TP1 Redes Neuronales

Suárez, Germán y Suárez Barón, Juan Carlos

Mayo 2019

Introducción

En este trabajo práctico se desarrolló un perceptrón de una capa oculta con una arquitectura a elección del usuario. Este perceptrón se aplicó a datos de dos problemas, uno de clasificación y otro de regresión. El primero consiste en diagnosticar si un tumor era benigno o maligno basado en atributos del mismo. El segundo problema, consiste en determinar las cargas para calefaccionar o refrigerar edificios de acuerdo a características arquitectónicas de los mismos.

1. Modos de uso del programa

El código del programa consiste en dos *scripts*, el *script perceptron_multicapa.py* en el cual se implementó un perceptrón multicapa de una capa oculta, y el *script* principal $TP1.py$, desde el cual se cargan los datos, se reescalan, se separan en datos de entrenamiento (80 %), validación (10 %) y testeo (10 %), y se les aplica el perceptrón desarrollado en el *script* anterior.
En el primero, para implementar la red multicapa se creó una clase llamada $PerceptronMulticapa$, la cual se inicializó con los argumentos $DimTrain$, que corresponde a la cantidad de entradas que tendrá la red, *neurons*, que corresponde a la cantidad de neuronas que tendrá la capa oculta, *outputs*, que corresponde a la cantidad de salidas que tendrá la red, y *beta*, que corresponde a la constante que incorpora la función sigmoidea, que se utiliza como función de activación, y que por defecto se le asignó el valor 1. Con estos argumentos, se crearon dos matrices $w1\_$ y $w2\_$, con valores inicializados de manera aleatoria entre -1 y 1. Los elementos de estas matrices corresponden a los pesos que unen las entradas con las neuronas de la capa oculta, y a los pesos que unen éstas últimas con las salidas, respectivamente. Además se crearon los vectores $v\_$ que contiene los valores que toman las neuronas de la capa oculta, $output\_$ que contiene los valores que toman las salidas, y $delta1\_$ y $delta2\_$, que son vectores que utilizan más adelante para la corrección de los pesos. Luego se creó la función $forward$, la cual se encargará de la etapa hacia adelante y, usando la función de activación, calcula la salida a partir de un determinado patrón de entrada. Esta función recibe como parametros un booleano, llamado *redEntrenada*, si es falso usará los pesos creados aleatoriamente en el paso anterior. Si es verdadero recibirá como pesos dos matrices $w1$ y $w2$ que se pasen como argumento. La función *backward* simplemente se encarga de corregir los pesos mediante *backpropagation*. La función *train* recibe como parámetros datos de entrenamiento y datos de validación. En esta función se recibe un número entero, correspondiente a la cantidad de épocas. Por cada época se entrena a la red llamando a la función $forward$ y *backward*, calculando y almacenando los errores entrenamiento y, posteriormente, los de validación. También en cada época se irá guardando los pesos para los cuales se obtiene, desde la época 1 hasta la época n, el menor error de validación. Estos pesos se utilizarán para calcular el error de *testing*.
En cuanto al *scaling* de los datos, se decidió utilizar *estandarización*, es decir, se les restó la media y dividió por su desvío estándar, para los atributos del conjunto de datos de ambos ejercicios. En el caso de las observaciones de la variable respuesta del primer ejercicio, se decidió convertir las 'M' a 0,95 (a modo de 1) y las 'B' a 0.05 (a modo de 0) ya que la sigmoidea no alcanza los valores 1 y 0, salvo en el límite. En el caso de las variables de respuesta del segundo ejercicio, se utilizó la normalización $min - max$, el cual deja los datos en el intervalo [0,1]. Luego se le aplicó una leve modificación que deja los datos en el intervalo [0.05,0.95], por la misma razón descrita anteriormente.

2. Resultados

A continuación se presentan los resultados de evaluación de los dos datasets.

2.1. Diagnóstico de cáncer de mamas

Luego de 1500 épocas de entrenamiento, y con un *learning rate* de 0.1, obtuvimos el siguiente gráfico de error para cada época:

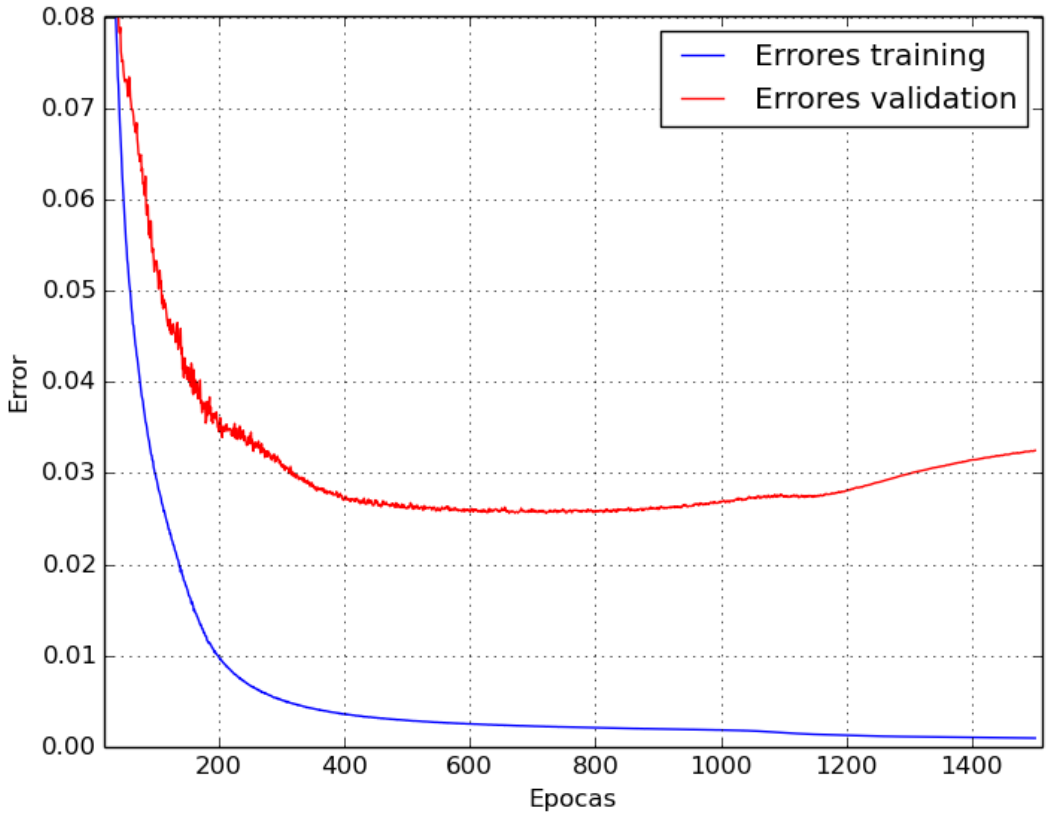

Figura 1: Error vs Épocas para el 1er conjunto de datos

En el gráfico puede apreciarse que entre las épocas 600 y 800, la red empieza a sobreentrenar y el error de validación comienza a crecer, es decir, la red aprende demasiado de los datos de entrenamiento y predice cada vez peor los datos de validación.

La matriz de confusión, que permite la visualización del desempeño de la red con los pesos de menor error de validación, se presenta a continuación:

	Maligno	**Benigno**
Maligno	13	2
Benigno	1	25

Cuadro 1: Matriz de confusión

Tasa de aciertos

La tasa de aciertos (*Accuracy* ó *Exactitud*) se define como la cantidad de aciertos dividido la cantidad total de datos de validación. En este caso se obtuvo:

$$Accuracy = \frac{tp + tn}{tn + tp + fn + fp} = \frac{13 + 25}{13 + 25 + 1 + 2} = \frac{38}{41} \approx 0,927$$

donde:

- tp : $true\ positive$
- tn : $true\ negative$
- fp : $false\ positive$
- fn : $false\ negative$

Precisión

La precisión se define como la cantidad de *true positive* divido por la suma de *true positive's* y *false positive's*. En este caso se obtuvo:

$$Precision = \frac{tp}{tp + fp} = \frac{13}{13 + 1} = \frac{13}{14} \approx 0,929$$

Recall

La exhaustividad o *recall* se define como la cantidad de *true positive* divido por la suma de *true positive's* y *false negative's*. En este caso se obtuvo:

$$Recall = \frac{tp}{tp + fn} = \frac{13}{13 + 2} = \frac{13}{15} \approx 0,867$$

2.2. Eficiencia energética

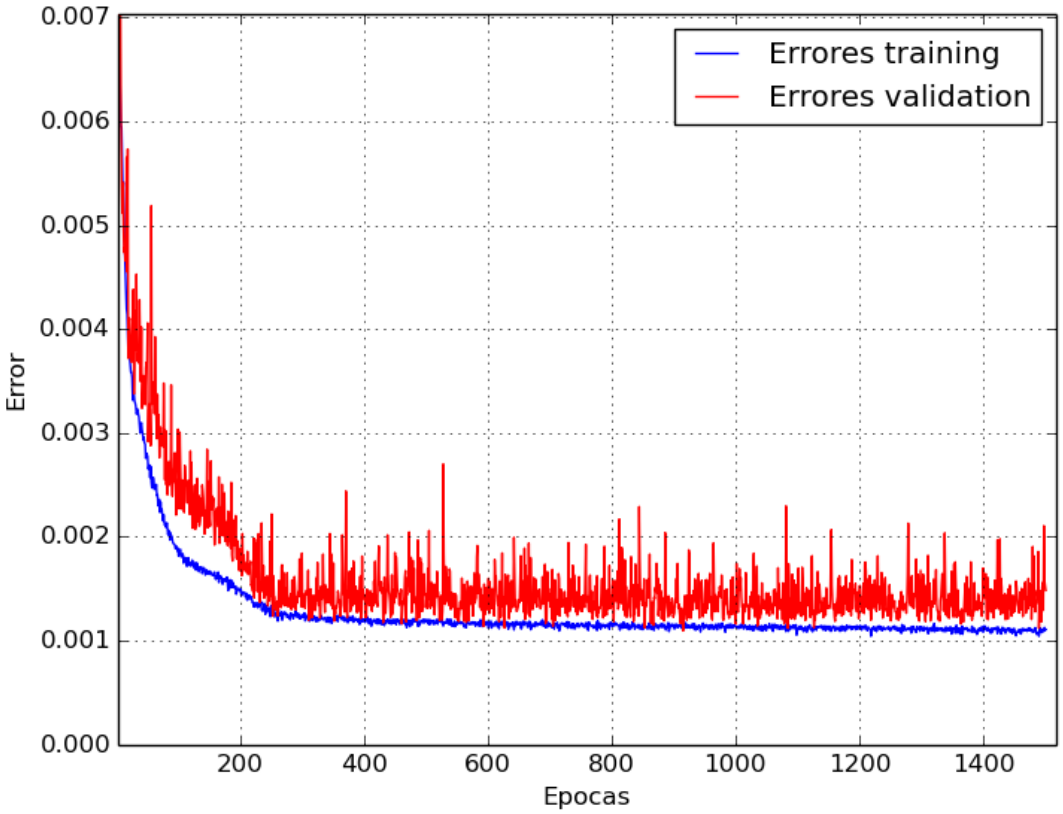

Figura 2: Error vs Épocas para el 2do conjunto de datos

En este caso, se obtuvo un error de *testing* de aproximadamente 0.0568. Igual que en el caso de la clasificación, comienza a verse un sobreentrenamiento a partir de, aproximadamente, la época 300, y a partir de ahí el error de validación deja de decrecer.

3. Conclusiones

El perceptrón multicapa es una herramienta útil que permite analizar datos y obtener modelos sin hacer suposiciones de las mediciones del *dataset*, por ejemplo, no se tuvo en cuenta la distribución que tenían éstas. Sin embargo, la principal desventaja del perceptrón multicapa es el tiempo que consume encontrar la arquitectura y parametros adecuados para obtener el mejor modelo.

El perceptrón de una capa oculta desarrollado para este trabajo permitió obtener resultados de exactitud y de precisión de 0.927 y 0.929, respectivamente, para la parte de clasificación (diagnóstico de cáncer de mámas) y un error de testeo de 0.0568 para la regresión (eficiencia energética). Estos resultados muestran un desempeño aceptable del modelo de red neuronal desarrollado.