Aprendizaje Supervisado en Redes Neuronales Multicapa

Sistemas de Inteligencia Artifical - ITBA

Carlos Sessa Lucas Pizzagalli Nicolás Purita

Introducción

Se implementó una red neuronal multicapa supervisada para resolver una función con la siguiente forma:

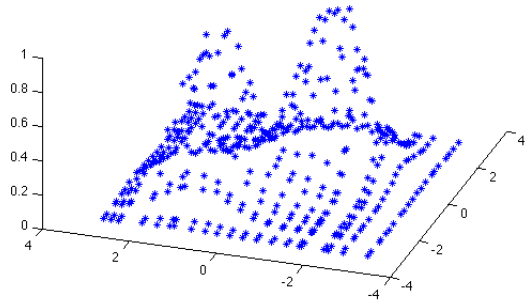

Figura 1: Distribución de puntos dada

En la figura 1 se puede observar que los puntos de la entrada pertenecen al intervalo $[-3.5, 3.5]$ y la salida se encuentra en el intervalo $(0, 1)$.
La implementación de la red fue realizada en *Matlab*. Se corren distintos experimentos, variando las parámetros y arquitecturas a fin de optimizar la generalización del problema. También se implementan dos mejoras al algoritmo para verificar aumentar la capacidad de la red de generalizar en menor tiempo y en forma óptima.

Desarrollo y Problemas encontrados

Los pesos se ajustan utilizando el algoritmo de *Backpropagation* y los patrones son presentados a la red en orden aleatorio durante el aprendizaje. Como condición de parada se considera una cota superior para el error (0.01) y a la vez una cota para el número de épocas (350) para el entrenamiento. Se realizaron varias corridas modificando los parámetros para analizar correctamente el problema. Teniendo en cuenta como los parámetros afectan la eficacia y performance de la red, eligiendo finalmente el que proporciona la mejor solución.

Un tema a destacar es la división y selección de los puntos de entradas, estos son divididos en *Puntos de entrenamiento* y *Puntos de Testeo*. Los puntos de entradas son cargados desde un archivo en donde se encuentran exactamente 441 puntos de la función, distribuidos uniformemente sobre el dominio. El criterio con el que se seleccionan los patrones es el siguiente, el 80% es tomado para el grupo de *Patrones de entrenamiento* y el 20% restante corresponden al *Puntos de Testeo*. Dado que los puntos dentro del archivo se encontraban distribuidos en forma aleatoria se crean los dos subconjuntos utilizando un random.

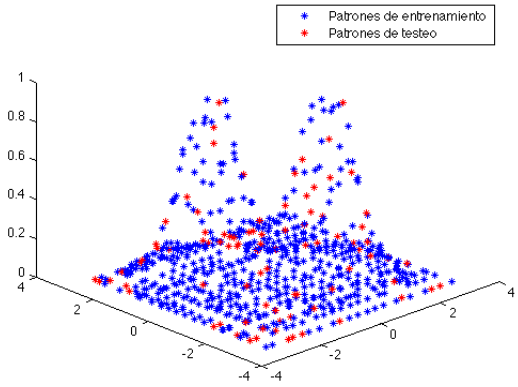

Figura 2: Distribución de puntos de entrenamiento y puntos de testeo.

En la figura 2 se puede ver los dos subconjuntos, el de *entrenamiento* y *testeo*. De esta forma verificamos las distribución de los puntos y podemos deducir que el conjunto de *testeo* es bueno[1].
Durante el desarrollo y en la búsqueda de la mejor arquitectura surgió el problema de que habían muchos puntos sobre la superficie de la función por lo que decidimos tomar todos los puntos de la superficie que cumplan la condición de $z \geq 0.01$. De esta forma no tenemos tanta densidad de puntos sobre la superficie. En la figura 3 se puede observar la nueva distribución de puntos para entrenar la red.

Un tema a destacar es la utilización de la función de activación $g(x) = tanh(\beta x)$ en la última capa. Sucede que el intervalo de salida de la función dada pertenece al $(0, 1)$ por ende se dedujo que sería más apropiado poner una función *exponencial* o *lineal* en la última capa, a comparación de la función *tangencial* que tiene su imagen en el intervalo

[1] Un conjunto de puntos es **bueno** si cubre la mayor superficie de la función.

Sessa, Carlos

Pizzagalli, Lucas

NICOLAS PURITA

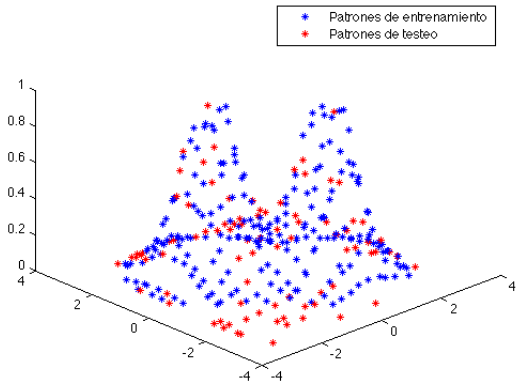

Figura 3: Distribución de puntos tomando los mayores a 0.01

$(-1, 1)$.

Los pesos iniciales se generan de forma aleatoria en el intervalo $[-0.5, 0.5]$. Por último, es necesario mencionar que el orden en que se le dan los patrones a la red durante el entrenamiento es aleatorio.

Resultados y Conclusiones

Para el $\eta - adaptativo$ se realizaron muchas variantes de los parámetros para ir ajustando los valores. Estos valores no son presentados en el informe ya que se deciden en función de la performance y comportamiento de la red. La variable de incremento del η es fija, esto quiere decir que se incrementa en 0.01 cada vez que sea necesario y se decrementa en un 5% del valor del η actual. Para el *Momentum*, luego de probar distintos valores, se tomó el mejor valor como $\alpha = 0.1$.
Para comparar la red entre distintas arquitecturas, se le enseña a la red con los mismos pesos iniciales y en un lapso de tiempo de 350 épocas. Se consideran en todas las arquitecturas 2 neuronas de entradas y 1 de salida.

Arquitectura	Error
[30 20 10]	0.0290183
[10 10]	0.0462676
[10]	1.65359
[10 10 10 10]	0.05552
[40 20]	0.115901
[10 10 10]	0.081405
[5 10 20]	0.405984

Tabla 1: Tabla de comparaciones entre distintas arquitecturas

Las arquitecturas mostradas en la tabla 1 poseen en las capas ocultas la función de activación *exponencial* y en la última capa tiene configurado para utilizar una función de activación *lineal*. La función de activación *exponencial* es configurada con un $\beta = 2$, este valor se obtuvo mediante pruebas y verificando la performance de la red.

η	*Momentum*	Error entrenamiento	Error testeo
No	No	0.0304268	0.0949303
Sí	No	0.0441066	0.0902171
No	Sí	0.0325069	0.0839944
Sí	Sí	0.0290183	0.0762365

Tabla 2: Mejor configuración con y sin mejoras

La configuración de la fila 1 de la tabla 1, se observa que es la red con menor error de entrenamiento obtenido. Con el objetivo de visualizar mejor que representa dicho error, se ha generado la figura 6. En dicha figura se puede apreciar que la red ha aprendido bastante ya que la relación entre valor esperado y valor obtenido por la red se asemejan mucho en la mayoría de los casos. Los puntos generan una figura que es muy similar a una recta (la recta es el caso ideal, en el que los datos obtenidos son iguales a los reales). De todos modos se aprecia que al cabo de 350 épocas existen algunos patrones que aún no fueron aprendidos, en particular los patrones de los picos de la función. Una forma de mejorar el aprendizaje es modificando los puntos de tal forma de que haya mas concentración de puntos de entrenamiento en los picos. Luego de 350 épocas se obtiene un error de entrenamiento de 0.0290183.

En la figura 5 se puede ver como evolucionan los errores a medida que transcurre el tiempo. A simple vista se observa como al principio el error de los patrones de testeo es menor a los patrones de entrenamiento. Esto se debe a que todavía la red no ajustó los pesos lo suficiente pero a medida que la misma aprende lo suficiente se ve como el error de entrenamiento desciende por debajo del error de testeo.

En la Tabla 2 se puede observar como el $\eta - adaptativo$ y el *Momentum* mejoran la performance y el conocimiento de la red.

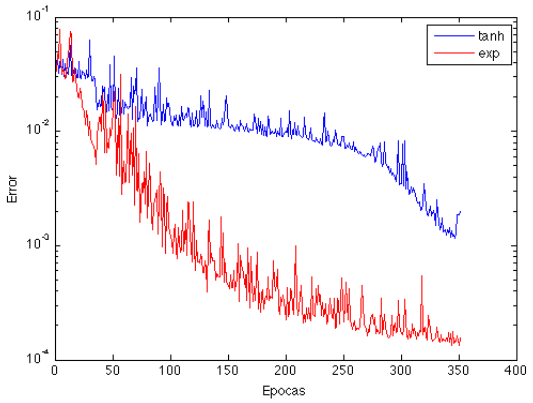

Figura 4: Comparación entre la tangencial y exponencial

En la figura 4 se puede ver como, utilizando la *exponencial* como función de activación, en las capas ocultas se logra un mejor resultado que utilizando la *tangencial*.

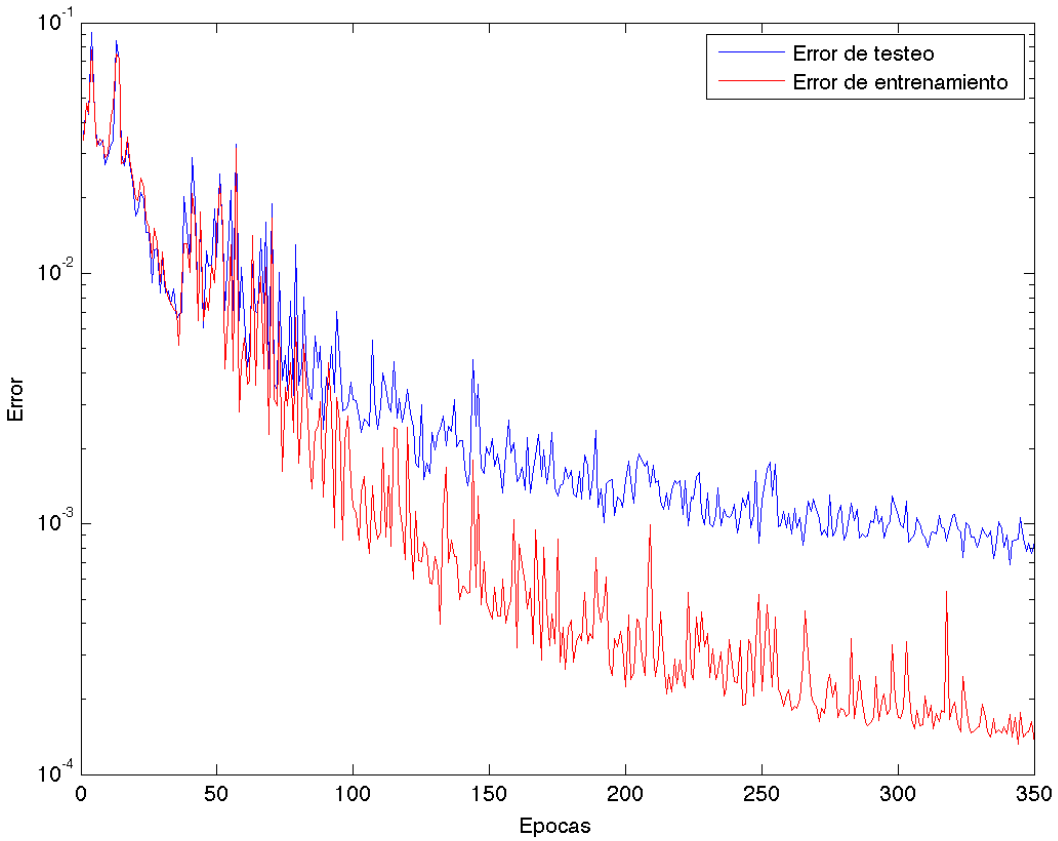

Figura 5: Evolución de error de testeo y entrenamiento durante el entrenamiento

Sessa, Carlos

Pizzagalli, Lucas

NICOLAS PURITA

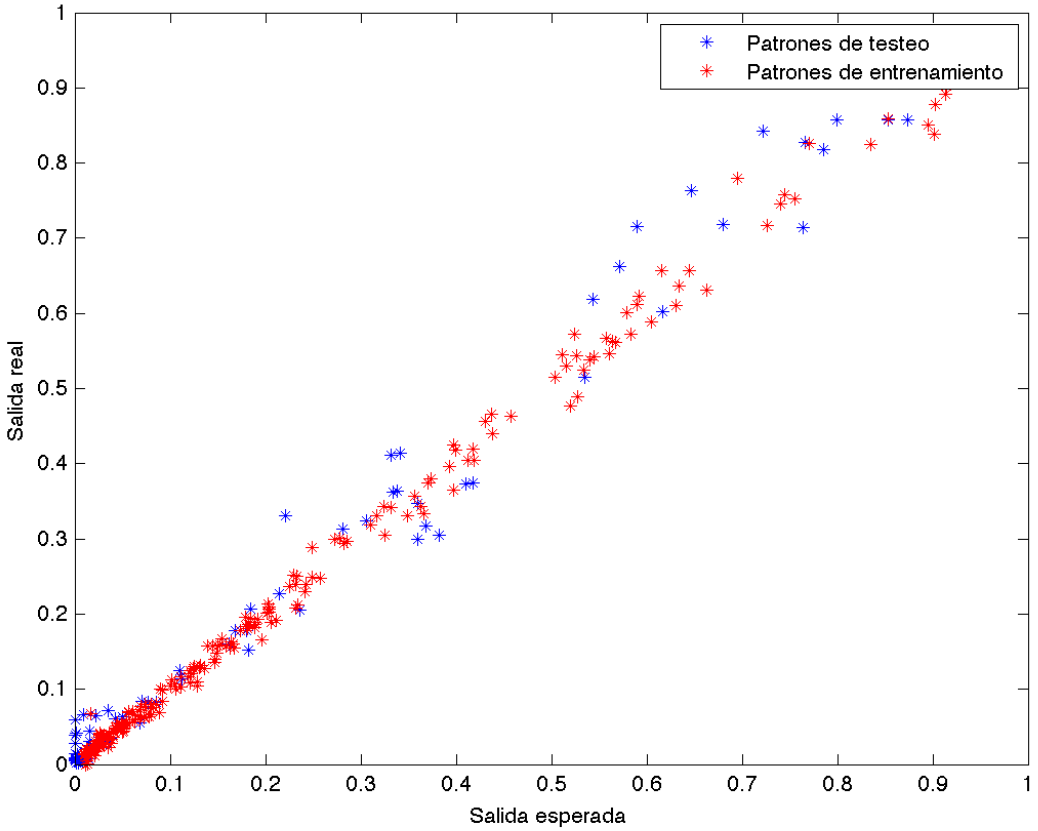

Figura 6: Comparación entre salidas esperadas y obtenidas

Sessa, Carlos

Pizzagalli, Lucas

NICOLAS PURITA