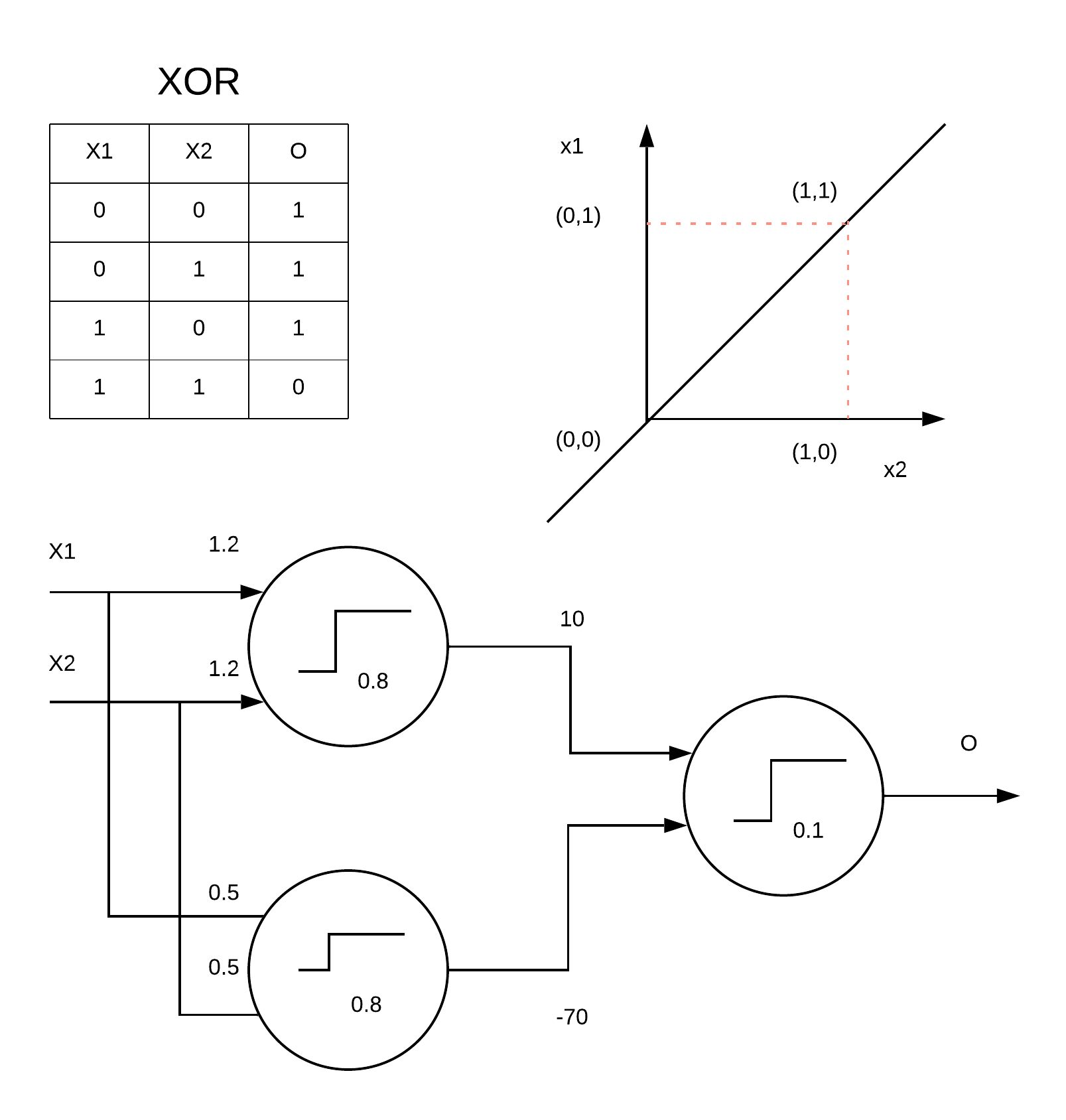
Para j = 1..q

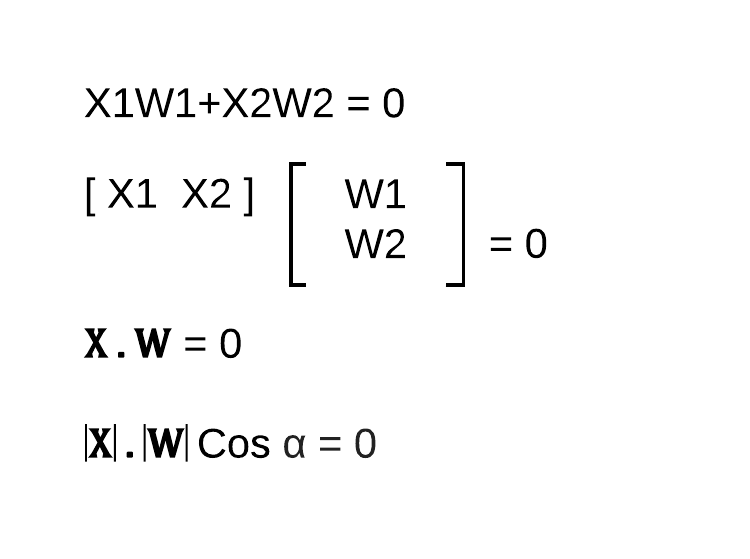
Si salida j es incorrecta, Entonces; W’ij = Wij - (dj –Yj) Xi; FinSi

FinPara [para cada salida (j) no correcta, Si Yj = 0 [x ε P; ΣiXi Wij < U], W’ij = Wij + Xi]

[Si Yj =1 [x ȫ N; ƶiXi Wij > U], W’ij = Wij - Xi]

5.- FIN





Según el Angulo que se forme entre X y W podemos organizar todos las coordenas en su correspondiente grupo

ALGORITMO

// # [Perceptron Classifier](http://en.wikipedia.org/wiki/Perceptron)

//

// This is a single-layer perceptron classifier that takes

// arrays of numbers and predicts whether they should be classified

// as either 0 or 1 (negative or positive examples).

function perceptron() {

var perceptron\_model = {},

// The weights, or coefficients of the model;

// weights are only populated when training with data.

weights = [],

// The bias term, or intercept; it is also a weight but

// it's stored separately for convenience as it is always

// multiplied by one.

bias = 0;

// ## Predict

// Use an array of features with the weight array and bias

// to predict whether an example is labeled 0 or 1.

perceptron\_model.predict = function(features) {

// Only predict if previously trained

// on the same size feature array(s).

if (features.length !== weights.length) return null;

// Calculate the sum of features times weights,

// with the bias added (implicitly times one).

var score = 0;

for (var i = 0; i < weights.length; i++) {

score += weights[i] \* features[i];

}

score += bias;

// Classify as 1 if the score is over 0, otherwise 0.

return score > 0 ? 1 : 0;

};

// ## Train

// Train the classifier with a new example, which is

// a numeric array of features and a 0 or 1 label.

perceptron\_model.train = function(features, label) {

// Require that only labels of 0 or 1 are considered.

if (label !== 0 && label !== 1) return null;

// The length of the feature array determines

// the length of the weight array.

// The perceptron will continue learning as long as

// it keeps seeing feature arrays of the same length.

// When it sees a new data shape, it initializes.

if (features.length !== weights.length) {

weights = features;

bias = 1;

}

// Make a prediction based on current weights.

var prediction = perceptron\_model.predict(features);

// Update the weights if the prediction is wrong.

if (prediction !== label) {

var gradient = label - prediction;

for (var i = 0; i < weights.length; i++) {

weights[i] += gradient \* features[i];

}

bias += gradient;

}

return perceptron\_model;

};

// Conveniently access the weights array.

perceptron\_model.weights = function() {

return weights;

};

// Conveniently access the bias.

perceptron\_model.bias = function() {

return bias;

};

// Return the completed model.

return perceptron\_model;

}

[1]

EXPLICACIÓN ALGORITMO

Esta implementación del perceptrón fue diseñada para tratar con clases linealmente separables, utilizando una función discriminante lineal para crear una frontera de decisión.

Inicialmente se crea una variable de tipo perceptrón, posteriormente una lista de pesos y un bias de 0.

Luego, para predecir la clase se usa un arreglo de características con el arreglo de pesos y el bias.atures with the weight array and bias para predecir si el ejemplo es 0 o 1.

La función de clasificación solo actúa si se entrenó previamente; primero se compara si el arreglo de características y pesos tiene tamaños iguales, esto con el fin de determinar si cada característica tiene asignado un peso y viceversa.

Se asigna la variable score al resultado de Característica\*Peso de cada elemento. Para clasificar en 0 o 1 se tiene que, si el score es mayor que 0, la clase es 1, de lo contrario es 0.

La función de entrenamiento descarta todas las clases que sean distintas de 0 o 1, el tamaño del arreglo de características determina el tamaño del arreglo de pesos, por lo que si son diferentes deberán igualarse; luego se define el bias con un valor de 1.

Luego, si la predicción es errónea (Si la predicción es distinta que la clase) se actualizan los pesos con un valor gradiente.

Finalmente se puede acceder a los arreglos de características y pesos y se retorna el modelo completo del perceptrón.

1. CONCLUSIONES

### El perceptrón, a pesar de ser una de las redes más utilizadas, no es una de las más potentes ya que posee ciertas limitaciones, por ejemplo, el caso del aprendizaje en problemas complejos.

### Este tipo de redes se pueden implementar en la vida moderna en ámbitos como análisis de series temporales, procesamiento de imágenes, reconocimiento automático del habla, diagnósticos médicos, entre otros.

### REFERENCIAS

[1] A JavaScript Perceptron. (2015) Disponible en:

<https://planspace.org/20150610-a_javascript_perceptron/>

[2] Perceptrón simple y multicapa. Disponible en: <https://es.slideshare.net/Jeffo92/perceptrn-simple-y-multicapa>