Sistemas de Inteligencia Artificial

Redes Neuronales

Trabajo Práctico Especial 2

Civile, Juan Pablo 50453
Ordano, Esteban 50753
Crespo, Alvaro 50758

24 de abril de 2013

1. Definición del problema

El problema consiste en la implementación de una red neuronal multicapa con aprendizaje supervisado que realice la predicción de series temporales. Para ello se supone que $x(t)$ (el valor de la serie en el paso temporal t) es alguna función desconocida de los valores de la serie en pasos anteriores, es decir, $x(t) = f(x(t-1), x(t-2), x(t-3), \dots)$. La red neuronal deberá poder aproximar a la función f desconocida. No se conoce la cantidad de valores anteriores de la serie respecto a los cuales f es función.

2. Implementación

Se implementó una red multicapa de tipo `Feed Forward`, usando el algoritmo `Back Propagation` para el entrenamiento de la misma. La distribución inicial de pesos se generó de manera aleatoria con valores acotados a un intervalo dependiente de la función de activación en uso. Como medida del error se usó la función de error cuadrático medio.

2.1. Funciones de activación

Se utilizaron las siguientes funciones de activación y sus respectivas derivadas:

- Función Sigmoidea

$$g(x) = \frac{1}{1 + exp^{-x}}$$

$$g'(x) = \frac{1}{(-2 * cosh(x) - 2)}$$

- Tangente hiperbólica

$$g(x) = tanh(x)$$

$$g'(x) = sech(x)^2$$

2.2. Normalización de la entrada y la salida

Dado que las entradas y salidades de la red pertenecen al intervalo $(-4, 4)$, y las entradas y las salidas de las funciones de activación utilizadas pertenecen al intervalo $(0, 1)$ para el caso de la sigmoidea, y al $(-1, 1)$ en el caso de la tangente hiperbólica, se debe realizar una normalización tanto en la entrada como en la salida de la red. Estas normalizaciones son las siguentes:

- Sigmoidea

  Para la entrada:

$$\xi_i = \frac{x_i + 4}{8}$$

  Para la salida:

$$o_i = x_i * 4$$

- Tangente hiperbólica

  Para la entrada:

  $$\xi_i = x/4$$

  Para la salida:

  $$o_i = (x - 0{,}5) * 8$$

2.3. Mejoras

2.3.1. Momentum

Momentum considera los valores de Δw_{ij} de pasos anteriores a la hora de actualizar los pesos del paso actual. En particular, se agrega al cambio dado por `Back Propagation`, el cambio de la iteración anterior pesado por un factor α.

2.3.2. η adaptativo

Esta mejora busca alterar el valor de η de acuerdo al progreso del entrenamiento. Para esto se considera que si se encuentra una secuencia de k pasos que disminuyen el error, entonces se tiene un buen camino de entrenamiento y por lo tanto se aumenta el valor en una constante a para continuar avanzando en este sentido. Y si por el contrario se encuentra en un mal camino, se disminuye el valor en un porcentaje b. Cuando se disminuye el valor, se desactiva temporalmente la mejora `Momentum` y se descarta el paso realizado. Una vez que se encuentra un paso que disminuye el error se reactiva la estrategia de `Momentum`.

2.3.3. Ruido

Se intentó introducir ruido a los pesos de la red entre pasos para evitar caer en mínimos locales. Se agregó a cada peso un valor aleatorio entre 0 y 1, adjustado por la cantidad de *rollbacks* (entrenamientos que se descartaron por no disminuir el error), y una constante. Esta optimización fue descartada porque en manera consistente no mejoraba los resultados obtenidos.

2.4. Conjunto de entrenamiento y prueba

Se tiene un conjunto de 1000 puntos de la serie a ser generalizada. Para el entrenamiento de la red se utilizaron los primeros 800 puntos, y los restantes 200 puntos fueron utilizados como conjunto de prueba. Siempre que se habla del error medio de una red, se refiere al error cuadrático medio del conjunto de prueba.

2.5. Disminución de la cantidad de entrenamientos descartados

Se consideró incorporar otros criterios para deshacer entrenamientos utilizando la estrategia de entrenamiento adaptativo. Además de considerar el error cuadrático medio, se consideró como un valor relevante el de un percentil alto (se seleccionó el percentil 90) de los errores del conjunto de entrenamiento. El objetivo es no descartar entrenamientos que son mejores, aunque el error medio no disminuya. Dos nuevos criterios utilizando este nuevo parámetro son no realizar *rollback* si:

- Disminuye el error cuadrático medio **y** disminuye el percentil 90.

- Disminuye el error cuadrático medio **o** disminuye el percentil 90.

2.6. Selección de los parámetros de configuración

El entrenamiento de la red neuronal con las estrategias implementadas tiene una extensa variedad de parámetros de configuración. Además de η, el coeficiente de aprendizaje, el entrenamiento con la estrategia *Momentum* requiere seleccionar un valor para α, el término que pesa el descenso promedio. La estrategia de entrenamiento adaptativo requiere la selección de más parámetros: los factores a, b, y la cantidad de pasos k. Se determinaron valores iniciales que producían resultados suficientemente satisfactorios en 1500 iteraciones. Se consideró cada parámetro independientemente de los demás y se iteró por valores relevantes para cada variable.

3. Resultados y Conclusiones

3.1. Comparación de Funciones de Activación

Para realizar una comparación entre las funciones de activación utilizadas, se definió un set de arquitecturas de prueba de manera arbitraria. Para esta comparación, se mantuvieron fijos los parámetros de entrada de la red y se limitó la cantidad de iteraciones a 1500. Según la función de activación usada en cada caso, se acotaron los valores aleatorios de los pesos utilizados como entrada al algoritmo de aprendizaje supervisado. Como era de esperarse, para la función sigmoidea, hubo que generar números muy pequeños en módulo (cercanos a 0), para que no se saturen las conexiones, y se impida el aprendizaje. Para el caso de la tangente hiperbólica, en cambio, esos pesos iniciales pequeños no retornaron resultados positivos, como si lo hicieron las pruebas realizadas con valores aleatorios pero con cotas mayores.

Arquitectura \Función de activación	Tangente hiperbólica	Sigmoidea
[2 5 4 1]	0,023053	0,31491
[3 10 6 1]	0,053163	0,30446
[2 10 6 1]	0,029276	0,31163
[2 4 2 1]	0,36896	0,31504
[3 14 8 1]	0,028585	0,30841

Cuadro 1: Comparación del error cuadrático medio entre las funciones de activación *Tangente hiperbólica* y *Sigmoidea*.

Se observa en 1 que las redes obtenidas con la función $tanh$ son ampliamente superiores a sus contrapartes con la función *Sigmoidea*. A partir de este resultado, se prosiguió el análisis en redes construidas con la función *tangente hiperbólica*.

3.2. Optimización de parámetros

Se muestran los resultados obtenidos de la selección descripta en la sección 2.6 para cada variable en la tabla 2. Al respecto, se hacen las siguientes observaciones:

- El mejor valor para el coeficiente $alpha$ es de $0{,}3$, mucho menor que el sugerido $0{,}9$.

- El mejor valor del parámetro a es de $0{,}0001$. Este valor es menor de lo esperado, ya que en pruebas preliminares un valor de $0{,}01$ no arrojaba peores resultados que correr el algoritmo sin la estrategia de entrenamiento adaptativo.
- El mejor valor del parámetro b es de $0{,}01$. Esto es mayor de lo esperado, el valor que se utilizaba inicialmente era de 0.001.

Parámetro	Mínimo valor	Incremento	Máximo valor	Mejor valor
$alpha$	0,1	$+0{,}1$	0,9	0,3
a	0,00001	10	0,01	0,0001
b	0,00001	10	0,01	0,01
k	1	$+1$	5	4

Cuadro 2: Parámetros de configuración de la red neuronal

Con este análisis se encontró que los valores utilizados como parámetros en la sección 3.1 resultaron muy cercanos a los óptimos. El único parámetro con valor distinto es a con un valor de $0{,}001$.

3.3. Estrategia de Rollback

Respecto a la sección 2.5, la disminución de la cantidad de entrenamientos descartados, se puede ver que la estrategia de hacer rollback si el percentil 90 no disminuye consigue mejores resultados que la estrategia convencional y la estrategia de hacer rollback sólo si no disminuye la media y el percentil. Esto se puede ver en el cuadro 3, donde este criterio mejora en un $25\,\%$.

Criterio	Rollbacks	$\overline{e}$	$\overline{\sigma^2}$	e_{max}
Solo la media	507,5	0,08498	0,14161	0,5671
And	476,6	0,06458	0,11795	0,55802
Or	495,4	0,09988	0,17392	0,76038

Cuadro 3: Criterio para descartar entrenamientos. Promedios obtenidos luego de 800 entrenamientos, sobre un conjunto de diez ejecuciones.

3.4. Optimización de arquitectura

Una vez conocidos los parámetros óptimos, se buscó optimizar la arquitectura utilizando estos valores. Para esto se corrieron repetidas ejecuciones de un abanico de arquitecturas de 2 entradas. No se utilizaron arquitecturas de 3 entradas ya que en la tabla 1 se observó que las redes de 3 entradas no presentan mejoras sustanciales sobre las de 2.

Las arquitecturas que se consideraron tienen 2 capas ocultas. La siguiente figura muestra los menores errores obtenidos en estas ejecuciones para cada arquitectura. Se describe una arquitectura de N neuronas en la primer capa oculta y M neuronas en la segunda capa oculta como una arquitectura de NxM.

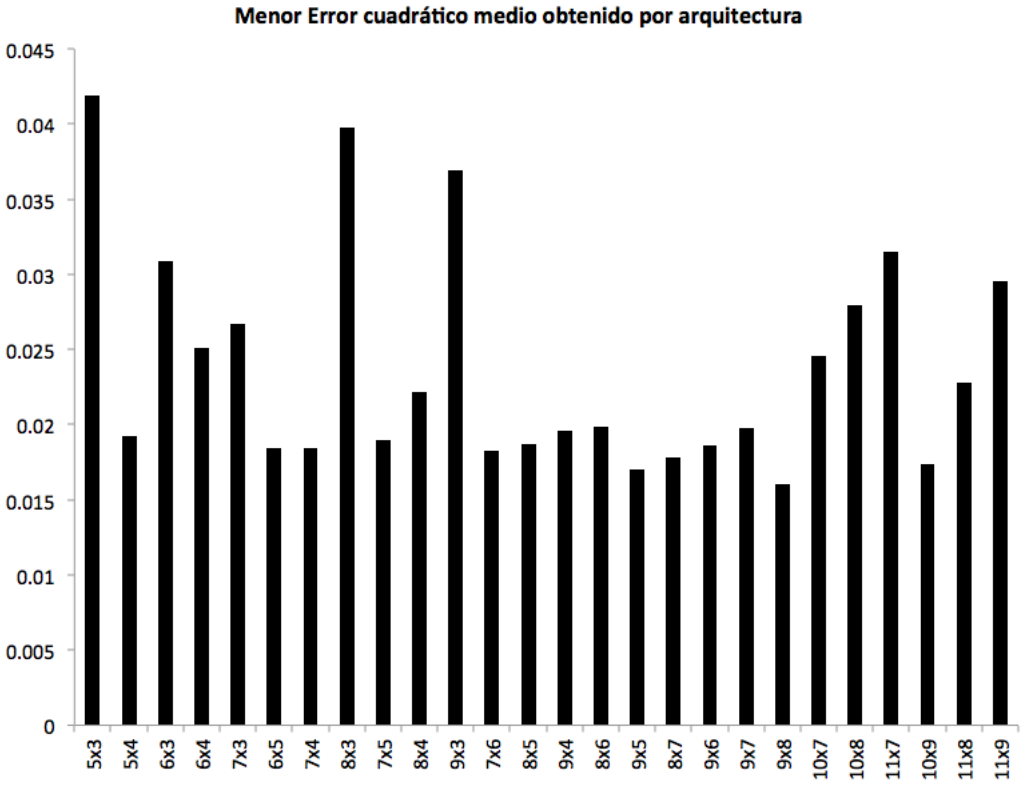

Figura 1: Error cuadrático medio obtenido para arquitecturas con 2 entradas variando la cantidad de neuronas en la primera y segunda capa oculta

De esta ejecución se pudo obtener nuestra mejor red. Es la de arquitectura de 2 entradas, 9 neuronas en la primer capa oculta y 8 en la segunda capa oculta.

Para esta red se obtuvo como error medio $0{,}01605$, una desviación standard del error de $0{,}025738$ y un error máximo de $0{,}11786$. Se puede comparar este resultado con el resto de las arquitecturas en la tabla 4 en el anexo.

Anexo: Resultados

Arquitectura	Error	Dev. Std.	Error Máx.
5x3	0.041884	0.067486	0.25286
5x4	0.019276	0.029188	0.10784
6x3	0.030846	0.047927	0.15913
6x4	0.025126	0.033176	0.13532
7x3	0.026747	0.033905	0.13142
6x5	0.018454	0.027995	0.10857
7x4	0.018442	0.027286	0.098194
8x3	0.039789	0.050507	0.1554
7x5	0.018988	0.028982	0.1068
8x4	0.022146	0.030715	0.10197
9x3	0.036945	0.047534	0.14866
7x6	0.018294	0.026754	0.11407
8x5	0.018705	0.029107	0.099821
9x4	0.019588	0.028886	0.10269
8x6	0.019858	0.028843	0.10546
9x5	0.017007	0.026745	0.10267
8x7	0.017853	0.026049	0.10511
9x6	0.018605	0.025973	0.093161
9x7	0.019796	0.027959	0.092556
9x8	0.01605	0.025738	0.11786
10x7	0.024545	0.034505	0.10629
10x8	0.02795	0.034333	0.10529
11x7	0.031485	0.037798	0.095195
10x9	0.017352	0.026223	0.12413
11x8	0.022803	0.030054	0.092118
11x9	0.029583	0.039647	0.11379

Cuadro 4: Detalle de los valores obtenidos para distintas arquitecturas utilizando la función $\tanh$ y usando los parámetros óptimos