

Sistemas de Inteligencia Artificial

Trabajo práctico especial 2

Grupo 5

Julián E. Gutiérrez F. (51141)

Alexis Medvedeff (50066)

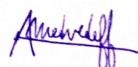
Javier Perez Cuñarro (49729)

ITBA

Primer Cuatrimestre 2014



JULIÁN E. GUTIÉRREZ F.



ALEXIS MEDVEDEFF



JAVIER PEREZ CUÑARRO

Introducción

El objetivo de este trabajo es implementar una red neuronal multicapa con aprendizaje supervisado que pueda ser usada para resolver un problema en particular.

Una red neuronal es un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. La red reciben una serie de entradas y mediante interconexiones emite una salida. Esta salida depende de la función de propagación, la función de activación y la función de transferencia elegidas.

En particular, un perceptrón multicapa es una red neuronal compuesta por una serie de capas: la capa de entrada, las capas ocultas y la capa de salida.

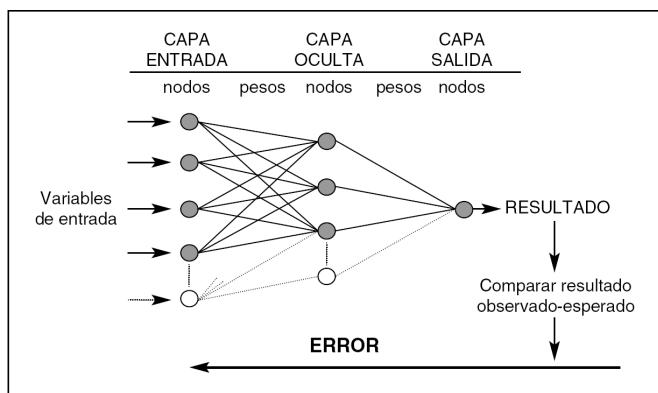


Figura 1: esquema del perceptrón multicapa¹

Para este trabajo, usamos una red neuronal para estimar una función escalar ($f: R^2 \rightarrow R$) descripta por un conjunto de puntos asignado por la cátedra. Definimos un perceptrón multicapa con sus funciones correspondientes y evaluamos los resultados obtenidos comparándolos con los valores esperados.

También comparamos el comportamiento de distintas funciones de activación y arquitecturas de red e implementamos dos mejoras al algoritmo de *backpropagation* para resolver el problema de forma más óptima.

Desarrollo

El perceptrón multicapa fue desarrollado íntegramente con la herramienta libre Octave².

Para modelar el perceptrón usamos una estructura de Octave que nos permite almacenar la información correspondiente a cada nivel del árbol. Como los índices para los arreglos de esta

¹ Esquema perteneciente al artículo

<http://www.medintensiva.org/es/redes-neuronales-artificiales-medicina-intensiva-/articulo/13071859/>

² Versión 3.8, descargado de <https://www.gnu.org/software/octave/>

JULIÁN C. GUTIÉRREZ F.

ALEXIS MEDVEDEFF

JAVIER PÉREZ WÍÑARRO

herramienta comienzan en 1, decidimos considerar lo que en matemática sería el nivel 0, es decir las entradas, como el nivel 1. Para cada nivel de este árbol se almacenan los pesos de las aristas de la siguiente manera:

Se usa una matriz que por cada fila contendrá las conexiones entre el nodo i, contado de izquierda a derecha y todos los nodos del piso inmediato inferior.

Adicionalmente se almacenan los valores de la suma pesada que ingresó a los nodos de ese nivel y los delta que se usan para la corrección de los pesos

Puntos de entrada

Trabajamos con los datos de la muestra número ocho³ que contiene 441 puntos tridimensionales. Las siguientes figuras ilustran la dispersión de estos puntos dentro del sistema cartesiano tridimensional.

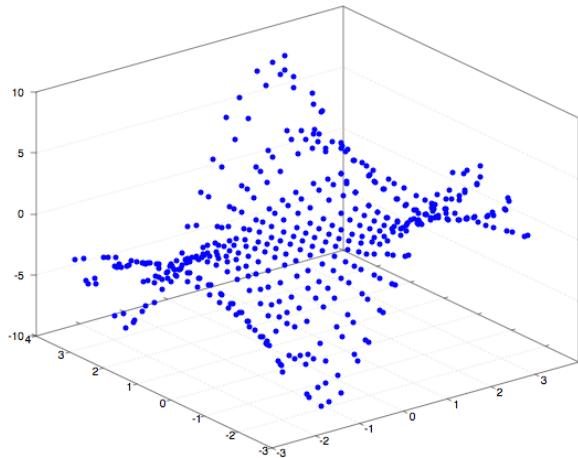


Figura 2⁴: datos de la muestra

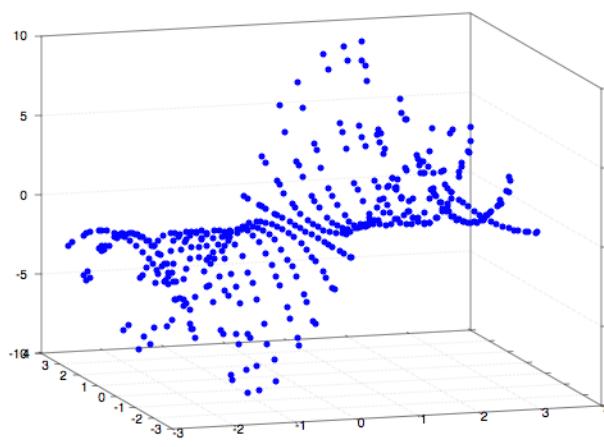


Figura 3: datos de la muestra (vista alternativa)

Analizando el gráfico, notamos que los puntos de entrada se encuentran en el intervalo (-2.9,3.1) para las componentes x e y, y la salida pertenece al intervalo (-9.4,9.4).

Dividimos los puntos en un subconjunto de datos para usar en la fase de aprendizaje del perceptrón y otro subconjunto para usar en la fase de testeo (la unión de ambos resulta en el conjunto inicial y la intersección es vacía). El porcentaje usado para la división de datos puede ser configurado en el asistente interactivo por el usuario. Los valores se dividen en los grupos de forma aleatoria.

³ Ver archivo samples8.txt (dentro de TPE2-samples.zip provisto por la cátedra, o en nuestro repositorio)

⁴ x, y entradas del sistema. z salida esperada.

JULIÁN C. GUTIÉRREZ

ALEXIS MEDVEDEFF

JAVIER PÉREZ WÍÑARRO

Para este informe se seleccionó en la mayoría de los casos un 80% de datos para entrenamiento y el 20% restante para testeo. Esto es porque si el muestreo de entrenamiento no es lo suficientemente significativo, la red generalizará mal los valores de testeo. Alguna de las causas por la que esto podría ocurrir es porque se pueden estar obviando máximos o dando un subconjunto de datos que represente una región de la función, dejando de lado la otra. En el anexo se muestra la tabla 1 con los resultados obtenidos utilizando distintos tamaños para las dos muestras. En los gráficos 1 y 3 del anexo se muestra la evolución del error cuadrático a lo largo de las 500 épocas. En los gráficos 2 y 4 se muestra un histograma de la diferencia entre los valores calculados por la red entrenada y los valores esperados utilizando los patrones de testeo. Se ve claramente cómo la mayor cantidad de puntos tiene un error bajo.

Funciones de activación

Entrenamos a la red haciendo uso de dos funciones de activación: la función tangente hiperbólica $f(x)$ y la función sigmoidea exponencial $g(x)$.

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

En el caso de la tangente hiperbólica, tomamos Beta en 1, mientras que en la exponencial tomamos Beta igual a 2 en la mayoría de los casos. Se muestra en la tabla 2 que, al menos con las pruebas realizadas, el cambio de los valores no resulta demasiado significativo. Es posible que en ciertas arquitecturas un valor dado pueda resultar conveniente sobre otro pero no es posible garantizar esto. La selección de estos valores se produce en forma empírica y con ellos se busca distinguir de la muestra los valores para la activación de la neurona de forma tal que active correctamente la sinapsis cuando corresponda.

Parametros especiales

Se definen los siguientes parámetros especiales:

Epsilon: se empleará para determinar si un valor calculado por la red se considera “igual” a su correspondiente salida esperada.

Eta inicial: es el valor inicial para el parámetro que refuerza las conexiones de la red. Mientras más pequeño sea menor será la modificación en los pesos de la red.

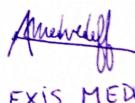
Alpha: este parámetro se emplea para la mejora *momentum*, que se explicará en la próxima sección.

Max epoch: se define como época a una iteración en la red sobre todos los patrones de entrenamiento. Teniendo esto en cuenta notamos que una posible herramienta de corte para la etapa de entrenamiento sería un número fijo de épocas. Este parámetro fija dicho máximo para la fase de entrenamiento.

Min learning rate: este parámetro será otro de los mecanismos de corte para la etapa de aprendizaje de la red. Si en una época la cantidad de aciertos supera a ese porcentaje del total,



JULIÁN C. GUTIÉRREZ



ALEXIS MEDVEDEFF



JAVIER PÉREZ WÁRRRO

decimos que la red está lo suficientemente entrenada, con lo cual cesa el entrenamiento sin importar si se alcanzó el máximo de épocas definido anteriormente.

Salvo que se indique lo contrario, tomamos los siguientes valores para dichos parámetros. Epsilon: 0.001, Eta inicial: 0.01, Max epoch: 500 , Alpha:0.9, Min learning rate: 0.7 Cota.

Mejoras al backpropagation

El algoritmo de backpropagation es un algoritmo costoso en términos computacionales, por lo cual se intenta aplicar mejoras para que el aprendizaje de la red se acelere y de este modo se realicen la menor cantidad de operaciones posibles durante esa fase. Las mejoras implementadas son las siguientes:

- Momentum: en las primeras iteraciones el error decrece significativamente. En otras palabras la red se encuentra aprendiendo mucho. El momentum, asociado al parámetro alpha, hará que la suma correctiva sea más grande en las primeras iteraciones y luego se “apagará” cuando la diferencia entre 2 pasos casi no disminuya. Se debe tener cuidado de elegir un alpha adecuado, ya que de otro modo podría ocurrir que la función se encuentre trabajando en una región cercana a un mínimo o máximo de la función y al “sumar de más” se pierda esta información resultando en una generalización incorrecta posteriormente.
- ETA “adaptativo”: el término asociado a ETA se encuentra dentro del cálculo de la corrección de los pesos y refuerza las conexiones en las neuronas. En general se trabaja con un ETA fijo, pero en esta mejora el valor se corrige dinámicamente, agregamos una cota máxima de 0.5a la función para asegurarnos que no disparara el error.

Redes entrenadas

Una vez que consideramos que la red está entrenada se genera automáticamente un archivo que contiene toda la información asociada a la red entrenada. Posteriormente el usuario puede acceder al mismo para testear otros datos o para continuar con el entrenamiento.

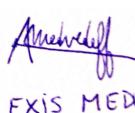
Resultados⁵

Evaluamos el efecto de cambiar el porcentaje de valores usados para entrenamiento vs testeo para una red neuronal con una arquitectura de 10, 20 y 30 neuronas en sus tres capas internas (de aquí en más, “[10 20 30”]), función de activación tangente hiperbólica y observamos 500 épocas. La tabla 1 muestra que con un porcentaje más bajo de patrones de entrenamiento los resultados fueron mejores que con un porcentaje mayor. Esto probablemente se debe a que los puntos que se tomaron aleatoriamente fueron más representativos en el caso del 40%.

En la tabla 2, comparamos los resultados de aplicar distintas funciones de activación con β variable. Tomamos una arquitectura de [10 20 30], datos de entrenamiento 80%, 100 épocas.

⁵ Obtenidos con procesador Intel Core i5 de 2.6 Ghz bajo Mac OSX 10.9.2.


JULIÁN C. GUTIÉRREZ


ALEXIS MEDVEDEFF


JAVIER PÉREZ MUÑOZ

Por otro lado evaluamos el comportamiento de la red usando distintas arquitecturas, es decir, variando la cantidad de capas y de neuronas por cada capa. Usamos la función de activación tangente hiperbólica, el 80% de los datos como datos de entrenamiento y 500 épocas. Se puede ver en la tabla 3 que la mejor arquitectura entre las probadas es [10 20 30]. Es posible que existan otras mejores para lo cual se necesitan realizar otras pruebas.

Para analizar el funcionamiento de las mejoras al algoritmo, presentamos las tablas 4 y 5 variando las condiciones de uso de los mismos. En el caso del momentum encontramos que entre los alpha probados, el mejor para esta arquitectura resultó ser 0.9. En el caso de eta adaptativo conviene utilizar saltos cada K=5 épocas y con una variación sobre el valor inicial del 25% para el incremento y 2.5% en el caso del decremento. El valor de alpha fuere obtenido con la arquitectura mencionada anteriormente y para 100 épocas, mientras que los demás se calcularon en función de la precisión que se desea obtener ($< 10^{-3}$) y los efectos que se observaba en los gráficos.

Conclusiones

Luego de probar distintas configuraciones posibles del perceptrón multicapa, hicimos un análisis comparativo de los indicadores obtenidos.

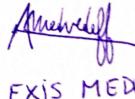
La decisión de la cantidad de neuronas y capas no es decidible de forma teórica. Es necesario realizar pruebas para ver si realmente conviene una forma o no por sobre otra. Lo mismo sucede con los distintos parámetros como puede ser el momentum o el eta y los parámetros de adaptación del mismo.

La selección de la arquitectura y los parámetros depende de las condiciones del problema.

Utilizar la función de activación exponencial puede llevar en algunos casos a estancarse en mínimos.



JULIÁN C. GUTIÉRREZ F.



ALEXIS MEDVEDEFF



JAVIER PÉREZ WÍÑARRO

Anexo A

Tablas de resultados

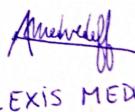
Porcentaje de entrenamiento	Porcentaje de testeo	Error cuadrático medio, promedio en testeo	Porcentaje de aciertos con error menor a 10^{-3}
40%	60%	0.8478%	48.6013%
60%	40%	0.9522%	44.9101%
80%	20%	0.7181%	41.1764%

Tabla 1: comparación del error de la red neuronal en relación al porcentaje de los datos usados para entrenar. Arquitectura [10 20 30], sin mejoras, tangente hiperbólica, máx 500 épocas.

Función de activación	Error cuadrático medio, promedio en testeo	Porcentaje de aciertos con error menor a 10^{-3}
tanh $\beta=1$	1.2850%	28.6863%
tanh $\beta=0.5$	1.4006%	17.6470%
exp $\beta=1$	1.3994%	32.6921%
exp $\beta=2$	0.7794%	35.3346%

Tabla 2: comparación del error de la red neuronal en relación a la función de activación usada. Arquitectura [10 20 30], sin mejoras, 80% de entrenamiento, máx 100 épocas.


JULIÁN C. GUTIÉRREZ F.


ALEXIS MEDVEDEFF


JAVIER PÉREZ WÍÑARRO

Arquitectura	Error cuadrático medio, promedio en testeo	Porcentaje de aciertos con error menor a 10^{-3}
[20]	1.2924%	36.7647%
[15 15]	0.9339%	30.8823%
[10 20 30]	0.7181%	41.1764%
[2 4 8 16]	1.5001%	25.0000%

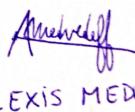
Tabla 3: comparación del error de la red neuronal en relación su arquitectura. Sin mejoras, 80% de entrenamiento, tangente hiperbólica, máx 500 épocas.

Mejoras aplicadas	Error cuadrático medio, promedio en testeo	Porcentaje de aciertos con error menor a 10^{-3}
Ninguna	1.5060%	26.4705%
Momentum ($\alpha=0.9$)	1.6598%	26.4705%
Momentum ($\alpha=0.1$)	1.5405%	26.4705%
η adaptativo	1.5187%	17.6470%
Momentum ($\alpha = 0.9$) + η adaptativo	2.2648%	8.8235%
Momentum ($\alpha = 0.1$) + n adapt	1.6079%	19.1176%

Tabla 4: comparación del error de la red neuronal en relación a las mejoras aplicadas a backpropagation. Arquitectura [10 20 30], 80% de entrenamiento, tangente hiperbólica, máx 100 épocas.



JULIÁN C. GUTIÉRREZ



ALEXIS MEDVEDEFF

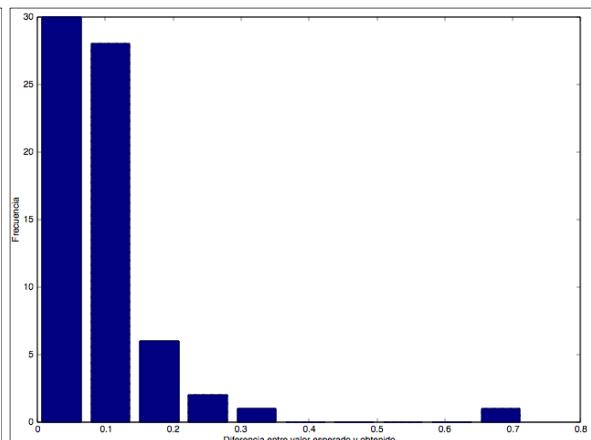
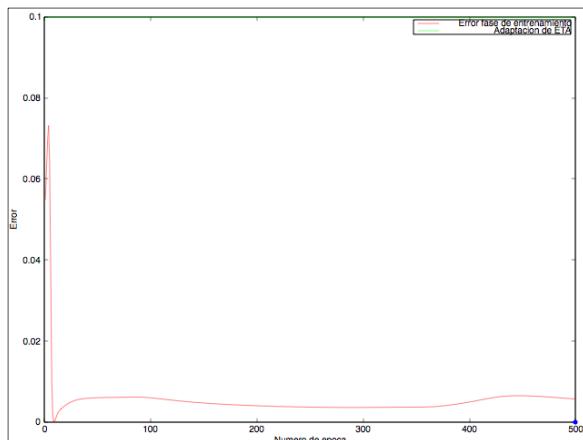


JAVIER PÉREZ MUÑOZ

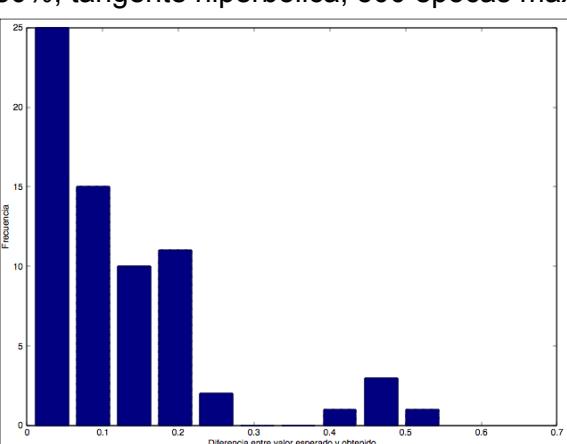
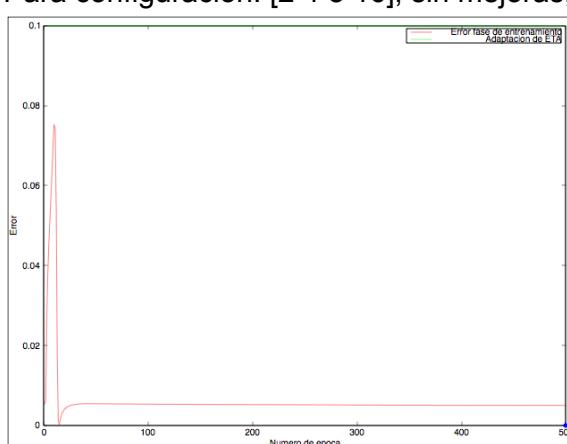
Anexo B

Gráficos

Para configuración: [15 15], sin mejoras, 80%, tangente hiperbólica, 500 épocas máx.



Para configuración: [2 4 8 16], sin mejoras, 80%, tangente hiperbólica, 500 épocas máx.



JULIÁN C. GUTIÉRREZ

ALEXIS MEDVEDEFF

JAVIER PÉREZ WÍÑARRO

Para configuración: [10 20 30], con ambas mejoras ($\alpha = 0.9$), 80%, tangente hiperbólica, 100 épocas máx.

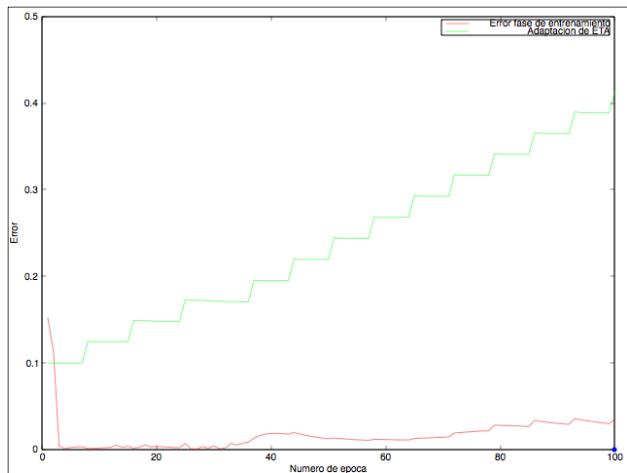


Gráfico 5: error cuadrático por época

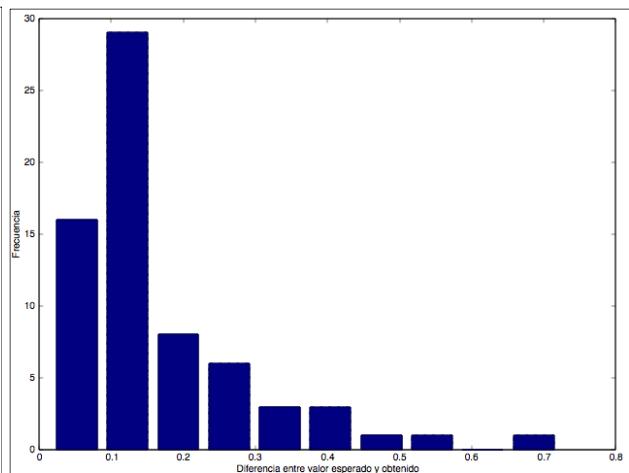


Gráfico 6: diferencia valor calculado y esperado

JULIÁN C. GUTIÉRREZ

ALEXIS MEDVEDEFF

JAVIER PÉREZ VÁZQUEZ