Sistemas de Inteligencia Artificial

Redes Neuronales

Predicción de series de tiempo

José Ignacio Galindo (50211), Federico Homovc (50418), Nicolás Loreti (49479)

*Instituto Tecnológico de Buenos Aires (ITBA)*

*24 de Abril 2013*

**Palabras clave - Artificial Neural Network, backpropagation, momentum, parámetros adaptativos, Time Series prediction**

# 1 - Introducción

Este trabajo describe la implementación de un programa para generar redes neuronales capaces de predecir el comportamiento de una serie de tiempo usando entrenamiento de forma supervisada. Para el entrenamiento de las redes se usó backpropagation con dos variaciones: Momentum y Parámetros adaptativos.

# 2 - Descripción del problema

El problema consiste en una serie de puntos ordenados cronológicamente dados como dato de entrada, de los cuales se debe poder predecir el comportamiento. Se supone que los datos corresponden a un fenómeno que puede ser predicho.

# 3 - Resolución

## 3.1 - Modelado

Para el modelado del problema se utilizaron matrices con números decimales en Octave/Matlab. Cada matriz representa los pesos de las conexiones entre dos capas consecutivas más un peso extra para el umbral. Para obtener la salida de una capa de la red, simplemente se multiplica la salida de la capa anterior por la matriz de pesos correspondiente y se le aplica una función de transferencia (o ). Este proceso se realiza capa por capa hasta obtener la salida final del sistema.

La red es entrenada utilizando un conjunto de parámetros de entrada con su respectiva salida esperada. Todos los parámetros de dicho conjunto son usados una vez en orden aleatorio o secuencial de acuerdo a lo elegido por el usuario, y luego se compara el error cuadrático medio entre la salida obtenida y la esperada contra un error máximo; en caso de ser el error cuadrático medio menor que el esperado se termina el algoritmo. En caso de que se prueben todos los parámetros del grupo de entrenamiento dos veces seguidas y la diferencia entre el error cuadrático en ambos casos sea menor que 10-10, se termina el algoritmo ya que se asume que se cayó en un mínimo local.

## 3.2 - Backpropagation

Una vez calculada la salida final del sistema *o*, se compara la misma con la salida esperada *s* y la diferencia entre ambas es usada para corregir los pesos entre la última capa y la de salida; a esta diferencia se la denomina *delta inicial* . A partir de este valor, se calculan los deltas restantes como donde *h* es el potencial de membrana, *g* es la función de activación, *W* es la matriz de pesos entre la capa actual y la siguiente, y es el vector de deltas de la capa anterior. Luego de calcular los deltas se procede a actualizar los pesos en las matrices de transición con la siguiente fórmula: donde , es el factor de aprendizaje y V el vector de salidas de la capa anterior.

### 3.2.1 - Momentum

Una variación a backpropagation que se puede usar para aumentar la velocidad de convergencia hacia la solución deseada es llamada *Momentum*. Básicamente, consiste en agregar un término a la corrección del error para que mida el descenso promedio, de la forma con . De esta forma, los pesos de la red tienden a mantener la convergencia hacia una solución y no se ven afectados por pequeños cambios en el espacio de soluciones.

### 

### 3.2.2 - Parámetros adaptativos

Otra forma de mejorar la convergencia de los pesos de la red neuronal hacia la salida deseada es ir modificando el factor de aprendizaje eta luego de iterar por todos los patrones de entrenamiento. Esto se puede hacer de dos maneras: decrementando eta progresivamente cada cierto número fijo de iteraciones (annealed) o ir modificandolo de acuerdo a como varía el error cuadrático medio (dinámico).

En el primer caso se toma como parámetro una cantidad de pasos y un factor de reducción y cada vez que se completa esa cantidad de pasos se decrementa eta en dicho factor. En el segundo, se toma en cuenta el error cuadrático medio de las últimas *n* iteraciones; si el error aumenta con respecto al paso anterior se decrementa el factor de aprendizaje una proporción determinada y se pone el alpha del momentum en cero (si es que la mejora de momentum está activada). Si en cambio el error disminuye una cantidad de veces de forma consecutiva, a partir de ese momento eta aumenta en una cantidad fija en cada iteración mientras el error no aumente.

# 4 - Análisis de resultados

Al momento de probar la red neuronal, hay que tener en cuenta todos los factores que pueden variar, tanto la arquitectura de la misma como los distintos parámetros que recibe: épocas, alpha del Momentum, orden de los patrones de aprendizaje, eta (factor de aprendizaje inicial), factor de disminución de eta, factor de crecimiento de eta, límite del contador para empezar a incrementar eta y la Función de transferencia.

**Entrenamiento Secuencial y Aleatorio**

Al momento de mezclar los patrones en la fase de entrenamiento, sucede que al principio del entrenamiento el error cuadrático medio oscila más cuando se mezclan en cada época que cuando no. Al final del entrenamiento se puede observar que no hay una mejora significativa al desordenar los patrones en la fase de entrenamiento (Tabla 1).

**Configuración Ideal**

Los mejores resultados se obtuvieron con una red con 3 entradas y 3 capas ocultas, de 9, 6 y 4 neuronas correspondientemente. Se puede observar en la Tabla 2 que el error cuadrático medio se hace más chico cuanto más neuronas y más capas ocultas se agreguen. Este resultado se condice con la teoría, que afirma que mientra más neuronas se agreguen en las capas ocultas más fácil será para la red aprender los patrones que se le dan en la fase de aprendizaje, pero, por otro lado, se dificulta su generalización. Con respecto a la generalización, se puede ver que en la fase de pruebas se obtuvieron mejores resultados con la red que obtuvo menor error cuadrático medio en la fase de aprendizaje. El agregar más capas y neuronas hizo posible un aprendizaje con menor error, pero sin llegar a hacer *overfitting.*

**Funciones de transferencia**

No se pudo concluir que alguna de las dos funciones de transferencia estudiadas funcione mejor que la otra, ya que esto último depende de la configuración y de la matriz de pesos iniciales (la cual se genera aleatoriamente). A lo largo de las diferentes pruebas no se identificaron grandes diferencias.

Lo que sí se pudo observar es que los valores de beta jugaban un valor importante al momento de entrenar la red con una u otra función, ya que algunos funcionan muy bien para la tangente hiperbólica,por ejemplo, y otros no.

Si bien el error cuadrático medio al testear ambas funciones es similar, el desvío estándar en la exponencial es mucho mayor en la fase de prueba, lo que implica que la exponencial no generaliza tan bien como la tangente hiperbólica. Esto se debe a que dicha función aproxima muy bien muchos valores pero presenta un error mucho más grande que la tangente hiperbólica en muchos otros (en la Tabla 4 puede verse como por lo general el error máximo es mayor en la exponencial y el error mínimo es menor en dicha función).

Lo anterior puede también verse en las figuras 1 y 2 del anexo. Si se observa el histograma de errores de las mismas, puede verse como hay muchas más ocurrencias del error cerca del cero en la tangente hiperbólica que en la exponencial, lo que indica que la desviación estándar es mucho mayor en la exponencial.

**Mejoras a *backpropagation***

Los mejores resultados se obtuvieron usando un factor de aprendizaje adaptativo de tipo dinámico. Esto se puede ver reflejado en la *Tabla 2*, la cual muestra los mejores resultados para cada configuración.

A su vez, como se puede ver en la *Tabla 3* -que muestra una comparación entre la performance de la red con configuración 3,5,1 (3 entradas y una capa oculta con 5 neuronas) utilizando las mejoras a backpropagation y sin ellas-, que el menor error cuadrático medio en la fase de aprendizaje se obtuvo usando un factor de aprendizaje dinámico. Luego, el segundo mejor error fue el correspondiente al factor de aprendizaje dinámico de tipo *annealed,* seguido de momentum. El error cuadrático medio de la misma red sin usar ninguna de las variantes de backpropagation fue el mayor, lo que indica que estas últimas realmente mejoran la performance general aunque no de manera significativa. Esto se puede ver al observar el que el error de las 4 pruebas tienen el mismo orden de magnitud.

# 5 - Conclusiones

* Luego de estudiar distintas configuraciones se puede ver que agregar más neuronas a la red en general facilita el aprendizaje de los patrones de entrenamiento, siempre y cuando se encuentren valores adecuados para los demás parámetros.
* En cuanto a los parámetros para el aprendizaje del problema, como conclusión se puede decir que el valor de los mismos depende tanto del problema en sí, como de la configuración y también de la función de transferencia que se esté usando. Esto quiere decir que no existe, por ejemplo, un valor de beta o un factor de aprendizaje que funcionen de forma óptima siempre, sino que están fuertemente determinados por la naturaleza del problema y de la red.
* La mejora a backpropagation que mejor funciona para el problema estudiado es el factor de aprendizaje variable de forma dinámica.
* Al momento de entrenar a la red, tomar una cantidad escasa de puntos de la serie puede dar como resultado un entrenamiento pobre, caso contrario tomando casi todos los puntos puede volver a la red muy rígida a ese conjunto de entrenamiento y no dar respuesta frente a otras entradas o conjuntos de pruebas.
* Las funciones de transferencia no presentan una diferencia significativa con respecto al error cuadrático medio obtenido para la fase de aprendizaje.
* La tangente hiperbólica provee una mejor generalización (con un mismo error en la fase de aprendizaje obtiene un error menor en la fase de prueba).

6 - Anexo

Resultados

Tabla 1: Comparación entre ordenamiento de los patrones en cada época

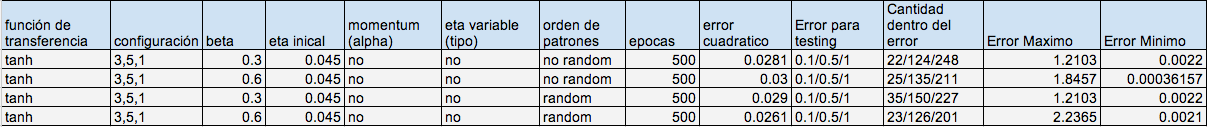


Tabla 2: Mejores resultados para cada configuración

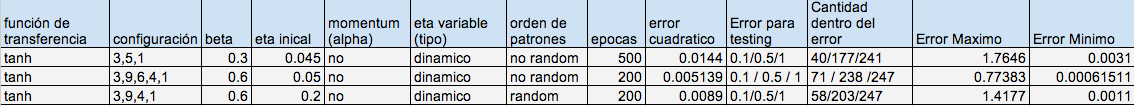


Tabla 3: Comparación con tangente entre annealed, momentum y dinámico

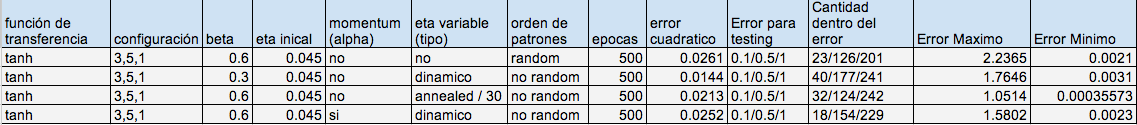


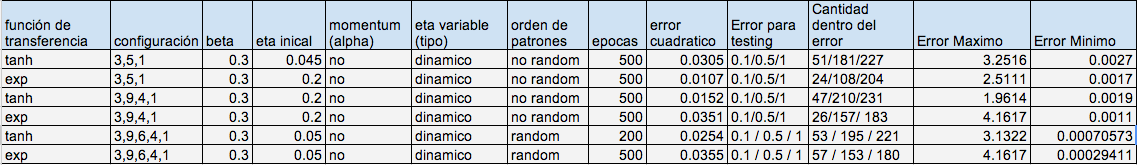
Tabla 4: Comparación entre función de transferencia exponencial y tangente hiperbólica (usando factor de aprendizaje dinámico dinámico) 

Figura 1 : Fase de prueba con configuración 3,5,1 y función exponencial

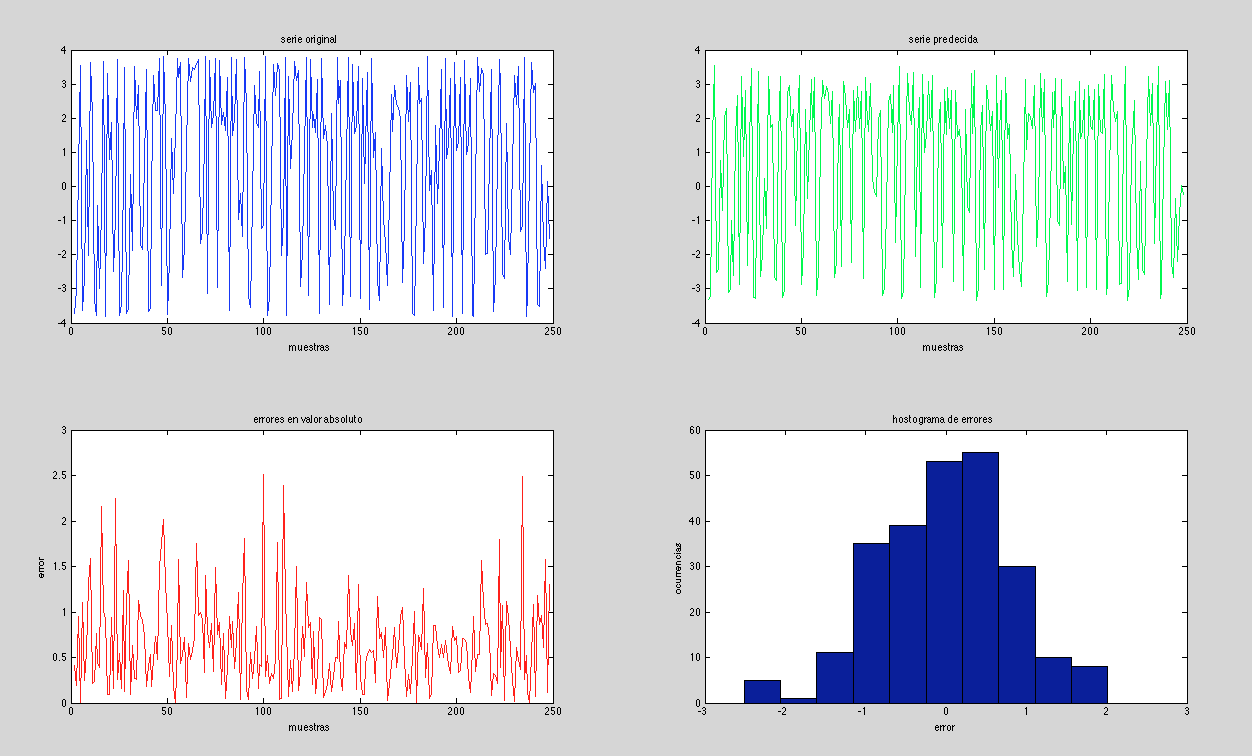


Figura 2 : Fase de prueba con configuración 3,5,1 y función tangente hiperbólica

