Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Ingeniería Informática
Especialidad: Computación
Cuarto curso. Primer cuatrimestre

Introducción a los modelos computacionales.

Práctica 4: Máquinas de vectores soporte

Antonio Ariza García
DNI
i62argaa@uco.es

Curso académico 2020-2021
Córdoba, 23 de septiembre de 2021

Índice

Índice de figuras

Índice de tablas

1. Bases de datos sintéticas

En la primera parte de la práctica utilizaremos una serie de experimentos sobre bases de datos sintéticas, que nos ayudarán a entender cómo funcionan las SVM y qué efectos producen sobre los resultados los dos parámetros que hay que especificar para entrenarlas.

1.1. Trabajando con el primer dataset, pregunta 1, 2 y 3

Este script comienza importando las librerías necesarias como es normal, en este caso numpy, matplotlib, pandas y sklearn para el acceso a SVM. A continuación lee el CSV.

Podemos comprobar como el kernel que se utiliza es 'lineal' y el parámetro C=1000.

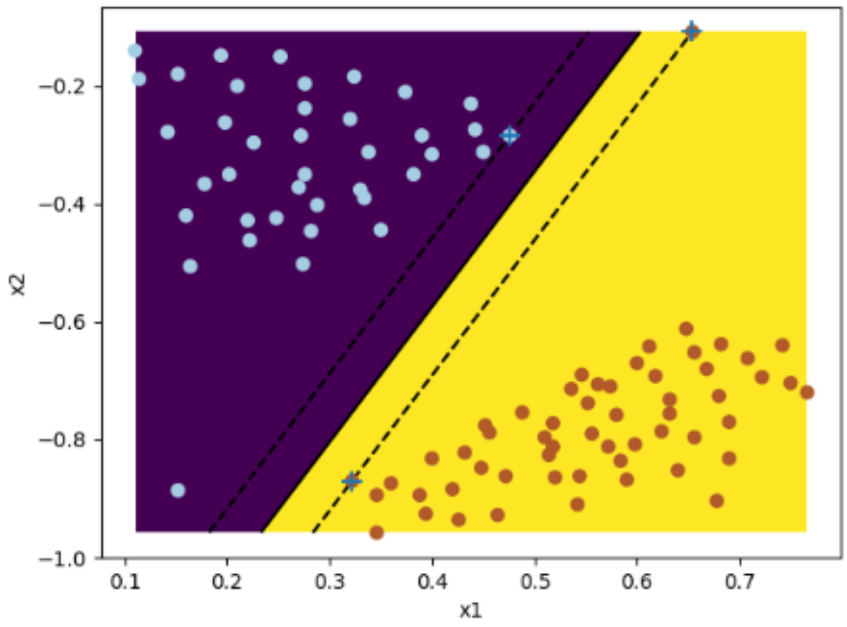


Figura 1: Dataset1, kernel lineal

En la Figura 1 tenemos las clases separadas mediante un hiperplano se-

parador lineal.

La siguiente figura2 corresponde solo a la nube de puntos del dataset1, es evidente que la manera más fácil de separar las clases es mediante un recta.

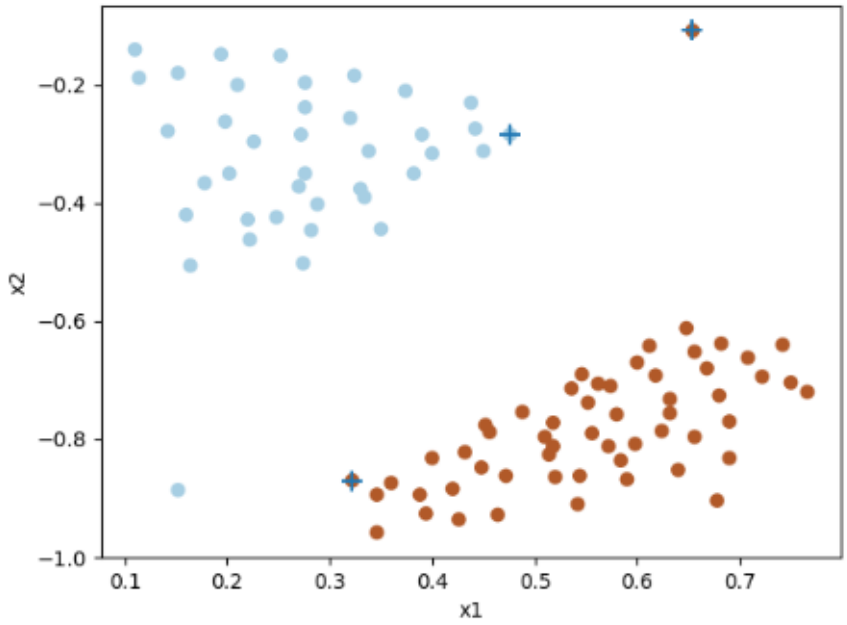
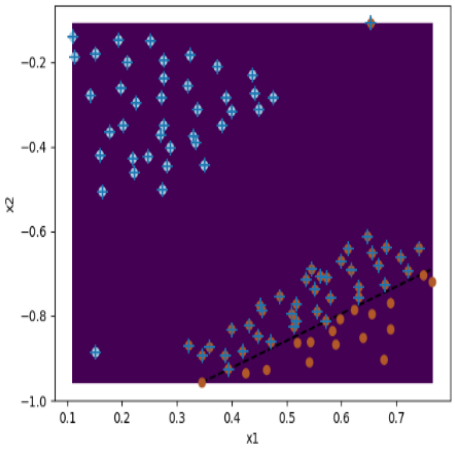
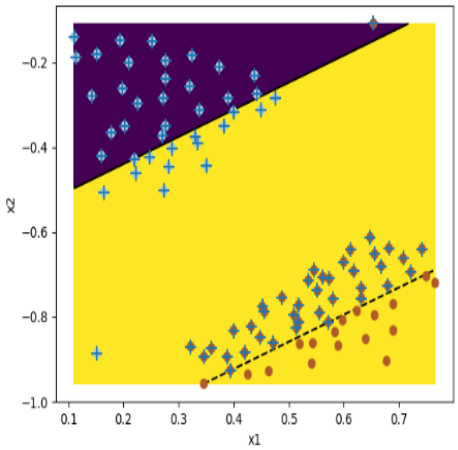


Figura 2: Nube de puntos dataset1

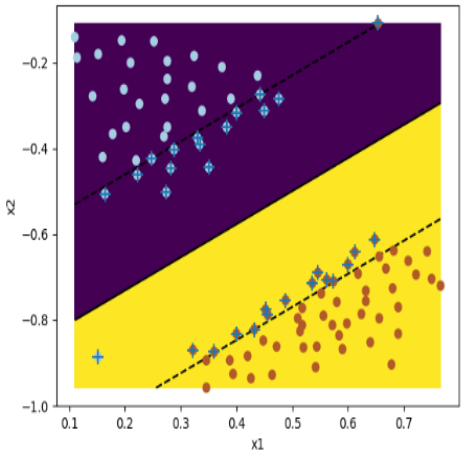
A continuación vamos a modificar el valor del parámetro C, podemos consultar su utilidad en `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`, el parámetro C es utilizado como regulador, define la penalización de que e cometan errores en la clasificación, la regularización es inversamente proporcional a C, un valor muy alto de C conllevará a un SVM que cometa el mínimo número de errores. Un valor de C pequeño por el contrario conllevará a un clasificador con el máximo margen, aunque se cometan errores en la clasificación. Por supuesto está relacionado con el posible sobre-entrenamiento.



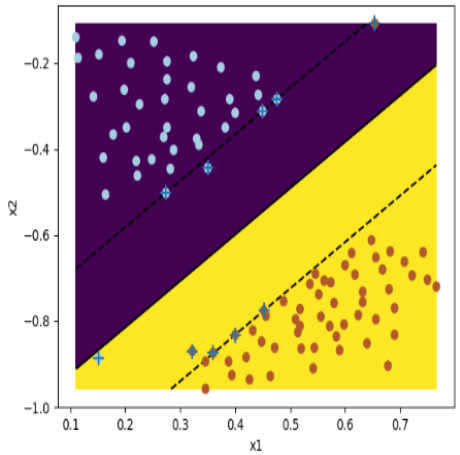
(a) $C = 10^{-2}$



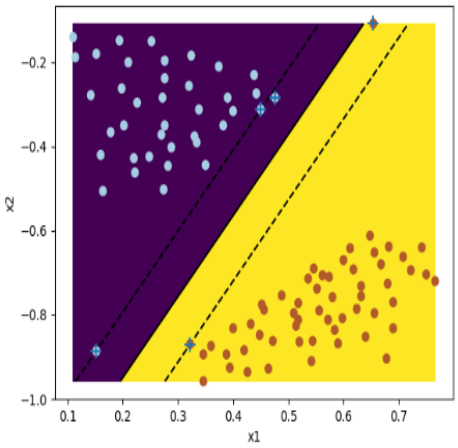
(b) $C = 10^{-1}$



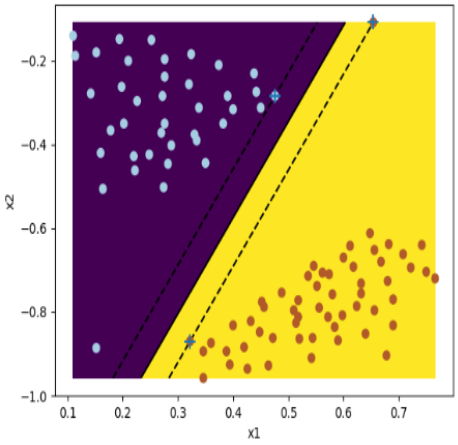
(c) $C = 10^{0}$



(d) $C = 10^{1}$



(e) $C = 10^{2}$



(f) $C = 10^{3}$

Figura 3: Variaciones del parámetro C

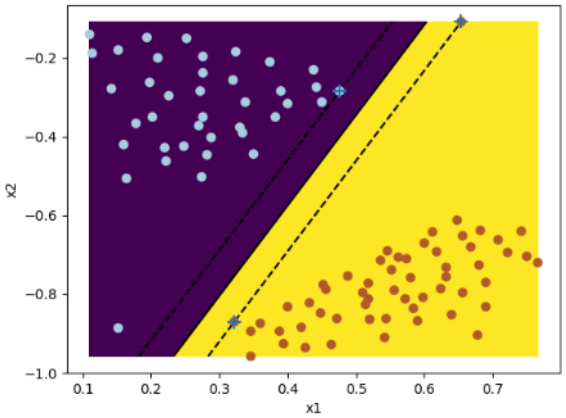


Figura 4: $C = 10^4$

1.2. Trabajando con el segundo dataset, preguntas 4 y 5

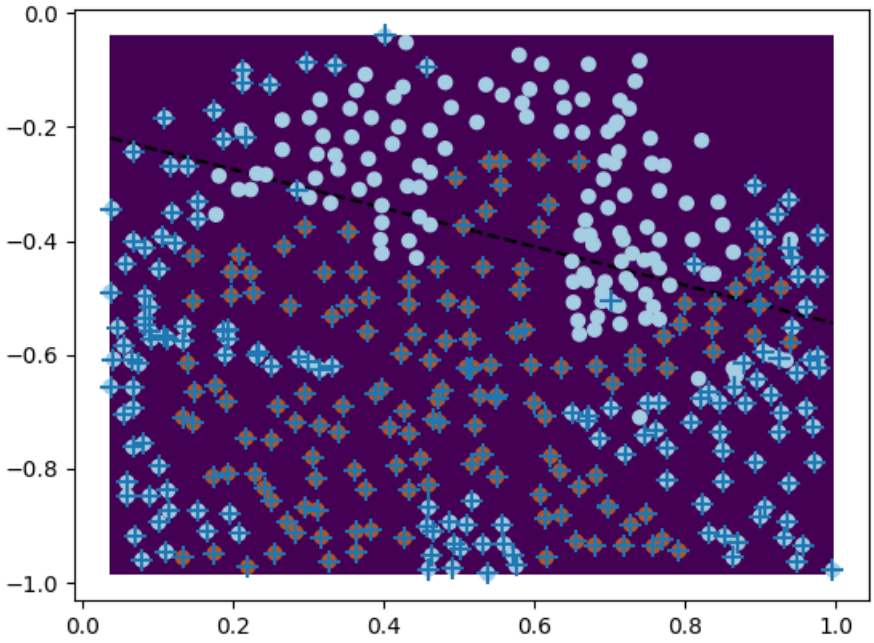


Figura 5: Dataset2 SVM lineal y $C = 1000$

La figura5 representa al segundo dataset con una SVM lineal y un parámetro $C = 1000$, en ella podemos comprobar como es imposible separar las clases mediante una línea, por mucho que modifiquemos el parámetro C.
Mediante el uso de funciones de kernel tipo RBF podemos conseguir trabajar con este dataset, este kernel necesita un parámetro gamma que iremos introduciendo hasta encontrar una buena solución. En libsvm $\gamma = 1/2 * radio^2$, para un radio alto tiende a soluciones más suaves, con menor sobre-entrenamiento, mientras que para un radio pequeño tiende a producir mas sobre-entrenamiento.

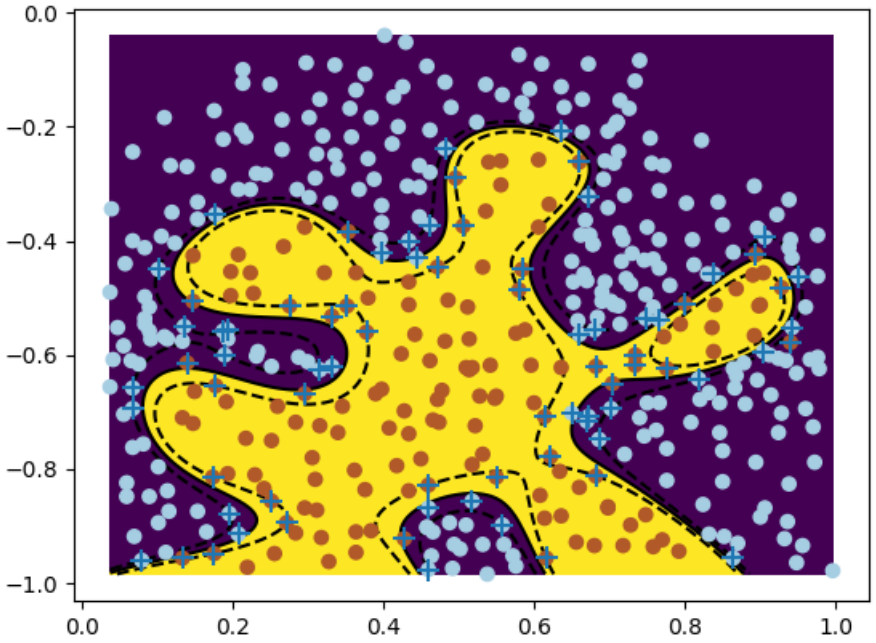
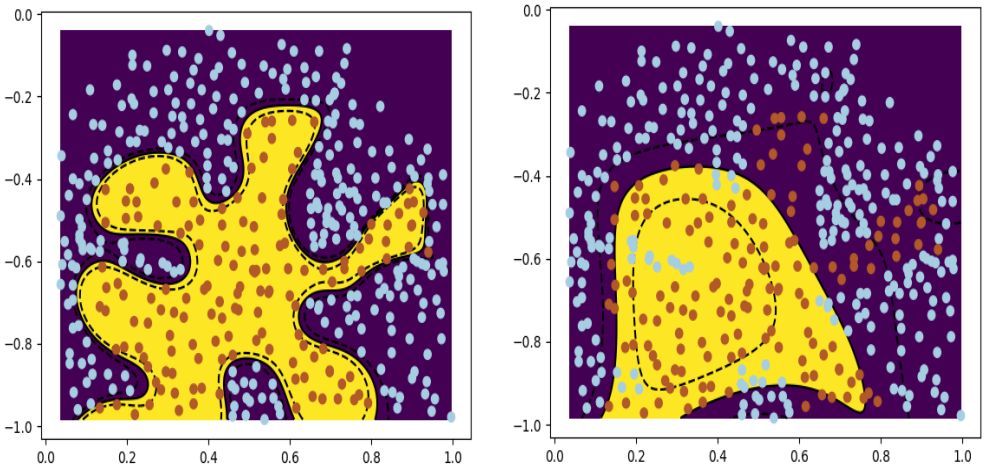


Figura 6: Dataset2 RBF, $C = 1000$ $\gamma = 13$



(a) $C = 1000$ $\gamma = 50$ (b) $C = 1000$ $\gamma = 2$

Figura 7: Sobreentrenamiento e infraentrenamiento

1.3. Tercer dataset, preguntas de la 6 a la 11

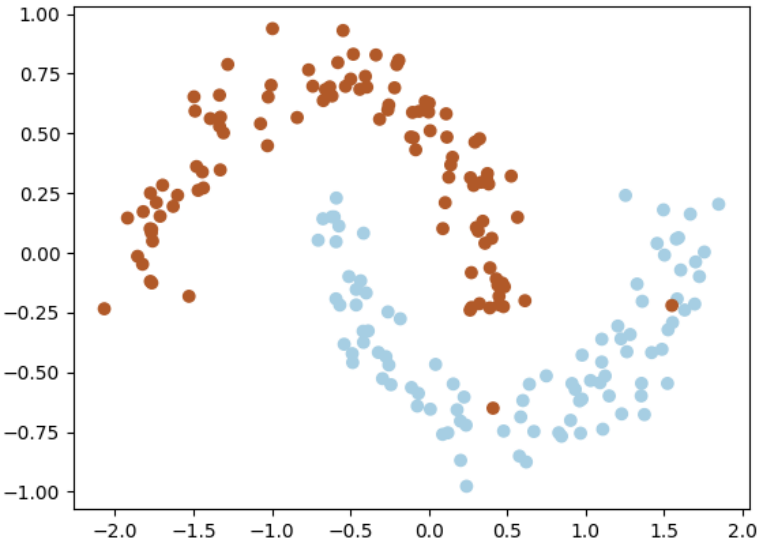


Figura 8: Dataset3 RBF, $C = 1000$ $\gamma = 1$

Comprobamos Fig8 que este dataset tampoco es linealmente separable, volvemos a utilizar un kernel de tipo RBF. Hay un par de puntos que se salen

de la densidad de la clase y se introducen en la otra, podríamos pensar que se tratan de outliers.

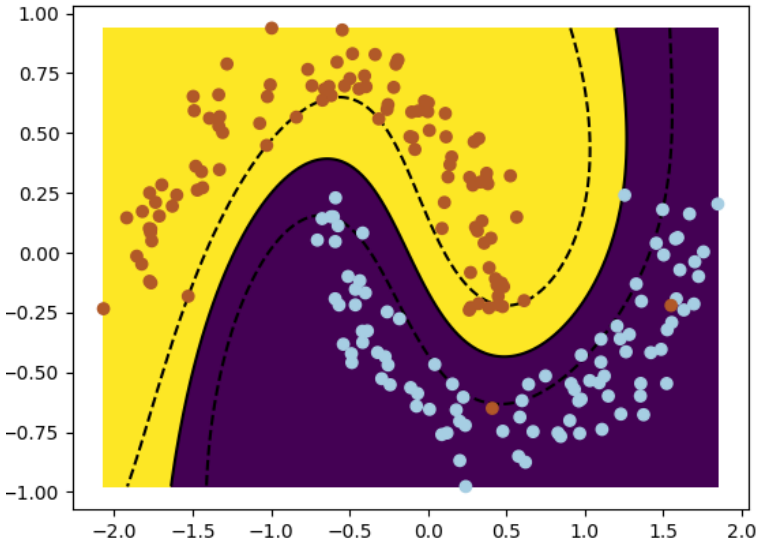
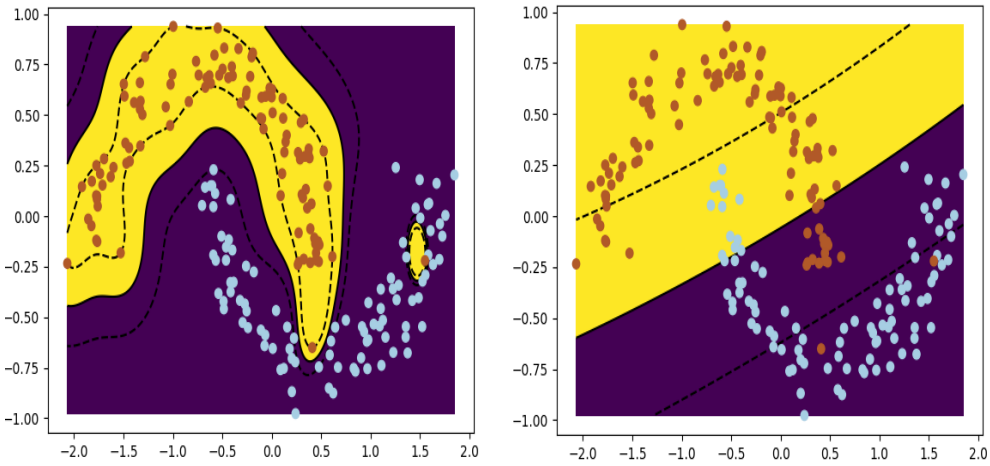


Figura 9: Dataset3 bien clasificado, RBF, $C = 10$ $\gamma = 1$



(a) $C = 1000$ $\gamma = 10$ (b) $C = 10$ $\gamma = 0.1$

Figura 10: Sobre-entrenamiento e infra-entrenamiento

1.4. Base de datos noMNIST

Esta base de datos está formada por 900 patrones de entrenamiento y 300 patrones de test. Está formada por un conjunto de letras (de la a hasta la f), están ajustadas a una rejilla cuadrada de 28 x 28 píxeles.

El proceso de validación cruzada se repite completo para todas las combinaciones posibles de parámetros C y γ, en este caso $7X7X5 = 245$. Obteniendo un CCR de 81,77 %.

A continuación en la tabla 2 indicamos el CCR obtenido junto con su tiempo para distintos tamaños de k-fold.

K	CCR	Tiempo
3	82,22 %	11.4 s
5	81,77 %	25.01 s
10	81,77 %	66.42 s

Tabla 1: Resultados validación cruzada K-fold

Comprobamos como el mejor resultado ha sido para $K = 3$, además también se consigue con menos tiempo.

1.5. Base de datos Clasificación de SPAM

Uno de los campos donde se utiliza el aprendizaje automático con un mayor éxito es la detección automática de spam en servidores de correo. El fichero de entrenamiento contiene 4000 correos, mientras que el de test tiene 1000 correos. Ambos utilizan la lista de vocabulario de 1899 palabras contenida en vocab.txt. Por tanto, cada patrón tiene 1899 valores binarios.

Entrenaremos un modelo lineal de SVM con distintos valores de C.

C	CCR
0.01	98 %
0.1	98.9 %
1	97.8 %
10	97.5 %

Tabla 2: Resultados bases datos SPAM