Trabajo 2 Fundamentos de inteligencia computacional

Edison David González Blandón
Ingeniería Electrónica
Facultad de Ingeniería, Universidad de Antioquia
Medellín, Colombia
edavid.gonzalez@udea.edu.co

Juan Esteban Velásquez Franco
Ingeniería de telecomunicaciones
Facultad de Ingeniería, Universidad de Antioquia
Medellín, Colombia
juan.velasquez12@udea.edu.co

***Abstract*—. En este artículo se trabaja 3 ejercicios propuestos para profundizar los conceptos de la teoría de redes a través de la implementación de los algoritmos de RNA y aplicándolos para la solución de un caso particular de clasificación. Se usan específicamente los métodos de LMS (Least Mean Square), regresión lineal para el primer punto, algoritmo de aprendizaje perceptrón y algoritmo de regresión logística para el segundo y algoritmo de back propagation.**

Keywords—RNA, regresión, Back propagation, Least mean square.

I. INTRODUCCIÓN

En la actualidad existe una tendencia de establecer un nuevo campo de computación para resolver problemas que no puede establecer un modelo matemático o algorítmico tradicional, lo que hace que surja la necesidad de nuevos algoritmos como los que se trabajarán en este documento.
Las Redes Neuronales Artificiales (RNA) se caracterizan por su intento de simular una neurona real con ciertas características. Estas características son aproximadas a través de datos de entrada, funciones de activación, y pesos sinápticos.
La función de activación se elige dependiendo del problema que se desea solucionar y los pesos sinápticos se moldean a través del entrenamiento de la neurona, en el cual se realizaran

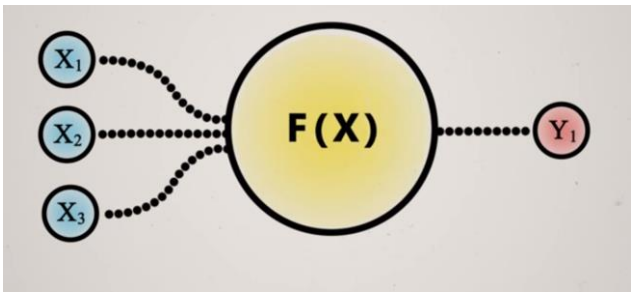


Fig. 1. Representación de una neurona real a una neurona artificial, donde X1, X2 y X3 son los datos de entrada, F(x) es la funcion de activación y el elemento que separa X1, X2 y X3 de F(x) se llama al peso sináptico

entrenamientos iterativos en el que se busca minimizar el error.

II. DESARROLLO

A. *Primer Problema*

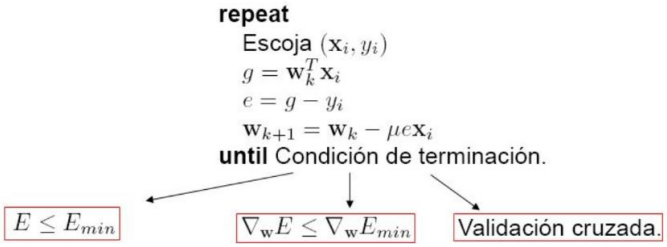
El problema consiste en la predicción del calor de una carga con 8 parámetros de entrada que son: Capacidad relativa, área superficial, área de muro, área de techo, altura total, área de acristalamiento, distribución del área de glaseado y como salida calentamiento de la carga. Entrenando una neurona usando el algoritmo de aprendizaje LMS y Regresión lineal.

El conjunto de datos se divide según el criterio de boostrapping, que es la división de los datos en un porcentaje dividido en un 80% y un 20% que a su vez el 80% se divide en 70% para entrenar y un 30% para validación del entrenamiento.
Luego de tener los datos seccionados mediante la función de entrenamiento y asignación de un Alpha entre 0 y 1 construyo el vector de pesos que posteriormente, en la validación verifico que el sistema quede bien entrenado, en nuestro ejercicio el Alpha más adecuado en los dos casos fue de 0.064, por último, se toma el 20% de los datos faltantes que son los datos de prueba y se acciona la neurona para posteriormente calcular el error, que más adelante se analizará cada uno.

LMS.

Este algoritmo usa la aproximación estocástica para el cálculo de la gradiente de la función mínima cuadrática MMSE (Minimun mean square Error). El MSE siempre sigue la dirección tangente a la superficie, el algoritmo LMS requiere un mayor número de iteraciones para llegar a la convergencia, también requiere un mayor número de coeficiente para funcionar, su cálculo de error es inferior al de los otros algoritmos adaptativos, pero tiene la ventaja de ser más fácil de entender matemáticamente, puesto que tiene características lineales, lo que hace que el sistema sea robusto. [1]
Este algoritmo es muy usado como filtro adaptativo para la cancelación de eco acústicos en sistemas de telecomunicaciones. [2] [3]
El algoritmo implementado para esta tarea es en lenguaje Python. Primero se importan los datos con la librería pandas, se almacenan en listas, siguiente a eso se hace el boostrapping que es adquiriendo el tamaño de la lista y creando nuevas con los porcentajes de datos necesarios.
Un proceso de filtrado, que involucra:

el cómputo de la salida de un filtro lineal en respuesta a una señal de entrada, y la generación de una estimación del error mediante la comparación de esta salida con la señal deseada.
Un proceso adaptativo, que involucra el ajuste automático de los parámetros del filtro de acuerdo al error estimado.
Ya para la implementación del código de LMS, se define una función que tiene como entrada una lista de entrenamiento, iteraciones, Alpha y error mínimo y retorna un vector de pesos para posteriormente con la función de validación verificar que el entrenamiento sea el correcto. La forma en cómo se varían los pesos y se obtienen unos pesos óptimos dependen de mu (mi) que en mi caso lo trate como Alpha (Alfa) (Siguen siendo solo una variable de representación) su valor se encuentra entre 0 y 1, después de analizar, se culminó viendo que el valor de error fluctúa menos y es mejor si se encuentra entre: 10**-2 y 10**-1, nuestro Alpha adecuado fue de 0.064
El proceso iterativo que se llevó a cabo en el algoritmo es el siguiente:
Se escoge un punto aleatorio y se asigna un peso aleatorio, luego se haya el gradiente "hacia debajo de la colina".



Ya cuando se hace la prueba se obtuvo un error del 8.86, este error es calculado con el promedio ponderado del error cuadrático medio de cada prueba.

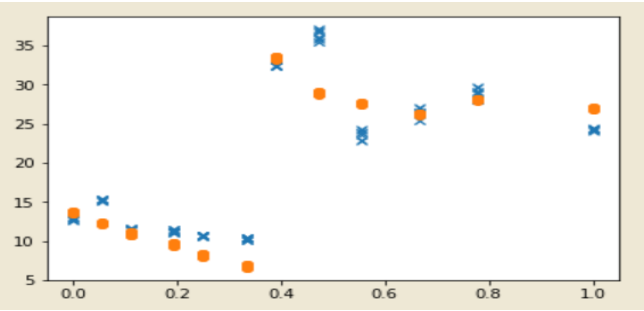


Fig. 2. LMS. Los puntos que están representado por X en color azul representan los valores reales, y los O de color naranja representa la predicción.

REGRESIÓN LINEAL.

De la misma forma que LMS se hace llamado de datos y se desordena como lo pide el bootstrapping,
La regresión lineal es un método de iteración para encontrar los valores de un peso óptimo para predecir el valor más acertadamente, a diferencia de LMS la regresión lineal se trabaja a base de vectores pero generalmente no se puede hacer ese tipo de operaciones, lo que se hace es hacer un modelo matemático llamado gradiente descendiente, en la cual a partir de la derivada de la función se conoce hacia donde esta crece según el gradiente, es decir, que decrece en sentido contrario al gradiente[4] , en cambio LMS trabaja punto por punto. Trabajar con este tipo de algoritmo hace que el costo computacional sea alto, por eso se recomienda no usar regresiones para un conjunto de datos amplio.
La ecuación, llamada función de costo, que rige este modelo es

$$w^{(i+1)} = w^{(i)} - \alpha * \frac{dJ(w)}{dw}$$ [4]

El proceso de aprendizaje es: los pesos se asignan aleatoriamente en la primera iteración, luego se calcula la función de costo, se actualizan los pesos y se repite este ciclo.
También se encontró que el Alpha más óptimo es de 0.064, dando un error del 10%, el cálculo de este error fue mediante la suma ponderada del error relativo de cada muestra y fue de 10.32%.

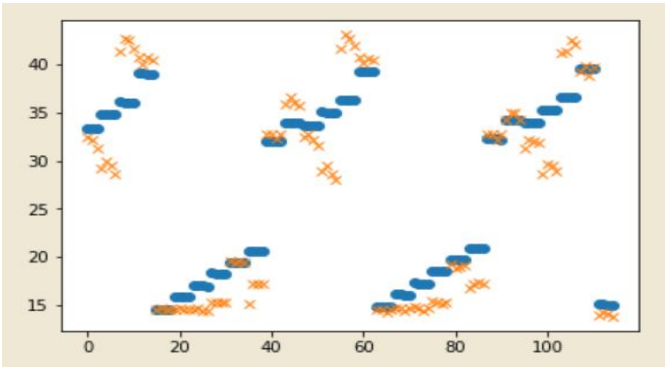


Fig. 3. Regresión Lineal. Los puntos que están representado por X en color azul representan los valores reales, y los O de color naranja representa la predicción.

B. Segundo problema

El conjunto de datos de "letter recignition"el vector de entrada consiste en una serie de atributos de una imagen, se tiene que convertir en un problema de clasificación binaria dividida en las 26 letras del alfabeto ingles en dos clase de la siguiente forma:
Clase 1: letras presentes en el primer apellido de cada uno de los integrantes
Clase 2: El resto de las letras

PERCEPTRÓN.

También conocido una red neuronal de una sola capa (Single-Layer Neural Network), es un algoritmo de clasificación binaria creado por Frank Rosenblatt a partir del modelo neuronal de Warren McCulloch y Walter Pitts desarrollado en 1943.

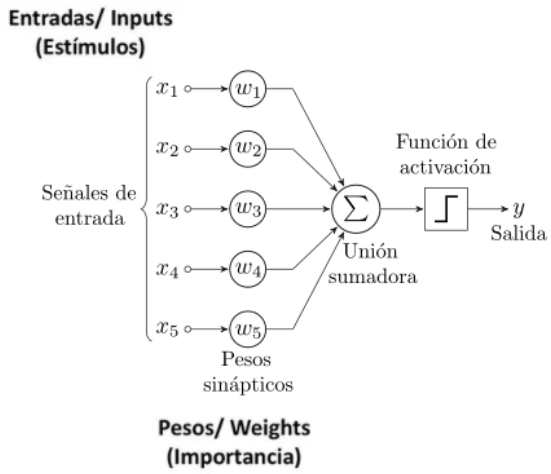


Fig. 4. Representación gráfica de perceptron

La neurona recibe impulsos externos (x) que son considerados con distinta importancia o peso (w) en una función de activación (z). Si el estímulo agregado sobrepasa cierto umbral (θ), la neurona se activa.

Matemáticamente, definimos x como el vector de estímulos y w como el vector de pesos, ambos m dimensiones, y z como la función de activación. El perceptrón ϕ(z) se considera activo cuando su valor es mayor o igual al umbral θ o inactivo en cualquier otro caso.

Se tiene un sistema de entrada normalizado, y un vector de salida de dos clases, una es -1 y otra 1.

Un vector de pesos es multiplicado con el producto punto con el vector de característica de una muestra de nosotros, esto lo hacemos pasar por la función de activación, este valor lo vamos a considerar como un valor predicho y como tenemos el vector de salida, o en especial para este dato conocemos si ha sido etiquetado como de la clase -1 o 1, entonces al hacer la multiplicación consideramos si fue clasificado correcta o incorrectamente, tal que, su representación es la de:

$W^T X_i$ (= Y_p)	Y_i = etiqueta	Y_p Y_i
-1	-1	+(= >0)
-1	1	-(= <0)
1	-1	-(= <0)
1	1	+(= >0)

El código consta de los siguientes pasos.

1. Obteniendo y normalizando dato.
2. Obtenemos la matriz de salida (Valores: Clase1: -1, Clase2: 1; [-1,1]).
3. Dividir la matriz de datos de entrada para entrenamiento, validación y prueba por medio de boostrapping.
4. Definiendo funciones necesarias para implementar el algoritmo perceptrón.
5. Implementando algoritmo Perceptrón para entrenamiento e Implementando validación según eficiencia.
6. Probando entrenamiento y validación.
7. Prueba Final o Testeo - Cálculo de la eficiencia Final.
8. Visualización del resultado.

La eficiencia de este método fue 72.1% en validación y 70.4% en prueba medido según el error relativo ponderado.

REGRESIÓN LOGÍSTICA.

El objetivo de la regresión logística es determinar la existencia o ausencia de relación entre una o más variables independientes (X,) y una variable dependiente dicotómica (Y), es decir, que sólo admite dos categorías que definen opciones o características mutualmente excluyentes u opuestas. [5]

El código se ejecutó de la siguiente forma:

1. Se calcula la función sigmoidal del producto punto del vector peso que se hace aleatoriamente para la primera iteración y la matriz de entrenamiento.
2. Se halla la función de costo que viene dada de la siguiente forma

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

$$P(y=1 \mid x;\theta) = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$ Taken from Prof. Andrew Ng.'s Coursera ML course

3. Se valida el entrenamiento para encontrar alpha óptimo que para este caso fue de 0.0017
4. Se escoje un umbral de decisión para cada muestra que en nuestro ejercicio es de 0.7
5. Se calcula el error de validación, por medio de la suma ponderada de error relativo
6. Se hace la prueba
7. Se calcula la suma ponderada de error relativo

El error relativo para la prueba fue muy alta, de 51,28% esto se debe a la forma en que Alpha, al hacerla variar, hace que cambie mucho el umbral de decisión porque la predicción de los valores cambia el rango en que estos se encuentran.

C. Tercer problema

Se tiene una base de datos con los cantos de especies de animales. Cada muestra corresponde a una vocalización a la cual se le extrajeron 76 descriptores. Se desea obtener un clasificador que permita reconocer cuando entre una nueva muestra, a cuál de las 12 clases esta muestra pertenece.

El desarrollo de este algoritmo se hizo de la siguiente forma:

1. Se extrajeron los datos, y se normalizaron
2. se crea la función de entrenamiento de la red neuronal que tiene como entrada los datos, y valores por defecto como Alpha.
3. Se genera un peso aleatorio y se empieza con el back propagation, que está dada por la siguiente fórmula

$$\delta_i^{(L)} = \sum_{j \in J} \delta_j^{(L+1)} * w_{ji}^L * g'(z_i^{(L-1)})$$

4. Se crea la función de predicción con el forward propagation que es el producto punto entre el peso y la entrada y luego hacer la función sigmoide entre ese resultado.

Las pruebas que se realizaron, se encontraron que tiene una efectividad para varios casos del 80% usando 2 capas.

III. CONCLUSIONES

- Todo modelo de predicción se puede ejecutar bajo, como mínimo, una red neuronal donde tiene sus funciones de entrada, pesos y función de activación.

- Predecir datos o valores es de vital importancia para los sistemas de alerta temprana, como predecir el nivel de contaminación que tendrá Medellín en los próximos días y tomar medidas sobre el pico y placa que tendrán los carros, en el ámbito clínico podría predecir cuándo se puede tener un infarto o saber los niveles de medidas de alguna sustancia química en el cuerpo. También podemos tener detección de errores o anomalías por ejemplo en el procesamiento digital de imágenes usado para los vehículos inteligentes, que ven un objeto extraño en la vía y toman decisiones como frenar o esquivar.

- La emulación del sistema neuronal humano ha hecho que problemas que no sean linealizables o que se puedan operar vectorialmente se puedan hacer mediante iteraciones con ayuda de métodos numéricos.

- El algoritmo LMS tuvo mejor rendimiento que el algoritmo de regresión lineal, esto se debe a que el algoritmo LMS calcula el peso muestra por muestra, lo que hace que pueda tener un mejor ajuste al entrenar, en cambio la regresión lineal hace el cálculo del peso vectorialmente, tomando en cuenta todos los valores a la vez.

- Aunque el algoritmo LMS es más efectivo que la regresión lineal, la gente suele usar más la regresión, ya que LMS tiene un alto costo computacional, ya que el desarrollo del entrenamiento es muestra por muestra, lo que hace que el programa termine de entrenar en un tiempo mayor que el de regresión lineal.

- El algoritmo de regresión logística es más efectivo que el perceptrón, ya que este utiliza funciones logarítmicas para hallar el peso.

- La regresión logística al tener cálculos con logaritmos hace que el programa tenga un gran costo computacional, lo que lo hace más lento.

- La regresión logística, al tratarse de un algoritmo de clasificación, necesita tener mucho conocimiento previo del problema, sobre todo para el momento que se tenga que decidir el umbral de decisión, ya que esta marca el camino para la clasificación.

- En nuestro ejercicio se notó que la regresión logística fue con la de menos efectividad, esto se debe principalmente a que para variaciones muy pequeñas de Alpha el resultado cambiaba abruptamente y se tenían que modificar de nuevo los umbrales de decisión. Al tener este problema se tomó la decisión de buscar un Alpha que, al validar el peso, los valores al sacarle la media estuvieran cerca de 0,5 de tal forma poder variar el umbral de decisión, el que más se acomodó a este ejercicio es 0,7 ya que hacía una clasificación más coherente.

- A la hora de escoger el algoritmo para desarrollar un problema no se puede tener como único criterio la efectividad, también entran variables como el costo computacional, la base de datos que se maneja, y como esta, está organizada.

- El costo computacional es quizás el criterio más importante que se tiene que tener en cuenta, ya que los métodos difieren en pocos porcentajes de efectividad. Para sistemas de tiempo real, generalmente estos algoritmos estarán en sistemas embebidos, ya que los datos provienen de sensores, por lo que hace que el sistema no pueda tener un alto costo computacional.

IV. REFERENCIAS

[1]https://www.dspace.espol.edu.ec/bitstream/123456789/16579/1/Implementaciones%20en%20Matlab%20de%20los%20Algoritmos%20Adaptativos%20para%20los%20Sistemas%20de%20Antenas%20Inteligentes.pdf

[2]http://www.academia.edu/29981791/Algoritmos_LMS_De_Filtrado_Adaptativo_Para_Cancelaci%C3%B3n_De_Eco_Ac%C3%BAstico_en_Sistemas_De_Telecomunicaciones.

[3]tesis.ipn.mx/jspui/bitstream/123456789/3722/1/ALGORITMO%20LMS.pdf

[4]https://github.com/williamegomez/Machine-Learning-Teaching/blob/master/lessons/2018-1/Neural%20Networks/Inteligencia%20Computacional%20-%20Redes%20Neuronales.pdf

[5]http://www.ine.es/ss/Satellite?blobcol=urldata&blobheader=application%2Fpdf&blobheadername1=Content-Disposition&blobheadervalue1=attachment%3B+filename%3D172%2F891%2F141_9.pdf&blobkey=urldata&blobtable=MungoBlobs&blobwhere=172%2F891%2F141_9.pdf&ssbinary=true