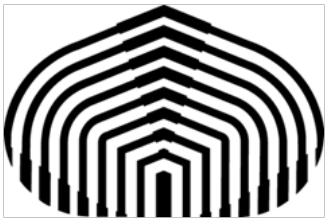


UNIVERSIDAD SIMÓN BOLÍVAR

DEPARTAMENTO DE COMPUTACIÓN Y T.I.

Proyecto 1_1

Autor:

Luis Miranda 10-10463

Jose Montenegro 10-10469

Sartenejas, Octubre 2014

Resumen

Las redes neurales (conocidas como RNA) son un paradigma de aprendizaje y procesamiento automático inspirado en el funcionamiento del sistema nervioso de los animales. Consiste en un sistema interconectado de neuronas que colaboran entre sí para producir un estímulo de salida.

La unidad básica de una red neural son las neuronas, cada una de estas recibe una serie de entradas y las traduce en una salida. Esta salida viene dada por tres funciones:

- Una funcion de propagación: se encarga de transformar las diferentes entradas que provienen de la sinapsis en el potencial de las neuronas.
- Una funcion de activación: combina el potencial postsináptico, que nos proporciona la función de propagación, con el estado actual para conseguir el estado futuro de activación de la neurona.
- Una funcion de transferencia: Esta función convierte el estado de la neurona en la salida hacia la siguiente neurona que se transmite por las sinapsis.

Un perceptron es el modelo matematico mas basico de una neurona. Este usa una matriz para representar las redes y es un discriminador terciario que traza su entrada un vector binario como entrada a un único valor de salida a traves de:

$$f(x) = \begin{cases} 1 & \text{si } w \cdot x - u > 0 \\ 0 & \text{en otro caso} \end{cases}$$

donde w es un vector de pesos reales y u es el umbral, el cual representa el grado de inhibición de la neurona y no depende de la entrada.

Un ADALINE, es un modelo muy similar al perceptron, excepto que su función de transferencia no es escalonada, si no lineal. El algoritmo que este usa, busca disminuir el error medio cuadrático, por consiguiente trata de mover la frontera de decisión tan lejos como sea posible del modelo de entrenamiento, cosa que el perceptron no hace, lo cual causa que el perceptron sea más sensible al ruido que hay en las muestras.

Tanto el ADALINE como el perceptron sufren de una misma limitación, ambos solo pueden resolver problemas linealmente separables.

Detalles de Implementación

Se implementaron 2 clases, Perceptron y Neurona cada con 2 metodos, resolver y aprender, aprender hace el proceso de aprendizaje y resolver aplica los pesos a los input y devuelve el resultado. En el caso de Perceptron, el resultado es -1 o 1 dependiendo de si la función lineal da mayor que 0. El aprendizaje es igual para ambas, aunque la condición de terminación en Perceptron es tener todos los ejemplos correctos, mientras que en Neurona solo se requiere llegar a un error mínimo

Para los casos de prueba se define un typedef llamado ejemplo que guarda la entrada y la salida de cada uno.

Resultados

Como se puede ver en las gráficas adjuntas, para el AND y el OR en el caso del perceptrn fueron necesarios 2 iteraciones para converger, contra las 11/12 que necesito el ADELINE. Para el caso del XOR aunque ninguno de los 2 logra llegar a un resultado aceptable (ya que XOR no es un problema linealmente separable) el ADELINE tarda entre 20 y 25 iteraciones en darse cuenta que el error no cambia y el perceptrón nunca llega a esta conclusión y por eso continua a mas 300 iteraciones

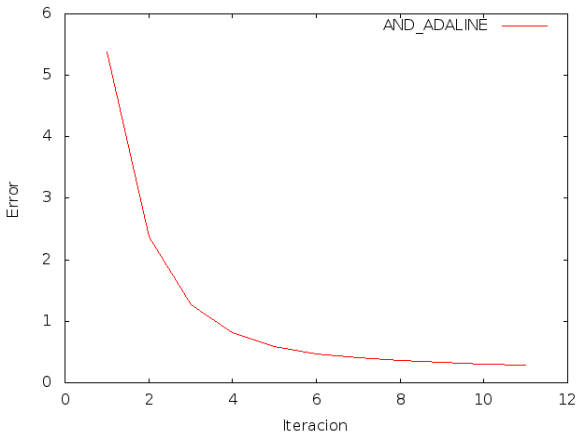
En el caso del perceptrón para las tasas de crecimiento de 0.01, 0.1, 0.2, 0.5 y 0.99 no se nota algún cambio drástico entre el tiempo que tardan entre llegar al resultado (son un máximo de 10 iteraciones)

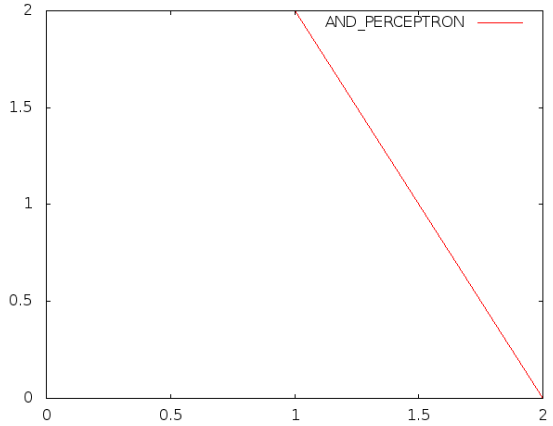
En el caso del ADALINE se observa que existe un menor tiempo de convergencia para el valor 0.1, y cualquier valor superior a ese hace que tarde más.Ademáss cabe destacar el caso OR/0.5, OR/0.99 y AND/0.99 donde la tasa es tan grande que hace que el pasa por alto el mínimo que se busca. De esto podemos concluir que ADALINE asegura convergencia hacia un mínimo para valores pequeños, pero mientras más pequeña la tasa, más tardará en llegar al mínimo.

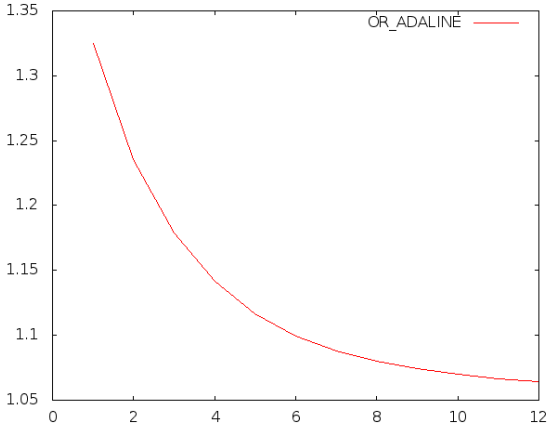
Anexos

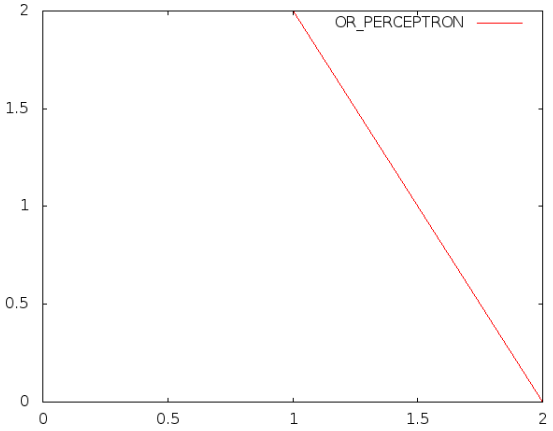
Nombre/Tasa	Tiempo en segundos
AND_PERCEPTRON,0.01	0.005
AND_PERCEPTRON,0.1	0.004
AND_PERCEPTRON,0.2	0.003
AND_PERCEPTRON,0.5	0.005
AND_PERCEPTRON,0.99	0.003
OR_PERCEPTRON,0.01	0.005
OR_PERCEPTRON,0.1	0.005
OR_PERCEPTRON,0.2	0.003
OR_PERCEPTRON,0.5	0.005
OR_PERCEPTRON,0.99	0.005

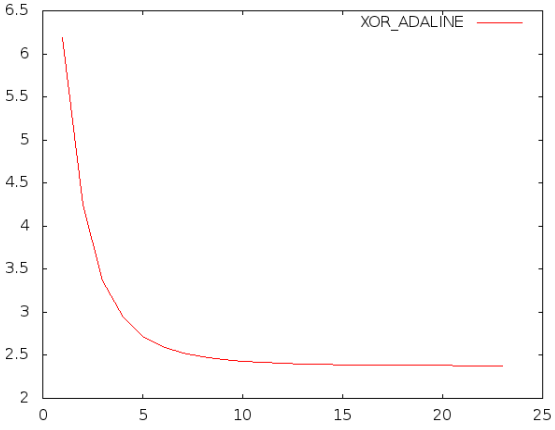
Nombre/Tasa	Tiempo en segundos
AND_ADALINE,0.01	0.006
AND_ADALINE,0.1	0.004
AND_ADALINE,0.2	0.005
AND_ADALINE,0.5	0.005
AND_ADALINE,0.99	0.009(error crece hasta infinito)
AND_ADALINE,no/i	0.143(error cambia en magnitudes de 10^-4)
OR_ADALINE,0.01	0.005
OR_ADALINE,0.1	0.003
OR_ADALINE,0.2	0.005
OR_ADALINE,0.5	0.005(error crece y converge a un er mayor al error inicial)
OR_ADALINE,0.99	0007(error crece hasta infinito)
OR_ADALINE,no/i	0.022(error cambia en magnitudes de 10^-4)
XOR_ADALINE,0.01	0.012(falla)
XOR_ADALINE,0.1	0.005(falla)
XOR_ADALINE,0.2	0.003(falla)
XOR_ADALINE,0.5	0.005(falla)
XOR_ADALINE,0.99	0.007(error crece hasta infinito)
XOR_ADALINE,no/i	1.262(error cambia en magnitudes de 10^-4)

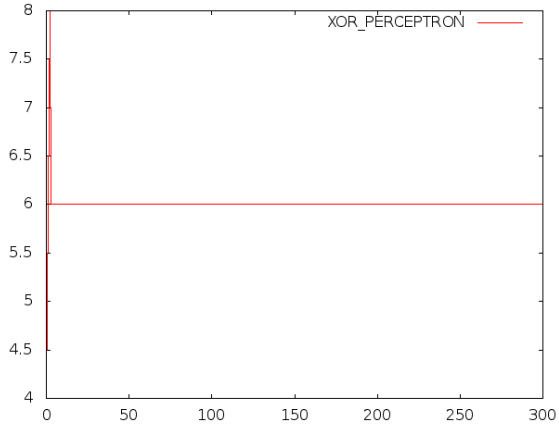

AND_ADALINE
Error
Iteracion
0
1
2
3
4
5
6
0
2
4
6
8
10
12


AND_PERCEPTRON
0
0.5
1
1.5
2


OR_ADALINE
1.05
1.1
1.15
1.2
1.25
1.3
1.35
0
2
4
6
8
10
12


OR_PERCEPTRON
0
0.5
1
1.5
2


XOR_ADALINE
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
0
5
10
15
20
25


XOR_PERCEPTRON
4
4.5
5
5.5
6
6.5
7
7.5
8
0
50
100
150
200
250
300