Práctica 6:

Support Vector Machines

Grupo 13:

David Ortiz Fernández.

Andrés Ortiz Loaiza.

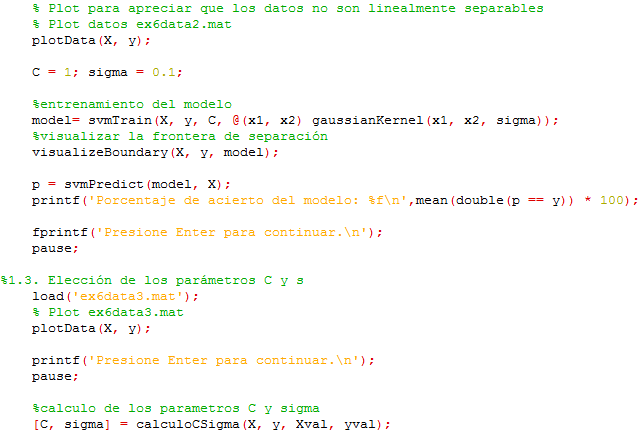
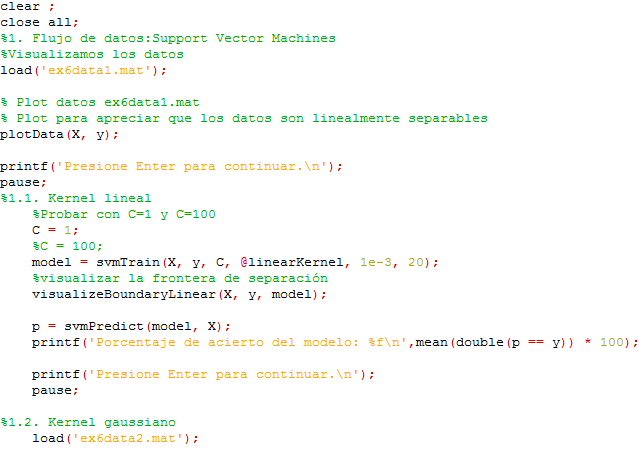
1. **Support Vector Machines**

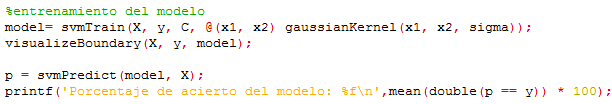
En esta primera parte de la práctica, se hace una introducción a SVM. Se ha comenzado comprobando como afecta el parámetro C a una SVM con kernell lineal y usando unos datos linealmente separables.

A continuación, se ha desarrollado un kernell gausiano y se han usado para comprobar su uso unos valores de C y sigma de 1 y de 0.1 respectivamente, en la gráfica obtenida se ha podido ver como se creaba una frontera de separación correcta en el calificador.

Po ultimo hemos calculado el porcentaje de acierto del modelo: 98.841251, el cual nos indica la calidad de nuestro clasificador para unos datos no linealmente separables. Seguidamente hemos creado una función para seleccionar los valores de C y sigma óptimos, tras lo que hemos obtenido un acierto de: 94.786730.

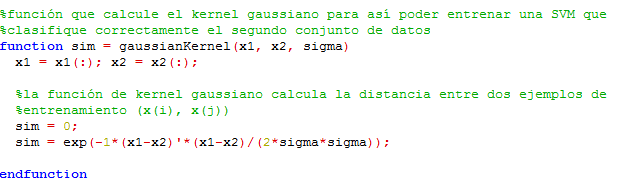
**Script principal**

****

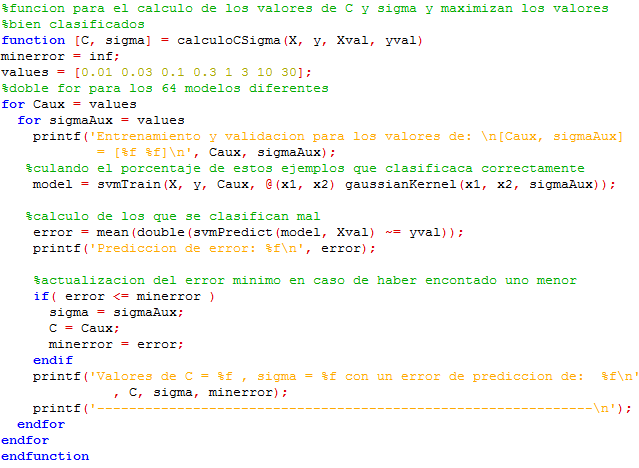


**Funciones :**

**gaussianKernel**

****

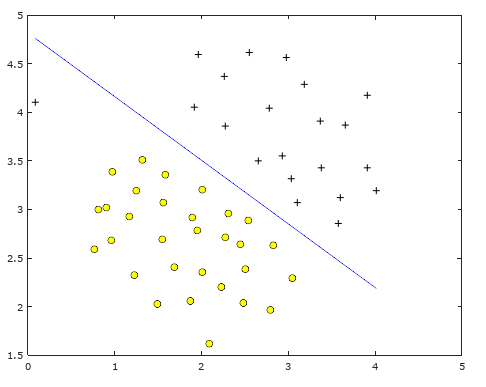
**calculoCSigma**

****

**Resultados:**

**1.1 Kernel lineal**

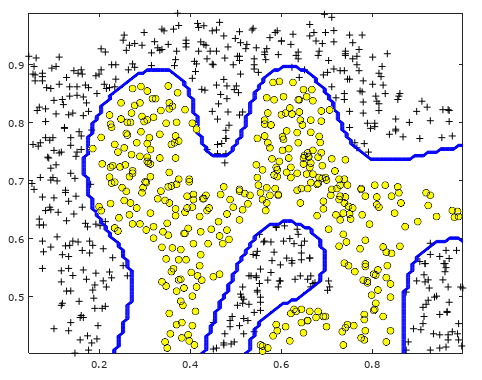
La siguiente gráfica muestra la frontera lineal definida por el modelo de SVM calculado con el kernel lineal para C = 1



Porcentaje de acierto del modelo: 98.039216

**1.2 Kernel gaussiano**

La siguiente gráfica muestra la frontera no lineal definida por el modelo de SVM calculado con el kernel gaussiano para C = 1 y sigma = 0.1

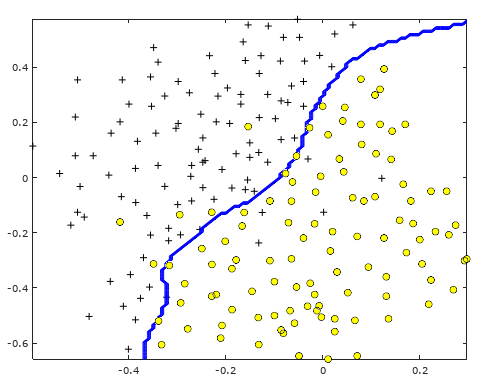


Porcentaje de acierto del modelo: 98.841251

.

**1.3 Kernel gaussiano**

La siguiente gráfica muestra la frontera no lineal definida por el modelo de SVM calculado con el kernel gaussiano para C y sigma calculados tomando valores del conjunto 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, generando así un total de 8 2 = 64 modelos diferentes para su cálculo.



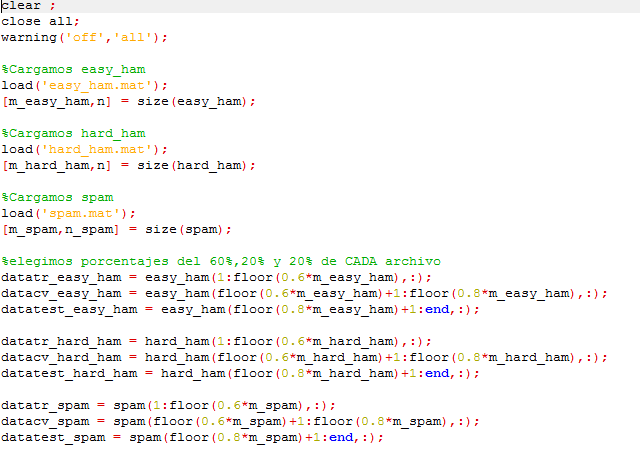
Porcentaje de acierto del modelo: 94.786730

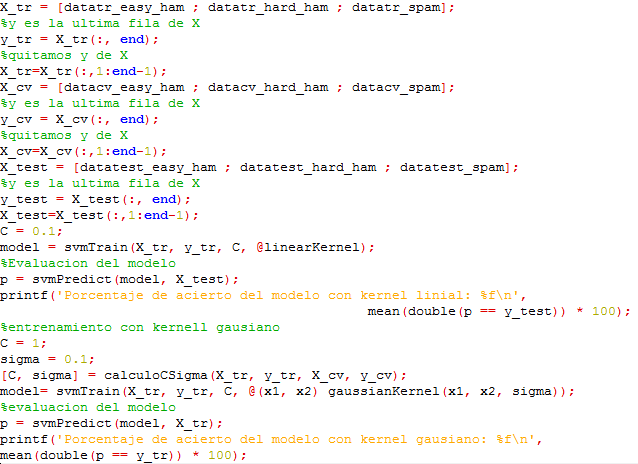
**2. Detección de spam**

En esta segunda parte nos hemos dispuesto usar una SVM para clasificar correos como spam y no spam. Hemos creado un script que lee todos los ficheros, los procesa y crea gracias a un diccionario un vector con las palabras típicas que identifican correos con spam. Con esto, la función crea 3 ficheros que contienen todos estos datos, un fichero para conjunto, easyham.mat, hardham.mat y spam.mat.

Tras esto hemos aplicado los conocimientos de la primera parte de la práctica y realizar así predicciones, para ello dividiendo los archivos por cross-validation. Este es el código de la función que se ha implementado:

**Script principal**





**Generación archivos**

