# Práctica 6: Support Vector Machines

Esta práctica consiste en emplear el clasificador SVM en una parte, y luego emplearlo de forma práctica en la segunda parte.

## Primera Parte

La primera parte de la práctica consiste en utilizar la función SVN para clasificar 2 conjuntos de datos. Para ello basta con cargar los datos y llamar a la función “svmTrain”.

Después, implementaremos la función del kernel gaussiano, que es tan fácil como desarrollar la fórmula del mismo. Luego utilizaremos este kernel para entrenar una SVN que clasifique el segundo conjunto de datos.

Por último, veremos la importancia de elegir un C y un sigma determinados, y veremos la diferencia en precisión de los mismos. Esto lo haremos creándonos una función “seleccionarParametros” que lo que hará será ir probando todas las posibles combinaciones de C y sigma (entre las que nosotros le demos), e ir calculando la precisión de cada una. Al final, se quedará con la combinación de C y sigma que minimice el error. Y así veremos que el resultado de emplear estos atributos es mejor que el calculado anteriormente.

## Segunda Parte

La segunda parte de la práctica consiste en utilizar estas mismas técnicas para llevar a cabo experimentos en la detección de spam en el correo electrónico.

Para ello lo primero que tenemos que hacer es generar datos de entrenamiento válidos. Lo que hemos decidido es implementar una función “crearDataSet” que dado un directorio y un número de mails, lee tantos ejemplos como le hayamos dicho, saca los atributos de los mismos, y lo convierte en una matriz “X”.

Para ello lo primero es llamar a la función “processEmail “, que nos devolverá el texto del email. Este texto se o pasaremos a la función “vectorAtributosEmail “, que se encarga de generar el vector X de ese ejemplo de entrenamiento concreto. Esto será así para cada uno de los ejemplos de entrenamiento (emails).

La función “vectorAtributosEmail “es quizás la más compleja (y que consume más tiempo de proceso). Lo que hace esta función es:

1. Calcula el número de palabras del “diccionario” almacenado en un Array (obtenido por medio de getVocabList();
2. Almacenamos este diccionario en una estructura que nos servirá de hash-map
3. Comprobamos una a una, si cada palabra del email se encuentra en el diccionario. Si esta palabra se encuentra en el diccionario, entonces guardaremos un “1” en la posición que ocupa esa palabra en el diccionario.

De esta manera nos quedará un vector de 1’s y 0’s (de las dimensiones del diccionario) en las que 1 representa que la palabra está en el email, y 0 que no está.

Al final, la matriz de ejemplos de entrenamiento X será una matriz de dimensiones (nº ej. Entrenamiento x nº palabras en el diccionario). Además, deberemos generar un vector “y” que nos clasifique si los ejemplos de entrenamiento son spam o no. Esto lo haremos simplemente sabiendo de qué fichero es cada ejemplo de entrenamiento (directorio).

Una vez tengamos las entradas X e y, emplearemos dichos datos para generar un entrenamiento mediante SVN con kernel lineal de algunos de ellos de forma aleatoria. Además podremos medir la precisión del modelo comparando los resultados con unos datos de test que no hayamos utilizado en el entrenamiento.

De esta manera podemos ver que la precisión de nuestro modelo es bastante elevada: