Neurocomputación

Práctica 1

Roberto García Teodoro
Mario Valdemaro García Roque

1.Neuronas de McCulloch-Pitts:

Explicación sobre la red neuronal:

Supusimos que la red debía tener dos capas ocultas, y que por lo tanto tendríamos dos ciclos de delay.

Las neuronas marcadas con X representan las neuronas de entrada, las neuronas marcadas con una Z son las entradas del ciclo anterior (ya que se activan al activarse sus antecesoras X).

De forma que las neuronas J son las que guardan si ha habido bajada o subida, por ejemplo, en el sensor de bajada, que se encuentra dibujado a continuación, supongamos que en primer lugar nos llega una entrada 100, en el primer ciclo las X serían 1 0 0 respectivamente, mientras el resto de neuronas se encontraría a 0.

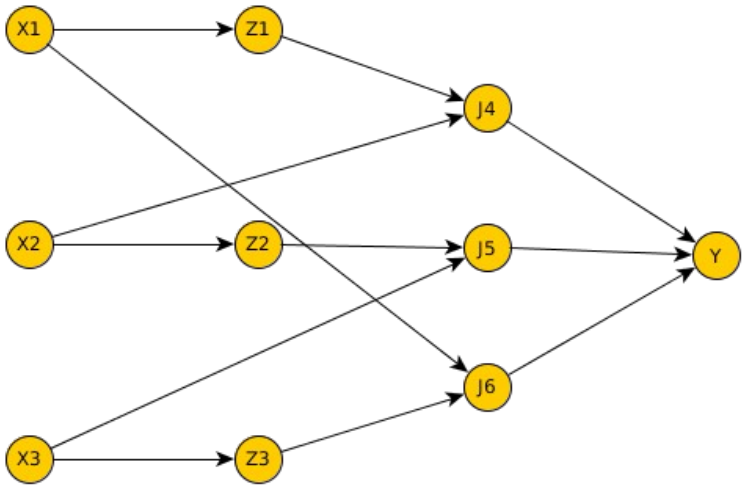
En el segundo ciclo (recibimos, por ejemplo, 010), los valores 1 0 0 han pasado a las neuronas Z, que ahora se encuentran a 1 0 0 respectivamente, mientras que las X recogen los nuevos valores 0 1 0.

Por tanto, vemos que ha bajado, así que las neuronas J4, J5 y J6 se pondrán a 1, ya que hacen de AND activándose si dados dos estados, en el segundo de ellos las entradas han bajado.

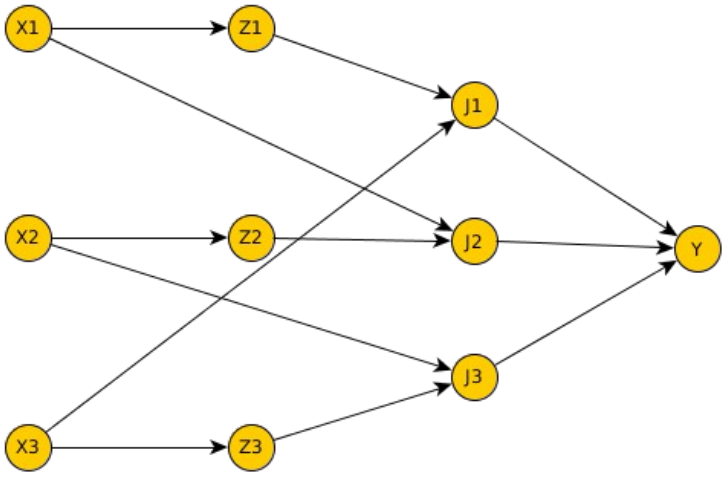
En el siguiente ciclo, la neurona de salida Y se pondrá a 1 si una de las neuronas J4, J5 o J6 se ha activado en el paso previo. Es decir, hace de OR.

El razonamiento es equivalente en la zona de arriba, solo que las líneas que unen las entradas de estado anterior y entrada con las neuronas J1, J2 y J3 están cambiadas para representar la subida.

Sensor abajo: el peso de las ramas es 1 para todas y la tasa de activación es un valor menor a 1 y mayor que 0 para todas neuronas salvo para las neuronas J que están entre 1 y 2.



Sensor arriba: el peso de las ramas es 1 para todas y la tasa de activación es un valor menor a 1 y mayor que 0 para todas neuronas salvo para las neuronas J que están entre 1 y 2.



Explicación sobre el programa:

En primer lugar procedemos a leer un fichero que contiene los estados de las neuronas de entrada en una secuencia, por ejemplo, 010100010010, donde cada 3 números corresponden a una entrada, en este caso se tratarían de 010, 100, 010 ,010
Tras esto inicializamos las neuronas con los pesos y los valores de activación.
Después actualizamos las neuronas de entrada con los valores de la entrada inicial y actualizamos las neuronas desde atrás hacia delante.
E imprimimos por la terminal los valores de las neuronas en cada iteración de la siguiente manera:

```
LECTURA DEL FICHERO
INICIALIZACION DE LA RED NEURONAL
ESTABLECIMIENTO DE LA RED NEURONAL
EVOLUCION DE LA NEURONA
 010 100 010 010 000 000 0
0       0       0
1       0       0
0       0       0       0:Arriba
                0
                0
                0       0:Abajo
entrada:100
valores de i:3 4 5
1       0       0
0       1       0
0       0       0       0:Arriba
                0
                0
                0       0:Abajo
entrada:010
valores de i:6 7 8
0       1       0
1       0       1
0       0       0       0:Arriba
                0
                0
                0       0:Abajo
```

```
entrada:010
valores de i:9 10 11
0       0       0
1       1       0
0       0       0       1:Arriba
                1
                0
                0       0:Abajo
entrada:000
valores de i:12 13 14
0       0       0
0       1       0
0       0       0       0:Arriba
                0
                0
                0       1:Abajo
entrada:000
valores de i:15 16 17
0       0       0
0       0       0
0       0       0       0:Arriba
                0
                0
                0       0:Abajo
entrada:000
valores de i:18 19 20
0       0       0
0       0       0
0       0       0       0:Arriba
                0
                0
                0       0:Abajo
```

De forma que podemos ver como para los valores 010 → 100 se activa la neurona de arriba (tras las dos iteraciones de delay), para los valores 100 → 010 se activa la neurona de abajo, etc.

2. Perceptrón y Adaline:

1.Perceptrón y adaline con problemas1, test=0,35 train=0,65

Ejecutamos el programa para los 2 tipos de neuronas y obtenemos las siguientes salida:

perceptrón:

tasa de fallo: 5.49 %
fallos:14 datos:255
pesos:-0.1400 0.2300 -0.4500 -0.1000 -0.0400 -0.2250 -0.3600 -0.3000 -0.4800 0.6000
pesos:0.1400 -0.2300 0.4500 0.1000 0.0400 0.2250 0.3600 0.3000 0.4800 -0.6000
tasa 0.1000 train 0.6500 test 0.3500 etapas 10000

adaline:

tasa de fallo: 3.94 %
fallos:10 datos:254
pesos:-0.7400 0.4400 -1.0000 -0.8400 0.2000 -0.7900 -1.2800 -0.1600 -1.0000 1.6000
pesos:1.1200 -0.3400 1.1600 0.6800 -0.4000 0.6000 1.0000 0.2200 1.0200 -1.6000
tasa 0.1000 train 0.6500 test 0.3500 tolerancia 0.0100000000000000 etapas 10000

no hemos incluido las predicciones, primero porque no entrarían en el documento y segundo porque incluso en el programa no son relevantes. Lo realmente importante es la tasa de fallos. Por lo que en lo sucesivo nos vamos a centrar en esto más que en lo que se predice.

Una cosa que podemos observar si realizamos varias mediciones de la tasa de fallo es que varía bastante, por tanto sería interesante realizar varias mediciones.

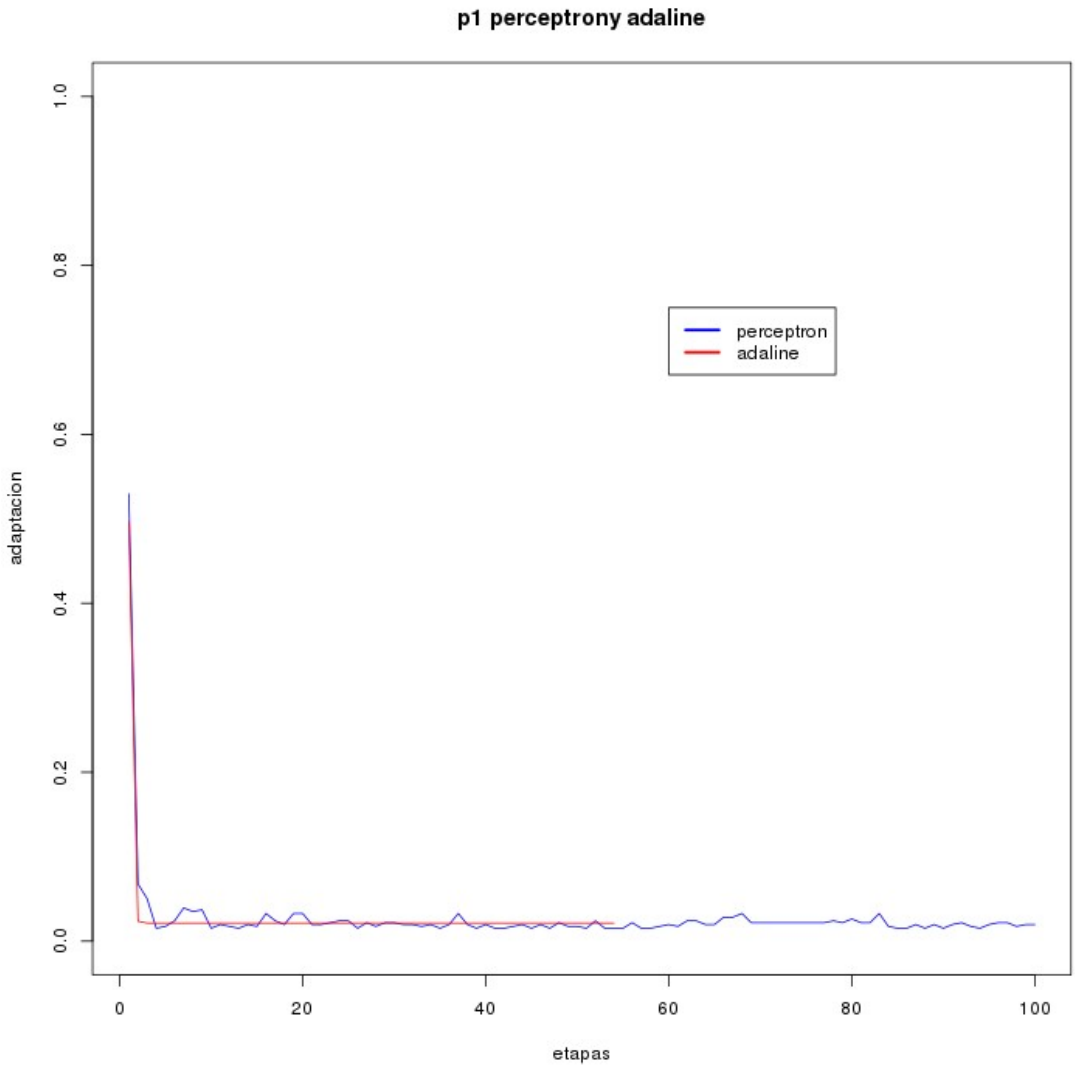
Si las realizamos veremos que estas se sitúan entre valores medios de 2%-3%.

2.Intentando ajustar el problema lo máximo posible.

Para este problema vamos a usar siempre un 65% de train y un 35% de test.

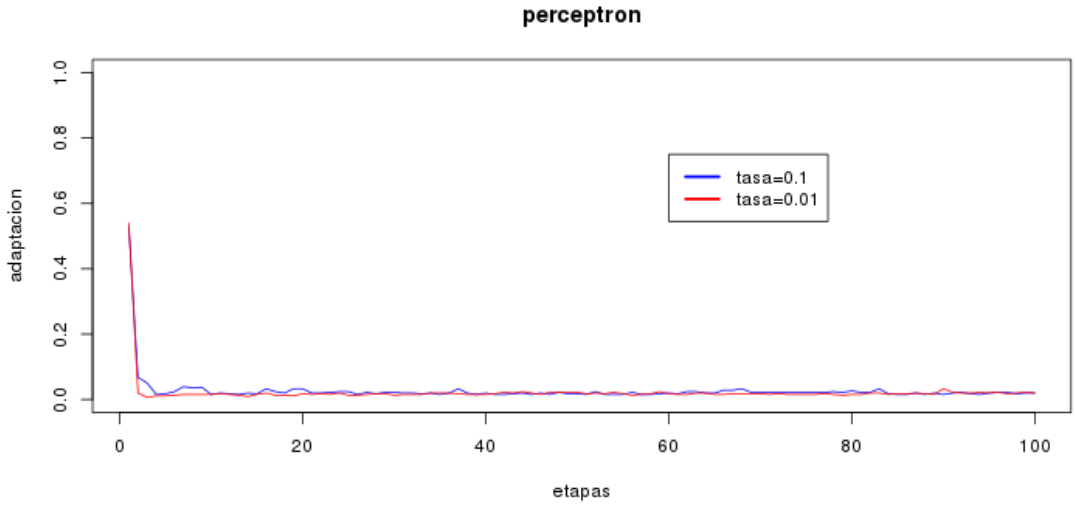
Lo primero que se nos ocurre es intentar ajustar lo máximo posible las etapas del problema. Por tanto vamos probando con distintas cantidades de etapas, pero sin embargo, obtenemos casi siempre los mismos resultados independientemente de las etapas que usemos, tanto para adaline como para el perceptrón. Por tanto investigamos este efecto.

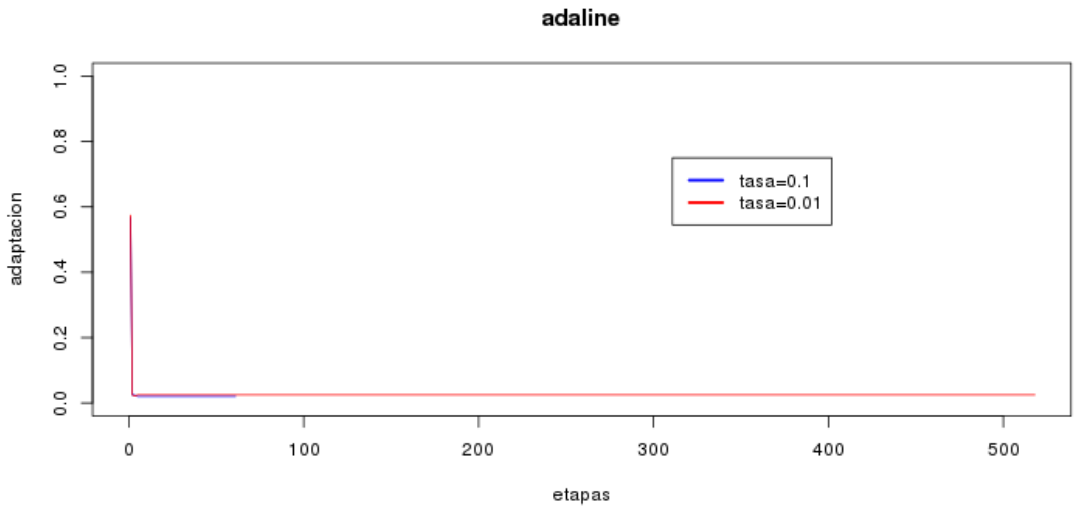
En la siguiente gráfica vemos como se va adaptando la red neuronal al problema que analizará:



Tanto para adaline como para el perceptrón observamos lo mismo, la adaptación al problema es muy rápida teniendo ya en las primeras 100 etapas casi la misma adaptación que en la ultima parte del problema.
Conclusión: el numero de etapas no es demasiado importante mientras esté en unos valores por encima de 1000, aproximadamente.

Lo siguiente que estudiaremos es la influencia de variar la tasa de aprendizaje en ambas redes sobre la adaptación al problema:





Como observamos, bajando la tasa de aprendizaje obtenemos unos resultados aparentemente bastante buenos, pero, ¿cómo de buenos son realmente estos datos?. Dado que, como hemos visto, los valores de los resultados varían mucho de una ejecución a otra, es difícil saber si el resultado que obtendremos sera el mejor o, por contra, un valor no demasiado bue-

no. Por tanto hemos decidido analizar la adaptación al problema a partir de 1000 etapas en el caso del perceptrón ya que parece mas estabilizado.

Una vez analizado obtenemos los siguientes datos estadísticos sobre la adaptación:

Perceptrón tasa de aprendizaje 0,1:

Error medio: 0,022

Perceptrón tasa de aprendizaje 0,01:

Error medio: 0,0224

Adaline tasa de aprendizaje 0,1:

Error medio: 0,043

Adaline tasa de aprendizaje 0,01:

Error medio: 0,036

Adaline tasa de aprendizaje 0,001:

Error medio: 0,0396

Observamos que bajar la tasa empeora los datos en el caso del perceptor y los mejora un poco en el caso de adaline, por tanto concluimos que usar una tasa de aprendizaje de 0,01 es adecuado para adaline y 0,1 para el perceptrón.

3.Clasificando problemas de puertas lógicas sin y con interpolación

Para ambas redes la tasa de aprendizaje es 0,1 y las etapas no son un factor demasiado relevante ya que la ejecución no dura mucho mas de 8 etapas, por tanto sera un valor mayor que esto.

Resultados:

Problema	Perceptrón	Adaline
NAND	Fallo: 0.00 %	Fallo: 0.00 %
NOR	Fallo: 0.00 %	Fallo: 0.00 %
XOR	Fallo: 100.00 %	Fallo: 50.00 %

Como vemos el problema de la XOR no se resuelve bien para ninguna de las 2 redes. Esto se debe a que los dos modelos hacen son divisores lineales del problema, es decir que dado un plano trazan una recta por donde separan el problema, y siendo el problema de la XOR no separable linealmente ambos modelos fallan al clasificar.

Una forma de resolver esto es “simular” que tuviésemos una capa oculta. La forma de hacer esto es con una operación matemática de varias columnas del problema, por ejemplo, el producto (que para este problema en concreto funciona). Por tanto para cada tupla hacemos el producto de las 2 columnas del problema y volvemos a entrenar obteniendo los siguientes resultados.

Problema	Perceptrón	Adaline
NAND	Fallo: 0.00 %	Fallo: 0.00 %
NOR	Fallo: 0.00 %	Fallo: 0.00 %
XOR	Fallo: 0.00 %	Fallo: 0.00 %

¿Como hemos conseguido resolver el problema de la puerta lógica XOR?
Si analizamos el problema de forma concreta vemos porqué:

Entrada	X1	X2	Interpolación	Sesgo
Pesos	-0.2000	-0.2000	0.5000	0.1000

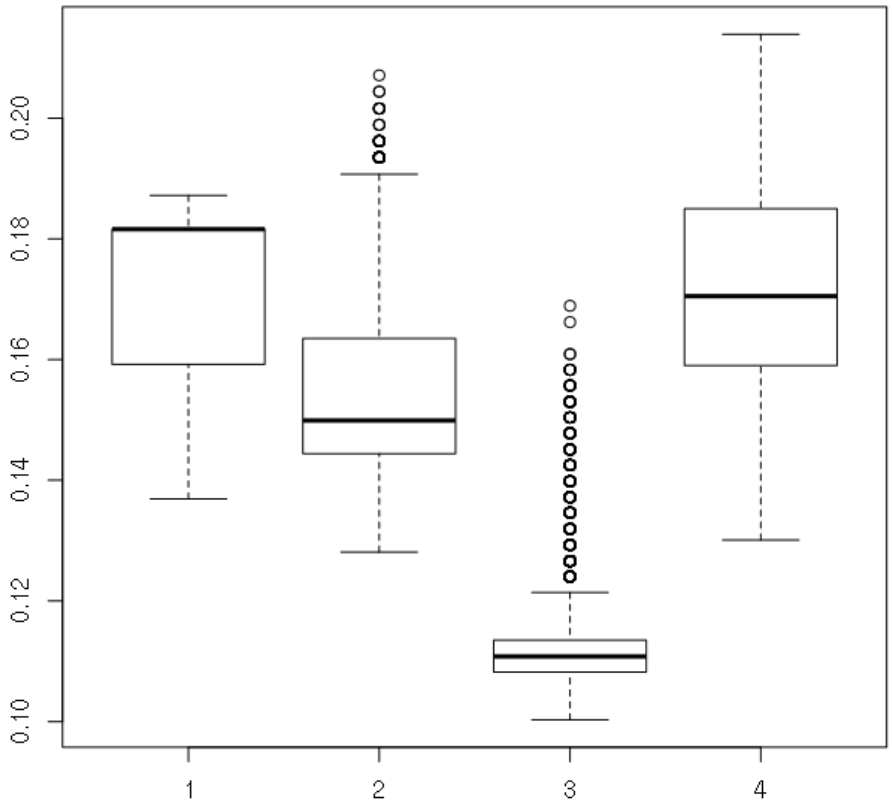
No hemos incluido la otra neurona porque es la opuesta a esta por tanto es solo cambiar el signo.

Como ya hemos explicado, la neurona de interpolación de este caso concreto es el producto de dos entradas, por tanto solo cuando X1 y X2 sean 1 su valor sera 1. Entonces podemos ver de forma concreta que clasifica bien ya que produciría estas salidas.

X1	X2	Entrada	Salida
0	0	0,1	1
0	1	-0,1	0
1	0	-0,1	0
1	1	0,2	1

4.Aplicar al problema real 2

Hemos visto que para el apartado anterior era útil usar interpolaciones, porque se adecuaba bien al problema, sin embargo parece que son algo muy concreto que sirve para un caso pero no para otros, por tanto hemos probado si obtenemos mejores datos con interpolaciones para el problema 2:



Esta gráfica corresponde a pruebas con el perceptrón, de izquierda a derecha se muestra: red sin interpolaciones, interpolación mediante el producto, interpolación mediante suma e interpolación mediante la media.

Método	Sin interpolar	InterPrd	InterSum	InterMed
Error	22,56	29,27	20,94	21,94

Conclusión, el mejor método es el de la suma.

Producto ya la hemos explicado, suma es la misma que el producto solo que con la operación suma, y media es hacer la media de todas las columnas.

Los datos de suma para perceptrón y de media para adaline parecen buenos para un train y test de 0,65 y 0,35. Sin embargo si entrenamos con todo la red se puede comprobar que predicen lo mismo que si no hiciésemos interpolaciones.

Todas las predicciones se encuentran en la carpeta predicciones.