Redes Neuronales

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

19 de Abril de 2012

Entrega Final - Informe

***Resumen*—El presente informe busca analizar redes neuronales multicapa utilizando funciones de activacion exponencial y tangente hiperbolica. Para la resolucion de del problema dado.**

***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 % restante de los puntos se asignan al conjunto de testeo (siempre se toman disjuntos).

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.

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 verifica 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 la función de activación la función tangente hiperbólica

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

y la función exponencial

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

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 debieron normalizar los valores de 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).

19 de Abril, 2012.

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 rotas tienen peso 0 y permiten diferenciar los dos inputs, ya que no necesariamente $f(x, y) = f(y, x)$.

III. Cálculo del error

Para los gráficos se optó por mostrar la **diferencia cuadrática media** calculada como sigue:

$$\sum_i \frac{(S_i - O_i)^2}{N} \tag{3}$$

En cuanto al error, se calcula el **error promedio** de la siguiente forma:

$$e = \sum_i \frac{|S_i - O_i|}{N} \tag{4}$$

Entonces, esto nos está diciendo, por ejemplo que, si esta sumatoria es igual a 0.01, cada valor tiene un error promedio de 0.01.

Por ultimo calculamos la **desviación del error**:

$$\sqrt{\sum_i \frac{((\|S_i - O_i\| - e))^2}{N}} \tag{5}$$

Si, por ejemplo, el desvío es igual a 0, significará que todos los valores tienen un error de 0.01. Es decir el error es fijo. Si es alto significa que hay algunos valores muy precisos (error casi 0) y otros con valor muy grande. Es decir el error es variado.

IV. Resultados

De todas formas, los resultados obtenidos no pueden ser descartados y/o tomados como óptimos o absulutos. Esto se debe a que el problema presenta numerosas variables que pueden incidir en el resultado del aprendizaje de una red neuronal: los distintos valores que pueden tomar β y η, la arquitectura utilizada, la cantidad de épocas, la presencia de ruido aleatorio, las conexiones muertas para romper simetría, etc.

Además, debido a la aleatoriedad de la elección de los puntos a utilizar en el conjunto de entrenamiento y la inicialización de los pesos de a red al azar, no podemos dar por descartadas malos resultados ya que es posible que, por ejemplo, se hayan seleccionado puntos poco representativos de la función o que los pesos no se hayan inicializado de la forma óptima.

Una forma de subsanar esta problemática sería correr numerosas veces cada caso de testeo antes de poder sacar una conclusión pero debido a la complejidad temporal del problema, realizar esto requeriría más tiempo del disponible.

V. Conclusión

Luego de realizadas las pruebas y analizado los resultados, se pudo observar lo siguiente:

- Para la resolucion de problemas no siempre es necesaria una mayor cantidad de neuronas para lograr una precisión aceptable (puede llevar a malas generalizaciones y tiempo de mas hasta alcanzar el error desdeado).
- El aprendizaje de las redes y la velocidad de convergencia dependen exclusivamente de la naturaleza del problema y de los parámetros adecuados para dicho problema en particular.
- Las mejoras al algoritmo de back propagation, no garantizan mejoras en la velocidad de convergencia para todos los problemas.
- Muchos patrones de entrenamiento pueden perjudicar la capacidad de la red de generalizar, sin embargo muy pocos pueden provocar un aprendizaje que no es el esperado.

Anexo A: Gráficos

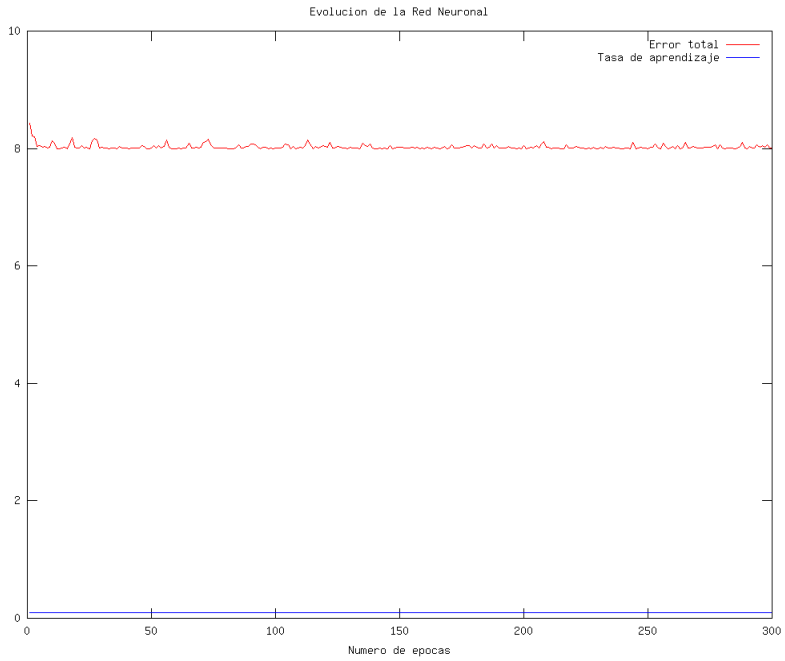

Figura 1: Ejemplo de estancamiento en mínimo local debido a la utilización de una función de transferencia lineal.

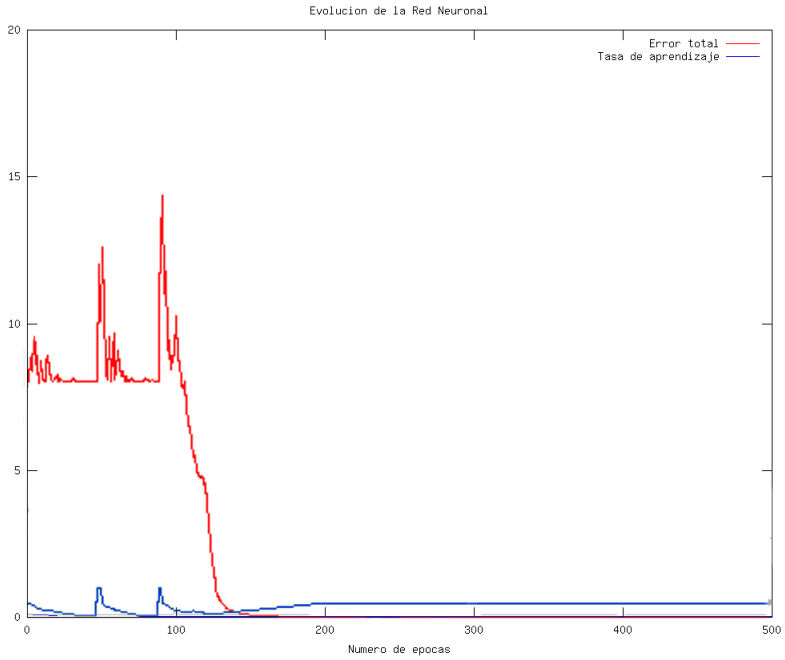

Figura 2: Escape de mínimo local utilizando eta dinámico para el problema de paridad de 3 bits usando 500 épocas y una arquitectura de una capa oculta con 3 neuronas.

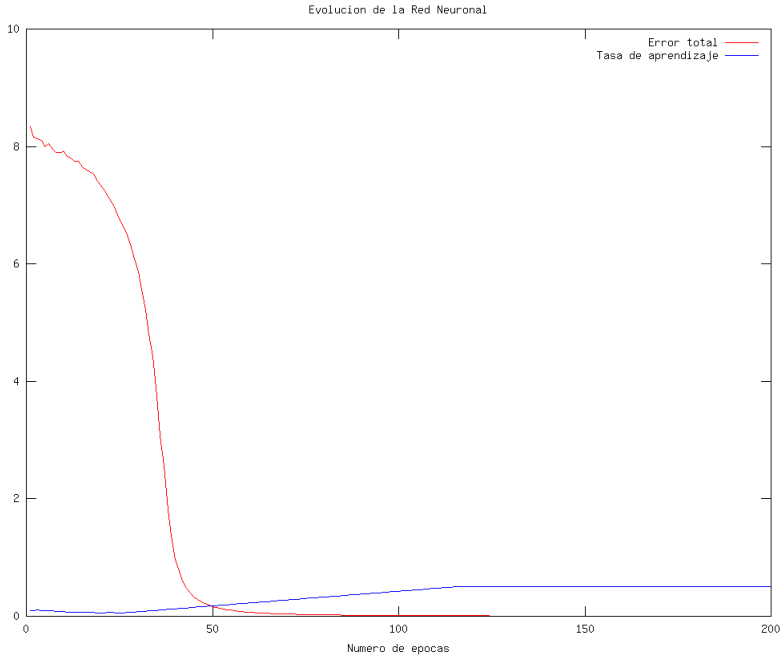

Figura 3: Muestra cómo se incrementa hasta el valor máximo eta cuando la red neuronal resuelve el problema con el error deseado.