**Práctica 1**

Introducción a las Redes Neuronales Artificiales

3 de Marzo de 2014

**AUTORES**

Mauricio Guerreiro Quatrin

Olympia Muruzábal Ishigetani

Índice

[Neuronas de McCulloch-Pitts 2](#_Toc381615744)

[Diseño 2](#_Toc381615745)

[Resultados 3](#_Toc381615746)

[Perceptrón y Adaline 4](#_Toc381615747)

[Perceptrón 4](#_Toc381615748)

[Diseño 4](#_Toc381615749)

[Resultados 4](#_Toc381615750)

[Adaline 5](#_Toc381615751)

[Diseño 5](#_Toc381615752)

[Resultados 5](#_Toc381615753)

[Discusión y Comparación de los Resultados 5](#_Toc381615754)

# Neuronas de McCulloch-Pitts

En este ejercicio se nos pide diseñar una red con neuronas de McCulloch-Pitts, con tres neuronas en la capa de entrada (x) y dos en la capa de salida (y). Esta red ha de distinguir la orientación del estimulo en los instantes t-1 y t-2, si es hacia arriba la salida en el instante t será (0 1), si es hacia abajo la salida será (1 0) y si está quieto será (0 0).

Como consideraciones adicionales, damos por supuesto que en cada instante de tiempo no habrá más de una entrada activa. Puesto que con más de una entrada activa no sería posible observar la orientación de los estímulos de entrada.

## Diseño

Para resolver el problema planteado en este ejercicio primero obtenemos la siguiente expresión lógica para cada salida, según de la interpretación del enunciado. A partir de dicha expresión realizamos el posterior diseño de la red.



Figura Diseño de la red de McCulloch-Pitts

Siendo el umbral de las neuronas de la red 2 el peso de las entradas será 1 en el caso de las operaciones lógicas and y 2 en el caso del or. Dado el diseño podemos predecir que serán necesarios tres tiempos hasta poder predecir la primera entrada.

## Resultados

A continuación, se muestran en varios ejemplos la activación de todas las neuronas en cada unidad de tiempo. Los nodos marcados están activos, es decir adquieren un valor de 1, mientras que los que no están marcados tienen un valor de 0.

***Arriba (0 1):***

t=0 t=1 t=2

  

t=3 t=4 t=5

  

***Abajo (1 0):***

t=0 t=1 t=2

  

t=3 t=4 t=5

  

Para ejecutar el código hay que pasar como parámetros de entrada la ruta al fichero de entrada y el fichero de salida, donde se guardarán los estados de cada neurona en cada paso de tiempo. En los ficheros adjuntos resultsUp.txt, resultsDown.txt y resultsData.txt se pueden observar los estados de cada neurona en cada instante de tiempo de las pruebas realizadas.

# Perceptrón y Adaline

En este apartado realizaremos un estudio sobre las reglas de aprendizaje del Perceptrón y Adaline y su aplicación a la clasificación de patrones. Para la implementación de las dos redes nos basamos en la teoría dada en clase.

## Perceptrón

En este caso implementamos un perceptrón simple puesto que no tiene capas ocultas, solo una de entrada y otra de salida. Que a diferencia del perceptrón multicapa solo es capaz de resolver problemas linealmente separables.

### Diseño

DESCRIPCIÓN DEL DISEÑO DEFINITIVO DEL PERCEPTRÓN

### Resultados

EXPERIMENTOS REALIZADOS

Tras programar el perceptrón, hemos comprobado su eficacia frente a varias fuentes de datos, cambiando los distintos parámetros de la red, como el número máximo de épocas o la constante de aprendizaje.

Primero con patrones linealmente separables que conocemos, como la operación lógica nand, cuyas tasas de error eran 0% tal como se esperaba. Después probamos con patrones que no son separables linealmente, como xor, cuya tasa de error variaba pero nunca era del 0%.

Posteriormente realizamos pruebas con las bases de datos que se nos proporcionaban prueba\_real1.txt y prueba\_real2.txt. Dado que las tasas de error eran mayores, hemos podido deducir que los datos de ambos ficheros no son linealmente separables.

## Adaline

Adaline y el perceptrón tienen básicamente el mismo algoritmo de aprendizaje, la diferencia reside en la regla de aprendizaje, conocida como regla delta o de mínimo error cuadrático medio.

### Diseño

DESCRIPCIÓN DEL DISEÑO DEFINITIVO DE ADALINE

### Resultados

EXPERIMENTOS REALIZADOS

Realizamos las mismas pruebas que aplicamos al perceptrón con la diferencia de que ajustamos a su vez la condición de parada. Comprobamos que la condición de parada que utilizamos –el mayor cambio de peso- hace que varíe el resultado de la red. Cuanto mayor es el cambio de peso mayor es la precisión de la red, si la diferencia es muy pequeña es posible que la red no aprenda en el entrenamiento. Con los resultados obtenidos reafirmamos las observaciones encontradas con el perceptrón.

## Discusión y Comparación de los Resultados

DISCUSIÓN CON LA COMPARACIÓN ENTRE LOS RESULTADOS OBTENIDOS CON AMBAS

Ambas redes resuelven problemas linealmente separables, es decir pueden realizar una separación de clases mediante hiperplanos. Por lo que resuelven sin problemas las operaciones nand y nor, pero no es así con xor o las bases de datos problema\_real1 y problema\_real2 que no son linealmente separables.

Si añadimos entradas adicionales, como combinaciones no lineales de los atributos originales, es posible solucionar problemas como el xor, puesto que cambiamos las dimensiones del problema lo que hace posible separar las clases con hiperplanos. Pero haciendo varias pruebas con las bases de datos reales comprobamos que nos siempre se mejora la predicción, depende de las entradas añadidas.

Hemos de tomar algunas precauciones para que una red neuronal aprenda adecuadamente a generalizar y no sobreajuste ya que, si sobreaprende algunos casos, los resultados en el análisis de algunas muestras pueden ser pésimos. El sobreajuste puede darse por un número excesivo de ciclos de aprendizaje o un número reducido de patrones de entrenamiento. Una medida es un correcto entrenamiento, desordenando adecuadamente el conjunto de datos. Tras aplicar esto por primera vez en la implementación de nuestros clasificadores, los resultados han sido mucho mejores, comprobando de esta forma que realmente funcionan.

Cabe mencionar que el método del gradiente que utilizamos como algoritmo de aprendizaje nos puede llevar a un mínimo local. Para evitar esto podemos partir de inicializaciones aleatorias.

Por último, para llegar a una tasa de fallos mínima, ha sido necesario realizar un ajuste muy fino sobre alguno de los parámetros, como es la constante de aprendizaje.