

Explicación del uso de la red neuronal:

-En la línea 57 podemos darle como parámetro el número de capas con la que trabajara la red neuronal.

-En la línea 58 al array "lSz" le damos como valores el número de neuronas por capa en ese orden, teniendo en cuenta que la primer posición es la capa de entrada y la ultima la capa de salida.

-En la línea 60 y 61 damos los parámetros para el entrenamiento como son por ejemplo: el "thresh" que sirve como umbral de error, al valor de alpha, número de iteraciones.

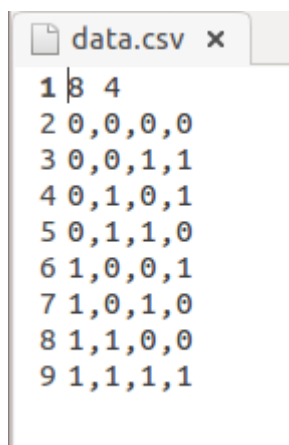
```
57 int numLayers =4 ;  
58 int *lSz=new int[numLayers]{columnas-1,3,3,1};  
59  
60 double beta = 0.2, alpha = 0.01, thresh = 0.00001;  
61 long num_iter = 1000000;  
62
```

-En la línea 65 declaramos las funciones como punteros a función que utilizaremos en cada capa en el mismo orden que se crearon las capas (línea 58). Tener en cuenta que la posición 0 de ese vector se ignorara puesto que es la capa de entrada y no se le aplica una función de activación.

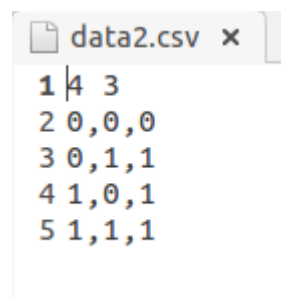
```
65 double (*funciones[])(double)={sigmoid,sigmoid,sigmoid,sigmoid};
```

Data de prueba:

leemos de un archivo llamado "data.csv" o "data2.csv" donde los dos primeros parámetros de la primera línea indican el número de filas y columnas respectivamente, siendo la última columna el valor de salida de cada predicción.



```
data.csv x  
1 8 4  
2 0,0,0,0  
3 0,0,1,1  
4 0,1,0,1  
5 0,1,1,0  
6 1,0,0,1  
7 1,0,1,0  
8 1,1,0,0  
9 1,1,1,1
```



```
data2.csv x  
1 4 3  
2 0,0,0  
3 0,1,1  
4 1,0,1  
5 1,1,1
```

Resultados:

-Al ejecutar el algoritmo de back propagation podemos notar que los resultados salen muy cercanos a los originales manteniendo un error muy pequeño.

Correcto

```
0 0 0 0
0 0 1 1
0 1 0 1
0 1 1 0
1 0 0 1
1 0 1 0
1 1 0 0
1 1 1 1
```

Salida

```
0 0 0 0.0056519
0 0 1 0.995637
0 1 0 0.994667
0 1 1 0.00477568
1 0 0 0.995637
1 0 1 0.00634299
1 1 0 0.00477652
1 1 1 0.996838
Press <RETURN> to close this window...
```