Computación Neuronal y Evolutiva

(Práctica 2)

UNIVERSIDAD DE BURGOS

Autores:

Mario Ubierna San Mamés

Jorge Navarro González

Índice de Contenido

Aspectos relevantes de la generación de datos y programación del script

Como vamos a poder observar en posteriores capturas, hemos organizado el programa de tal forma que vamos a diferenciar cuatro scripts distintos, en los cuales uno llama a los demás para completar con todo lo necesario para hacer el programa completo.

Este último será el que llamará al MLP y al RBF con los cuales vamos a obtener los valores de tiempo, error y las gráficas correspondientes para poder estudiar la aproximación como haremos posteriormente.

```
function [ valor ] = funcion(x)
%Función que nos devuelve la y de la siguiente función:
%((-x).*(x+3).*(x-2))+(15*cos(5*x))

    valor = ((-x).*(x+3).*(x-2))+(15*cos(5*x));
end
```

Ilustración 1. Función que devuelve el valor de la función.

Esta como vamos a ver es el script más sencillo que vamos a tener en nuestro programa. Como se puede observar, lo único que va a hacer es, dado un parámetro pasado por cabecera “x”, va a devolver ese valor que tendría nuestra función en ese punto “x” en la variable “valor”.

```
function [ error, tiempo ] = MLP(neuronas)
    tiempo = cputime;
    x = -6:.01:6;
    y = funcion(x);
    plot(x,y,'k+'); grid;
    xlabel('tiempo (s)');
    ylabel('salida');
    title('((-x).*(x+3).*(x-2))+(15*cos(5*x))');
    net = feedforwardnet(neuronas,'trainlm');
    net = configure(net, [-6 6], [147,-214]);
    net.trainParam.lr = 0.005;
    net.trainParam.epochs = 100;
    net.trainPAram.max_fail = 45;
    net.trainParam.max_fail = 45;
    net.trainParam.goal = 1e-15;
    P=x; T=y; [net,TR] = train(net,P,T);
    a = -5.995:.01:6;
    b = funcion(a);
    c = sim(net,a);
    plot(a,b,'k+');
    hold on;
    plot(a,c,'bs');
    plot(a,b-c,'r');
    grid;
    error = mse(net,b,c);
    tiempo = cputime - tiempo;
end
```

Ilustración 2. Función MLP

Esta segunda imagen se va a corresponder con el script del MLP, como podemos observar lo que le pasamos por cabecera va a ser el número de neuronas, y lo que nos va a devolver el script va a ser dos variables, el error que comete y el tiempo que tarda.

En el script se puede ver como cogemos primero el intervalo, evaluando esa función en dicho intervalo y a continuación vamos a hacer la aproximación que hará nuestra red neuronal. Para ello vamos a definir otro intervalo y procederemos al entrenamiento de nuestra red como está en los apuntes del tema.

Por último, vamos a devolver en esta función el error cometido y el tiempo, de tal manera que podamos saber exactamente qué valor es el de dichas variables, y como podemos ver en el código también vamos a dibujar la gráfica, de tal forma que dibuje la gráfica real y la aproximación que hace nuestra red neuronal.

```
function [ error, tiempo ] = RBF(neuronas)
    tiempo = cputime;
    x = -6:.01:6;
    y = funcion(x);
    figure;
    plot(x,y,'k+'); grid;
    xlabel('tiempo (s)');
    ylabel('salida');
    title('((-x).*(x+3).*(x-2))+(15*cos(5*x))');

    net = newrb(x,y,1e-15,1,neuronas);

    a = -5.995:.01:6;
    b = funcion(a);
    c = sim(net,a);
    plot(a,b,'k+');
    hold on;
    plot(a,c,'bs');
    plot(a,b-c,'r');
    grid;

    error = mse(net,b,c);
    tiempo = cputime - tiempo;
end
```

Ilustración 3. Función RBF.

En este siguiente script, vamos a ver lo mismo que en el anterior pero esta vez con la función de RBF. A este script también le vamos a pasar por cabecera el número de neuronas y al igual que el anterior nos va a devolver los valores del error y el tiempo que se tarda en ejecutar.

De la misma forma que antes, también nos va a dibujar lo que es la gráfica de cómo la función es realmente y cómo es la aproximación que hace la red neuronal, con la finalidad de que podamos compararlas de una forma muy sencilla.

```
function [ errorMLP, tiempoMLP, errorRBF, tiempoRBF ] = main(neuronas)
% Programa principal

    [ errorMLP, tiempoMLP ] = MLP(neuronas);
    [ errorRBF, tiempoRBF ] = RBF(neuronas);
end
```

Ilustración 4. Función main.

Este sería el último script, el cual realmente lo único que hace es llamar a los dos anteriores de tal forma que tengamos los errores y los tiempos de cada una de las redes y además las gráficas de las dos redes al mismo tiempo.

El script lo que nos va a devolver va a ser los errores y tiempos de cada una de las redes y las gráficas que ha dibujado de forma que sea sencillo su estudio.

Resultados obtenidos

A continuación, vamos a ver que hemos hecho cuatro casos con distintos números de neuronas para ver las distintas aproximaciones que va a sacar nuestras redes neuronales, comparando una con otra y viendo cuál aproxima mejor en el caso de que sólo tuviéramos en cuenta el número de neuronas.

Este trabajo consta de 4 casos, el desarrollo va a seguir el siguiente orden:

- Caso 1 (1 neuronas).
- Caso 2 (5 neuronas).
- Caso 3 (10 neuronas).
- Caso 4 (15 neuronas).

Como podemos ver el número de neuronas es muy bajo, esto lo hemos pensado debido a que la función que se desea estudiar es muy simple, por tanto, a partir de quince neuronas ya nos daba un resultado bastante cercano a un error nulo, por tanto, hemos elegido los números que consideramos que más se adecúan y posibilitan la visualización clara de la evolución.

Caso 1

Para el primer caso como hemos visto anteriormente va a ser en el que estudiemos la evolución con una sola neurona en nuestra red, para ello vamos a ejecutar el script main y veremos lo que nos devuelve como resultado.

En primer lugar, vamos a tener los errores y sus tiempos:

```
>> [ errorMLP, tiempoMLP, errorRBF, tiempoRBF ] = main(1)
NEWRB, neurons = 0, MSE = 4260.76

errorMLP =

  863.1874


tiempoMLP =

    0.9063


errorRBF =

   1.2383e+03


tiempoRBF =

    0.3281
```

Ilustración 5. Errores y tiempos en el Caso 1.

Como se puede observar el error que comete el MLP en este caso va a ser menor que en el caso del RBF, sin embargo, vamos a observar también cómo el RBF va a tardar mucho menos en aproximar la función de lo que tarda el otro.

Esto que nos ocurre ahora va a ser la tónica general que va a seguir toda la comparativa con todos los casos que vamos a seguir.

Otra de las cosas que nos saca por pantalla van a ser las dos gráficas en las que veremos la aproximación:

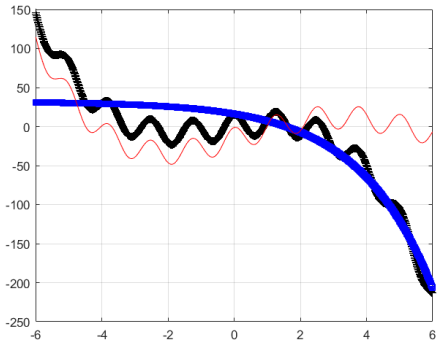

Ilustración 6. MLP con una neurona.

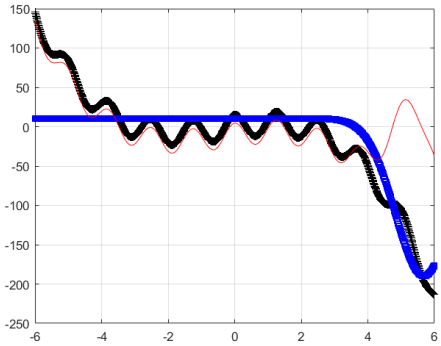

Ilustración 7. RBF con una neurona.

Como hemos dicho anteriormente con los valores numéricos, se puede ver perfectamente como la aproximación ejecutada por el MLP, en este caso arriba, es mucho mejor que la del RBF, que es bastante peor.

Caso 2

Para el segundo caso vamos a seleccionar las dos redes con cinco neuronas, de tal manera que podemos pensar que los resultados van a ser más ajustados y los errores por tanto deberían ser menores que en el caso anterior.

Valores de tiempo y error medidos:

```
>> [ errorMLP, tiempoMLP, errorRBF, tiempoRBF ] = main(5)
NEWRB, neurons = 0, MSE = 4260.76

errorMLP =

   85.1823


tiempoMLP =

    0.5781


errorRBF =

  126.8572


tiempoRBF =

    0.6094
```

Ilustración 8. Errores y tiempos en el Caso 2.

Como podemos ver, el error es bastante menor que en el caso anterior, aunque sigue la misma tónica y el error del MLP es mucho menor que el error cometido por el RBF, podemos ver que el tiempo en este caso del MLP ha sido un poco más bajo que en el caso del MLP.

Gráficas de aproximación:

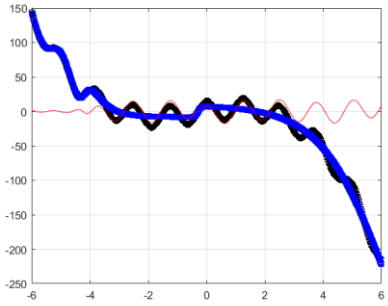

Ilustración 9. MLP con cinco neuronas.

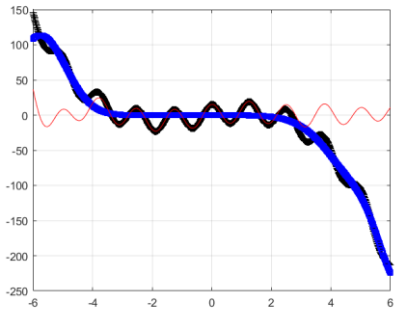

Ilustración 10. RBF con cinco neuronas.

Se puede observar que la aproximación del MLP sigue siendo mejor que la del RBF aunque se puede apreciar que ambas dos, tal y como indican sus resultados numéricos, cada vez tienen menor error y, por tanto, aproximan mucho mejor la función cuantas más neuronas pongamos.

Caso 3

En este tercer caso vamos a cambiar el número de neuronas a diez y observaremos cómo se va aproximando aún más la gráfica y cómo el error va a tender a cero cada vez más.

Valores de tiempo y error medidos:

```
>> [ errorMLP, tiempoMLP, errorRBF, tiempoRBF ] = main(10)
NEWRB, neurons = 0, MSE = 4260.76

errorMLP =

   40.5333


tiempoMLP =

    0.6719


errorRBF =

  102.7577


tiempoRBF =

    0.9531
```

Ilustración 11. Errores y tiempos medidos en el Caso 3.

El error en este caso va a seguir yendo a menos cada vez más y tendiendo a cero, como podemos ver en el caso del MLP se ha reducido ya casi a la mitad respecto del anterior caso. El RBF sigue dando un error bastante por encima de lo que debería ocurrir y el tiempo respecto al MLP es superior, aunque se puede observar también que el tiempo no aumenta tan proporcionalmente como lo hace la disminución del error. Por lo que podríamos decir que en general el RBF maneja mejor los tiempos que tarda en dar con la aproximación.

Gráficas de aproximación:

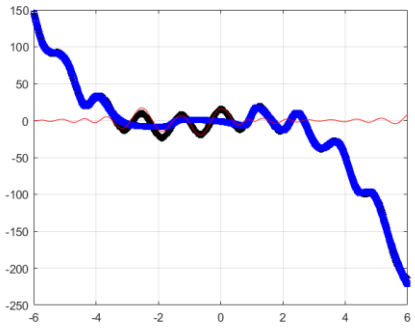

Ilustración 12. MLP con diez neuronas.

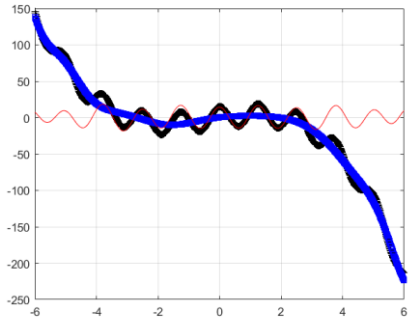

Ilustración 13. RBF con diez neuronas.

Como vemos, como en todos los casos anteriores, se corresponde muy bien lo que apreciamos a nivel numérico con lo que vemos a nivel gráfico, se puede ver como ya la red neuronal del MLP tiene casi aproximada la gráfica de forma muy exacta, no como el RBF que vemos que le cuesta bastante más encontrarlo.

Caso 4

Vamos a ir con el último de los casos que hemos ejecutado, en este caso va a ser el de quince neuronas donde vamos a obtener los siguientes resultados numéricos.

Valores de tiempo y error medidos:

```
>> [ errorMLP, tiempoMLP, errorRBF, tiempoRBF ] = main(15)
NEWRB, neurons = 0, MSE = 4260.76

errorMLP =

    1.8083


tiempoMLP =

    1.4063


errorRBF =

   78.0370


tiempoRBF =

    1.3125
```

Ilustración 14. Errores y tiempos medidos en el Caso 4.

Vemos que en el error del MLP ya tiende casi a cero, dejándolo en uno y pico. En el caso del RBF aún tiene bastante más error que el MLP en el caso anterior. Vemos que en cuanto al tiempo como dijimos en el anterior caso, el del RBF es menor debido a que gestiona mucho mejor el tiempo y a que encuentra una solución en un tiempo menor que el MLP generalmente.

Gráficas de aproximación:

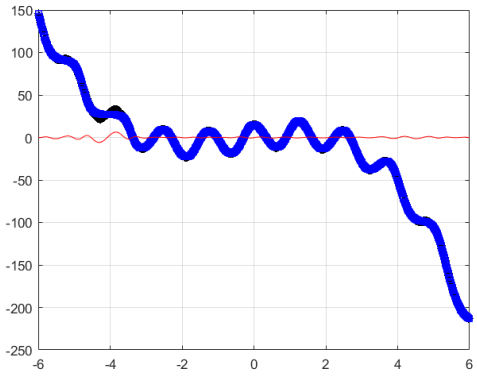

Ilustración 15. MLP con quince neuronas.

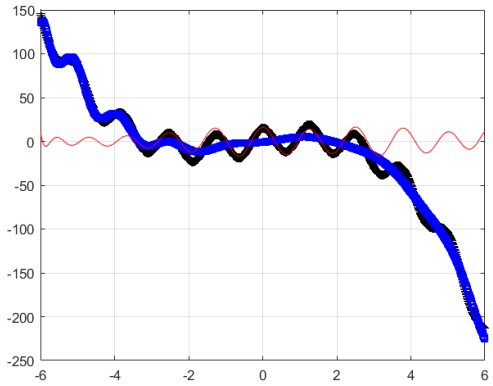

Ilustración 16. RBF con quince neuronas.

Podemos ver un ligero error de aproximación en torno al -4 en el eje de las X del MLP pero por lo general podemos ver que la función la ha aproximado casi a la perfección, como reflejaban los datos numéricos que teníamos anteriormente.

En el caso del RBF todavía se puede ver como el error aún sigue siendo demasiado grande como para poder decir que ha aproximado con exactitud.

Conclusiones Obtenidas

Podemos sacar unas conclusiones claras de este trabajo realizado. En primer lugar, cabe mencionar que la función es sencilla y por tanto no supone un gran problema para la red de aproximar sin necesidad de usar un número muy elevado de neuronas.

Podemos ver que cuando damos un mayor número de neuronas nuestra red aproxima de una forma mucho más exacta y así sucesivamente mientras aumenta el número de neuronas por tanto podemos ver que a mayor número de neuronas mejor va a aproximar la red que tenemos.

Estas conclusiones que podemos sacar son comunes a ambas redes, ya que, en ambas dos, aunque no de la misma forma, mientras se iba subiendo el número de neuronas de la red, mejor aproximación teníamos en dicha red.

Cabe destacar de la comparativa de las dos redes que en cuanto a la red MLP, si sólo comparamos con el número de neuronas, es una red mucho mejor para esta aproximación que la red RBF, ya que consigue reducir su error considerablemente aumentando muy poco el número de neuronas y además tiene un error mucho menos que la red RBF teniendo el mismo número de neuronas.

También hay que decir que la red RBF tiene mejores tiempos que la red MLP, es decir, generalmente, tarda menos en encontrar la aproximación que la red MLP por tanto podríamos decir que en este aspecto es mejor.

Probablemente vamos a concluir que si comparamos solo el número de neuronas va a ganar la red MLP pero que probablemente si aumentamos el número de neuronas de la red RBF podríamos obtener un resultado bastante acertado en un tiempo menor que la red MLP. Es decir, respecto al número de neuronas es mejor la red MLP, pero respecto al tiempo que tarda cada red en encontrar soluciones va a ser mejor la red RBF, con pocas neuronas como en nuestro ejemplo vamos a ver que la diferencia tampoco es tan grande, pero si aumentamos mucho el número de neuronas podremos ver que tarda menos de forma considerable.