INGENIERIA INFORMATICA
Escuela Politécnica Superior
Universidad Autónoma De Madrid

Práctica Neurocomputación

P2

David Teófilo Garitagoitia Romero
Daniel Cerrato Sánchez

18/03/2023

Índice de Contenidos

Lista de Tablas y Figuras

1. Introducción

Práctica 2 de la asignatura de Neurocomputación.

Objetivos: Implementación de una red neuronal de tipo perceptrón multicapa con entrenamiento siguiendo el método de backpropagation y resolver las cuestiones planteadas a continuación.

2. Algoritmo de retropropagación

El algoritmo de retropropagación implementado es el que se han visto en las transparencias de teoría llevadas a una programación en C e intentando hacerlo algo más eficiente.

El algoritmo conlleva un bucle externo que solo para cuando una bandera de entrenamiento deja de estar activa o si se ha llegado al número de épocas indicado por el usuario. Este bucle simula las épocas de entrenamiento.

Al inicio de cada época, la bandera se desactiva para que, al final de esta, se vuelva a activar si no se cumple la condición de parada referente a la tolerancia. Más adelante extenderemos esto.

Durante cada época, se recorren los patrones de entrenamiento uno a uno en otro bucle.

Para cada patrón, se inicializa la capa de entrada con los valores de este. Luego, se procede a hacer la fase de *feedforward*, que dispara las neuronas de las capas y propaga el resultado hacia la siguiente capa, desde la capa de entrada hasta la capa de salida. En este caso, no reiniciamos los valores de entrada de las neuronas porque los necesitamos para la retropropagación.

El siguiente paso es calcular los deltas de cada neurona. Primero, se calculan los deltas de la capa de salida. Cada neurona tiene su delta correspondiente que se calcula de la siguiente manera:

$$\delta_k = (t_k - y_k) f'(y\_in_k)$$

Para las neuronas de las capas ocultas, se calculan los deltas menos para las neuronas de sesgo. Estas neuronas conllevan dos cálculos: Primero, se calculan los deltas de entrada de la siguiente forma:

$$\delta\_in_j = \Sigma w_{jk} \delta_k$$

Es decir, para cada neurona oculta, se multiplican sus pesos hacia las neuronas de la siguiente capa por los deltas de esas neuronas; y se suman todos los valores. Después, para calcular el delta de las neuronas ocultas se lleva a cabo el siguiente cálculo:

$$\delta_j = \delta\_in_j * f'(z\_in_j)$$

Estos deltas se van añadiendo a vector de deltas. Como hacemos *push_back* de los deltas de las capas finales hacia las primeras, las capas de deltas acaban en orden inverso, por eso es esencial llevar un buen manejo de este apunte. Al final de estos dos bucles de cálculo de deltas para la capa de salida y de las ocultas, se reinician los valores de entrada de las neuronas.

Ahora toca actualizar los pesos, para ello recorremos las capas desde la inicial hasta la última oculta, realizando el siguiente cálculo según el tipo de neurona:

Si no es de sesgo: $w_{ij} = w_{ij} + \alpha * \delta_j * x_i$

Si es de sesgo: $w_{0j} = w_{0j} + \alpha * \delta_j$

Finalmente, se comprueba si el error cuadrático de la clasificación del patrón es menor a una cierta tolerancia dada por el usuario. Si es así, la bandera se deja inactiva, sino, se activa. Esto permite conocer si, en algún patrón, el error es aún demasiado grande y se debe seguir entrenando.

NOTA: No hemos conseguido que las redes aprendan debidamente, no hemos conseguido encontrar el error en el entrenamiento y, por tanto, no hemos realizado los siguientes apartados, puesto que no podíamos contrastar resultados fiables.

Aún así, para el apartado de problemas reales y normalización, se ha contestado ya que conocemos el procedimiento de normalización de datos.

3. Problemas reales

4. Problemas reales y normalización

La normalización de los datos se realiza siguiendo un algoritmo sencillo.

Para cada atributo, se calcula la media y la desviación típica usando todos los datos disponibles. Después de calcular estos datos en un atributo, se realiza una pasada por cada valor, actualizándolo con este cálculo:

$$x' = \frac{x - \mu}{\sigma}$$

Donde *x* es el valor original, *x'* es el valor actualizado, μ es la media y σ es la desviación típica.

Esta transformación de los datos es común hacerla cuando hay atributos con rangos de valores muy dispares. Si permitimos estos rangos, pasa que los cálculos que se realizan en las neuronas están desbalanceados, ya que los valores más altos tienen mayor impacto en el valor de entrada de las neuronas; lo que se refleja en una mayor importancia del atributo, cuando esto puede no ser cierto o correcto.

Para evitar este problema, se llevan todos los valores al mismo rango mediante la normalización, típicamente de 0 a 1 si se quiere en binario o de -1 a 1 si se quiere en bipolar. De esta forma, todos los atributos empiezan con una importancia semejante y permite a la red determinar qué atributos son más determinantes a la hora de clasificar.

[FINAL DE DOCUMENTO]