Algoritmo de Entrenamiento del Perceptrón

Computación Blanda

Perceptron Training Algorithm

Soft Computing

Autor: Jairo Alejandro Castrillon Libreros

*Ingeniería de Sistemas y Computación, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: alejandro.castrillon@utp.edu.co

***Resumen*—** 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.

### INTRODUCCIÓN

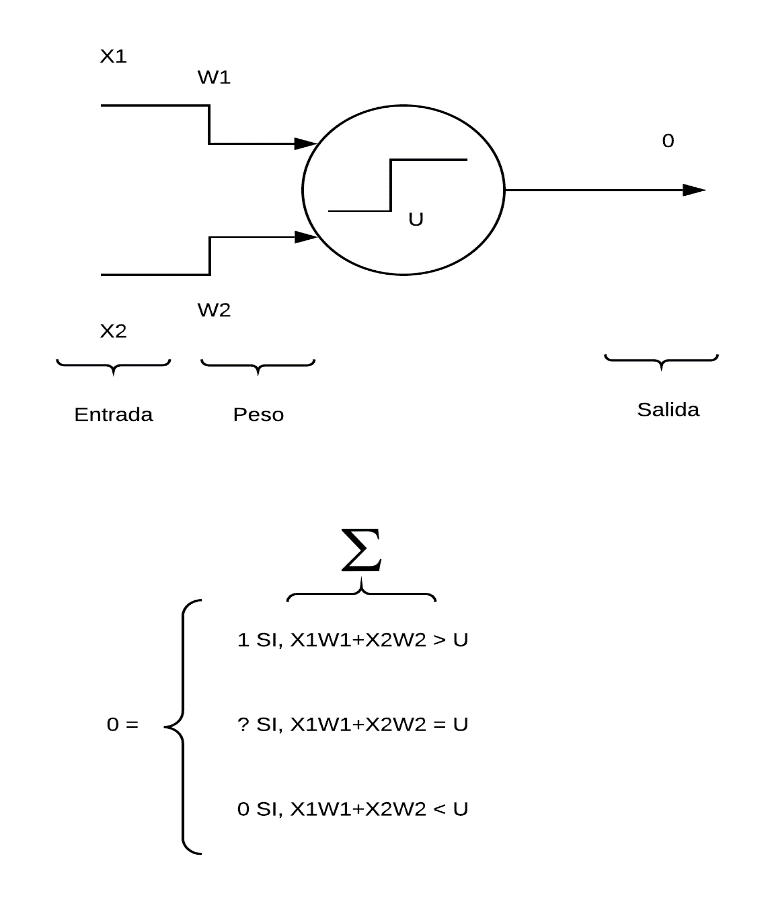
El psicólogo Frank Rosenblat desarrolló un modelo simple de neurona basado en el modelo de McCulloch y Pitts y en una regla de aprendizaje basada en la corrección del error. A este modelo le llamó Perceptrón en 1958. Una de las principales características de las redes neuronales es la capacidad que posee de aprender a partir de alguna fuente de información interactuando con su entorno. Pero la característica 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.

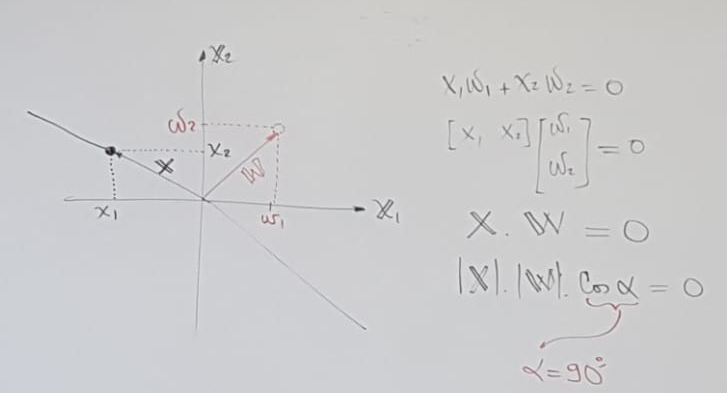
Sin embargo, este modelo tiene demasiados límites, como, por ejemplo, uno de esos límites es que no es capaz de aprender la función lógica XOR; además tuvieron que pasar varios años hasta que se propusiera la regla de aprendizaje de retropropagación del error para demostrarse que el Perceptrón multicapa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

### DESARROLLO

[3]

PROCESO





Según el ángulo 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 DEL 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 sólo 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. [2]

### 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>

[3] El perceptrón. Disponible en:

<http://www.lcc.uma.es/~munozp/documentos/modelos_computacionales/temas/Tema4MC-05.pdf>