Redes Neuronales

Gonzalo V. Castiglione, Alan E. Karpovsky, Martín Sturla
Estudiantes Instituto Tecnológico de Buenos Aires (ITBA)

19 de Abril de 2012

Entrega Preliminar 3 - Informe

***Resumen*—El presente informe busca analizar redes neuronales multicapa con aprendizaje supervisado que resuelvan el problema del *Simetría* y *Paridad* lógica para N bits con $2 \leq N \leq 5$ a través del uso de tres variantes de funciones de transferencia.**

***Palabras clave*—perceptrón, función de transferencia, red neuronal, aprendizaje supervisado, conjunto de entrenamiento, conjunto de testeo**

I. Introducción

Se analizó el comportamiento de distintas redes neuronales multicapa para el problema de la generalización de una función matemática a partir de un conjunto de datos (puntos de la función). Con este fin se implementó un algoritmo que permite definir la arquitectura y las propiedades de la red neuronal a generar de manera de hacer más simple y práctico su estudio.

Asimismo se optó por escribir funciones que permitan tomar subconjuntos de forma aleatoria del conjunto de muestras de entrada, dado un determinado porcentaje. Esto es verdaderamente util si se quiere probar el poder de generalización de la red: Se toma, por ejemplo, un conjunto de entrenamiento con el 70 % de los valores de entrada y el 30 % restabte de los puntos se asignan al conjunto de testeo.

II. Desarrollo

A. Modelado del problema

Se representó la red neuronal como una **matriz de pesos**. Cada neurona es una columna de pesos, cada capa de neuronas es una matriz de pesos, la red neuronal, por consiguiente, es un vector de matrices. Lo interesante es que hallar la salida de la red neuronal con una cierta entrada se reduce a multiplicar el vector entrada por cada una de estas matrices. Cabe destacar que al vector y a todos los pasos intermedios se les agrega siempre un -1 como último valor para el sesgo.

Se decidió utilizar el **error cuadrático medio**.

19 de Abril, 2012.

B. Mejoras

Se implementaron tres tipos de funciones para modificar la variable η *(learn rate)*: La primera de ellas es a valor **constante**, , la segunda es ***annealed*** que reduce η exponencialmente y por último se tiene un *learning rate* **adaptative** que modifica η en función de los últimos errores obtenidos. El crecimiento es aritmético, con cota en $\eta = 0{,}5$, la reducción es exponencial.

Debido a la existencia de mínimos locales se desarrolló un algoritmo ***persistent search*** que dado un cierto error, entrena la red neuronal la cantidad de épocas especificada y verificar si el error obtenido es mayor al deseado. En cuyo caso reinicia el algoritmo, asigna los pesos al azar y vuelve a comenzar el entrenamiento.

Otra estrategia que se utilizó para subsanar este problema es la de darle un impulso a η al detectar que el algoritmo está estancado en un mínimo local, haciendo que los valores de los pesos cambien significativamente en un paso (sería análogo a agregar poco ruido a todas la matrices).

Funciones de activación

Para todos los casos de prueba se entrenó a la red utilizando como función de activación la función tangente hiperbólica

$$g(x) = tanh(x) \tag{1}$$

y la función exponencial

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

C. Normalización de datos

En este problema en particular, los valores de las muestras de entrada $f(x, y) = z$ tenían las siguientes características: Aproximadamente $x \in [-3, 3]$, $y \in [-3, 3]$ y $z \in [0, 1]$. Como se comentó anteriormente, la red neuronal puede funcionar con la función de activación tangente hiperbólica en todas sus capas o con la función exponencial. En el caso de optar por utilizar la primera de ellas, la imagen de la función tangente hiperbólica es $(-1, 1)$ y debido a esto se optó por normalizar los datos x e y y llevarlos al $(-1, 1)$. Para hacer esto se dividió al vector $\vec{x}$ por el máximo número en valor absoluto y se realizó el procedimiento análogo para el vector $\vec{y}$ ($\vec{x}$ contiene todos

los puntos x de las muestras de entrada e $\vec{y}$ los puntos y).

Asimismo, teniendo en cuenta que z toma valores entre 0 y 1, se decidió utilizar la función de activación exponencial en la capa de salida. Como cada capa es independiente de las demás, pueden utilizarse distintas funciones de activación en cada capa (lógicamente utilizando la misma función para todas las neuronas de una misma capa). Realizando esta pequeña modificación subsanamos el problema de comparar valores que están entre $(-1, 1)$ (salida de la red neuronal con función de activación tangente hiperbólica en la última capa) con valores de z que están entre $(0, 1)$.

D. Ruptura de simetría

A diferencia de los problemas tratados anteriormente como ser *AND*, *OR*, *Simetría* y *Paridad*, el problema del aprendizaje de una función desconocida no es simétrico. Debido a esta cualidad del problema, lo que se hizo fue agregar conexiones rotas/muertas en la primer capa. Estas conexiones tienen peso 0 y nos permiten diferenciar los dos inputs, ya que no necesariamente $f(x, y) = f(y, x)$.

III. Resultados

IV. Conclusión

Anexo A: Gráficos

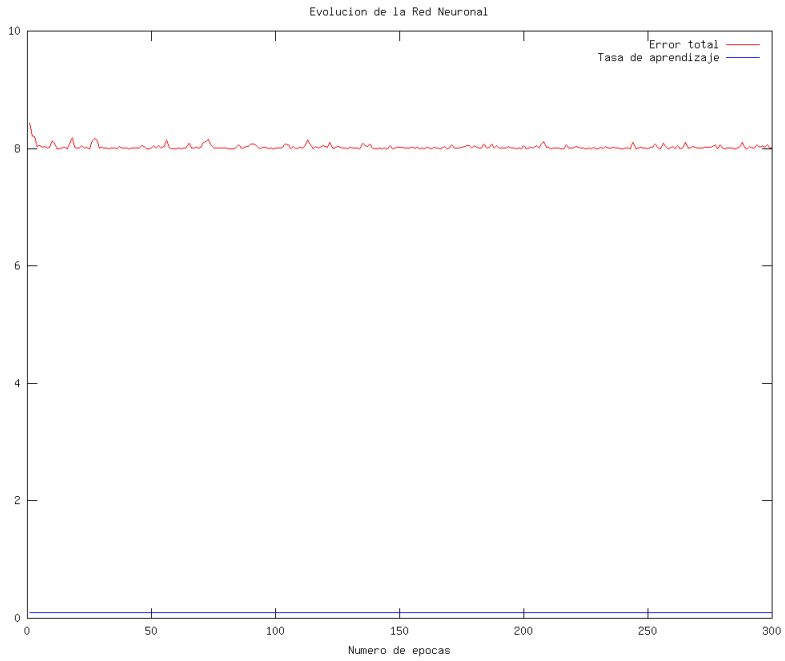

Figura 1: Ejemplo de estancamiento en mínimo local debido a la utilización de una función de transferencia lineal.

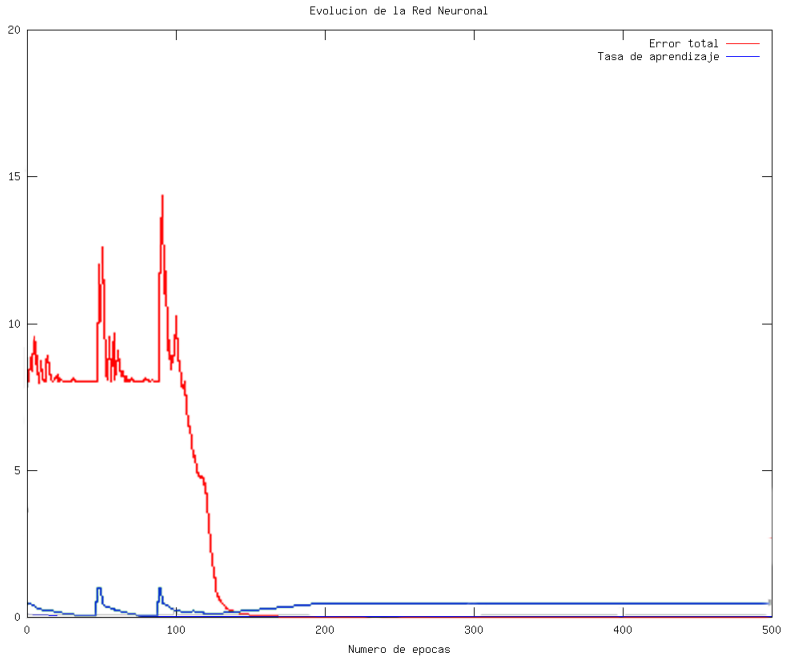

Figura 2: Escape de mínimo local utilizando eta dinámico para el problema de paridad de 3 bits usando 500 épocas y una arquitectura de una capa oculta con 3 neuronas.

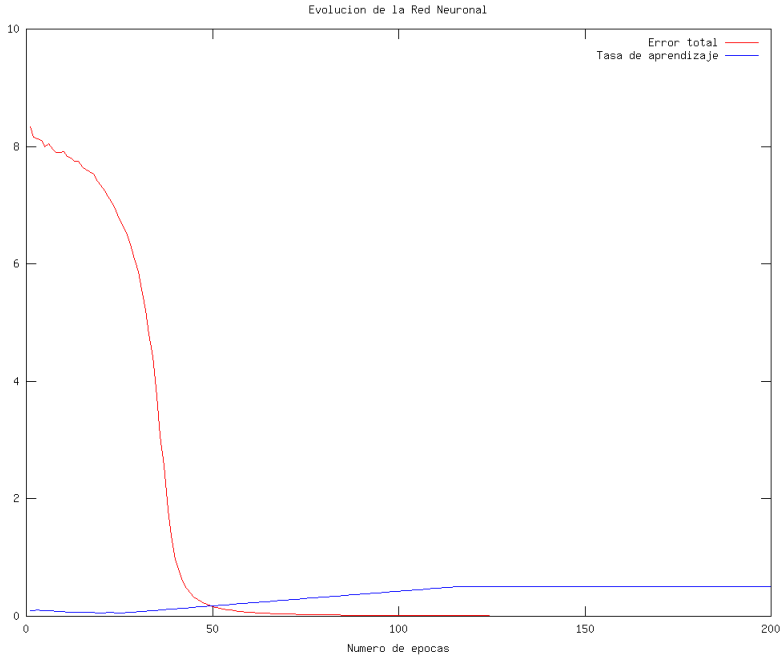

Figura 3: Muestra cómo se incrementa hasta el valor máximo eta cuando la red neuronal resuelve el problema con el error deseado.