**Práctica 6 (Support Vector Machines)**

¿Cómo he planteado la práctica?

Primero necesitamos los datos, en este caso leemos los .MAT ‘ex6data1’, ‘ex6data2’ y ‘ex6data3’, y separamos los datos en X e Y y Xval e Yval.

En el paso siguiente comparamos los resultados de las distintas C.

Posteriormente creamos un Kernel Gaussiano y una función que nos permita escoger los mejores parámetros.

Finalmente mostramos el resultado con los mejores parámetros obtenidos.

Código comentado y gráficas

1. Importar las librerías

Texto

Descripción generada automáticamente

1. Obtener los datos del CSV:

Texto

Descripción generada automáticamente

1. Ahora comparamos los distintos resultados con las distintas C para el primer conjunto de datos.

Gráfico, Gráfico de dispersión

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Como observación, creo que la que mejor se ajusta es C = 1, ya que es la que menos cerca está de todos los datos.

1. Ahora, para el segundo conjunto de datos, como no podemos hacer una función lineal, usaremos un kernel Gaussiano, en el enunciado especifica que C = 1 y .

Texto

Descripción generada automáticamente

Y el resultado es:

Gráfico, Gráfico de dispersión, Código QR

Descripción generada automáticamente

1. Posteriormente implementamos una función que nos devuelve los mejores parámetros C y .

Texto

Descripción generada automáticamente

La función ‘predict’ (#14) predice los datos de un X, utilizando el modelo entrenado de la Support Vector Machine, por lo cual, para conocer la precisión de nuestra inteligencia artificial, debemos comparar los valores de con los valores predichos

1. Finalmente comprobamos el resultado y dibujamos la gráfica.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

**Práctica 6 (Detector de Spam)**

¿Cómo he planteado la práctica?

Primero necesitamos los datos, en este caso leemos los directorios ‘spam’, ‘easy\_ham’ y ‘hard\_ham’. Para ello nos apoyaremos en una función

En el paso siguiente, juntamos las X de los anteriores archivos y las dividimos en dos subconjuntos, y .

Posteriormente creamos las Y, con el mismo tamaño que y .

Finalmente entrenamos la Inteligencia Artificial y obtenemos su porcentaje de acierto.

Código comentado y gráficas

1. Importar las librerías

Texto

Descripción generada automáticamente

1. Creamos una función que nos devuelva la X a partir de varios archivos, cada uno tiene varios emails:

Texto

Descripción generada automáticamente

Primero obtenemos el diccionario de vocabulario (#2) y leemos todos los archivos(#5). Después inicializamos un array de ceros(#15), en el que, en cada entrada del diccionario(#17), comprobamos si existe en el email leído(#18), si existe, lo ponemos a 1(#19), si no, nada. Finalmente devolvemos todas las X en un array(#24).

1. Leemos los archivos ‘spam’, ‘easy\_ham’ y ‘hard\_ham’.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Ahora establecemos un tamaño para el Train Set, en este caso será de un 80%, frente a un 20% de Cross Validation.

Y separamos el Train Set (#9) del CV Set (#13)

Texto

Descripción generada automáticamente

1. Posteriormente reutilizamos la función ‘parameter\_election’ de la anterior parte de la práctica, esta función nos devolvía la , la y la .



*NOTA: Lógicamente usamos el Kernel Gaussiano, ya que la distribución de datos no puede ser lineal.*

1. Finalmente mostramos la precisión obtenida:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente