# Práctica 2
# **Máquinas de Vectores Soporte**

Vicente Gras Mas

# Introducción

El objetivo de esta práctica es experimentar con las máquinas de vectores soporte en la plataforma Octave usando la librería LibSVM.

Emplearemos esta técnica de entrenamiento y clasificación con dos tipos de datos, unos linealmente separables ( trSep.dat y trSeplabels.dat) y otros no separables linealmente ( tr.dat y trlabels.dat ).
Por último haremos un experimento con las máquinas de vectores soporte con los datos de la práctica anterior (MNIST) variando el tipo de kernel utilizado y el valor de C para ver que combinaciones serían las más prometedoras.

# Ejercicio 3

Este ejercicio contará con dos subapartados, uno para el conjunto de datos separable linealmente y otro para un conjunto de datos no separable linealmente. A continuación se resolverá primero el separable y luego el no separable:

## Datos linealmente separables:

1. Obtener una SVM sin kernel (ses decir, kernel tipo lineal). Para simular la optimización estandar del caso separable basta usar un valor grande de C.

Para obtener dicha SVM usaremos la función svmtrain con las matrices de datos como parámetros y la opción -t 0 indica que se usará un kernel de tipo lineal, siendo la instrucción tal que así:

```
res = svmtrain(trlabels, tr, '-t 0 -c 1000');
```

2. Determinar a) los multiplicadores de Lagrange, alpha, asociados a cada dato de entrenamiento, b)  los vectores soporte, c) el vector de pesos y umbral de la función discriminante lineal, y d) el margen correspondiente.

Todos estos datos están dentro del objeto res, accediendo a ellos obtenemos los apartados siguientes:
a) Multiplicadores de Lagrange:

Se accederá a ellos con res.sv_coef.

0.87472
0.74989
-1.62461

b) Vectores soporte:

Se accederá a ellos con tr(res.sv_indices,:).

| | |
|---|---|
| 1 | 4 |
| 4 | 2 |
| 3 | 4 |

c) Vector de pesos y umbral de la función discriminante lineal:

El vector peso es el producto entre los multiplicadores de Lagrange y los vectores soporte.

-0.99955  -1.49977

El umbral lo hemos obtenido usando svmtrain y lo tendremos en el campo rho, pudiendo acceder a él con res.rho, y poniendo dicho valor en valor absoluto, obtendremos que vale 7.9981.

d) Margen

El margen lo podemos obtener substituyendo en la fórmula vista en teoría:

margen = 2/norm(Vector de Pesos)

Obteniendo su valor, el cual será 1.1097.

3. Calcular los parámetros de la frontera lineal (recta) de separación.

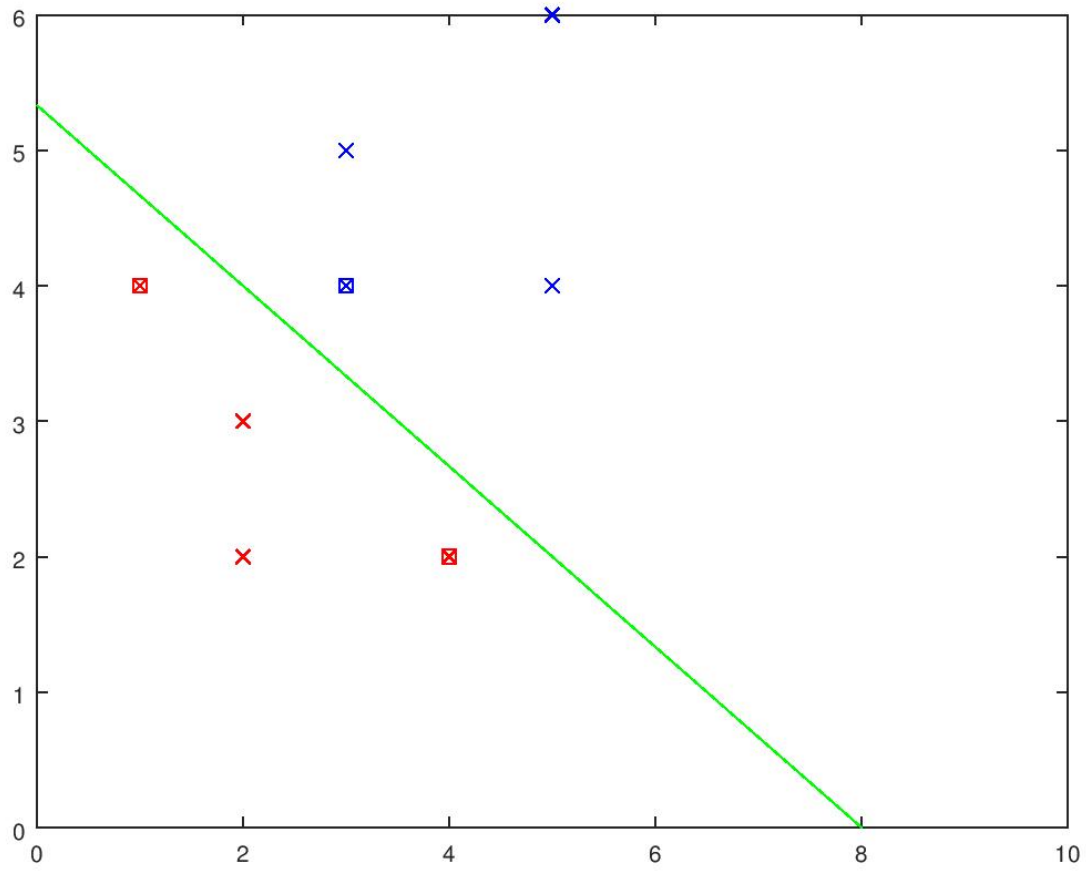
Calculando los cortes con el eje X y eje Y los cuales se sabrán tomando el valor de umbral / VectorPesos (1 o 2 dependiendo de que eje) obtenemos los puntos para trazar la recta de separación de clases:

```
Px = [res.rho/peso(1,1) 0];
Py = [0 res.rho/peso(1,2)];
```

Siendo peso el vector peso.
De este modo conseguimos los puntos para trazar la recta los cuales són: (8.00166  0) - (0  5.33283).

4. Representar gráficamente los vectores de entrenamiento, marcando los que son vectores soporte, y la recta separadora correspondiente.

En la gráfica mostrada a continuación, podemos ver la frontera de decisión (linea verde), los puntos de cada clase (X rojas y X azules) y los puntos que forman parte de los vectores soporte de cada clase (Las X dentro de un cuadrado de su mismo color).




A continuación haremos el mismo ejercicio con los datos no linealmente separables, para no repetir información, nos ahorraremos la explicación de cómo hemos resuelto los apartados que ya se han explicado en la resolución anterior (Puntos 1, 2, 3, 4).

## Datos linealmente No Separables:

1. Obtener una SVM sin kernel (ses decir, kernel tipo lineal). Para simular la optimización estandar del caso separable basta usar un valor grande de C.

```
res = svmtrain(trlabels, tr, '-t 0 -c 1000');
```

2. Determinar a) los multiplicadores de Lagrange, alpha, asociados a cada dato de entrenamiento, b)  los vectores soporte, c) el vector de pesos y umbral de la función discriminante lineal, y d) el margen correspondiente.

a) Multiplicadores de Lagrange:

0.87472
0.74989
-1.62461

b) Vectores soporte:

| | |
|---|---|
| 1 | 4 |
| 4 | 2 |
| 3 | 4 |

c) Vector de pesos y umbral de la función discriminante lineal:

-0.99955  -1.49977

Umbral: 7.9981

d) Margen

1.109.

2.1 Determinar los valores de tolerancia de margen, , asociados a cada dato de entrenamiento.

Para determinar dichos valores, aplicaremos la fórmula vista en teoría para ver la tolerancia de un margen y poder aplicarla a la frontera de decisión para poder dibujarla en nuestra representación gráfica:

```
tolerancia = 1 - sign(res.sv_coef').*(peso*Vsop' - res.rho);
```

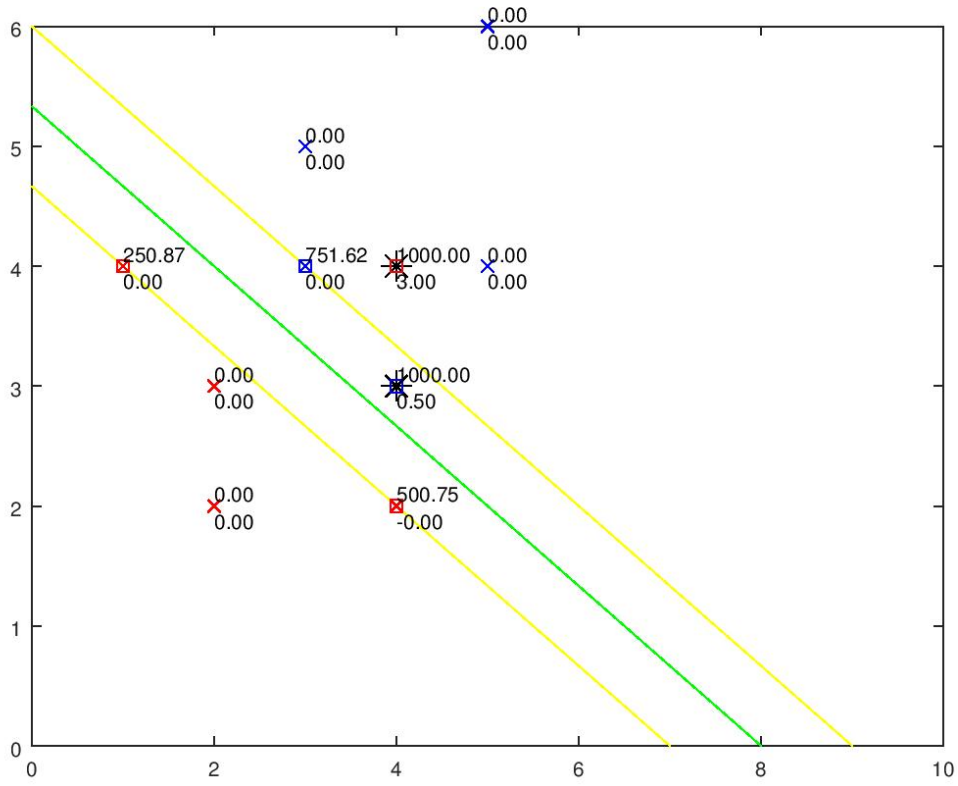
Donde peso*Vsop será el peso que se le da a cada Vector soporte.

3. Calcular los parámetros de la frontera lineal (recta) de separación.

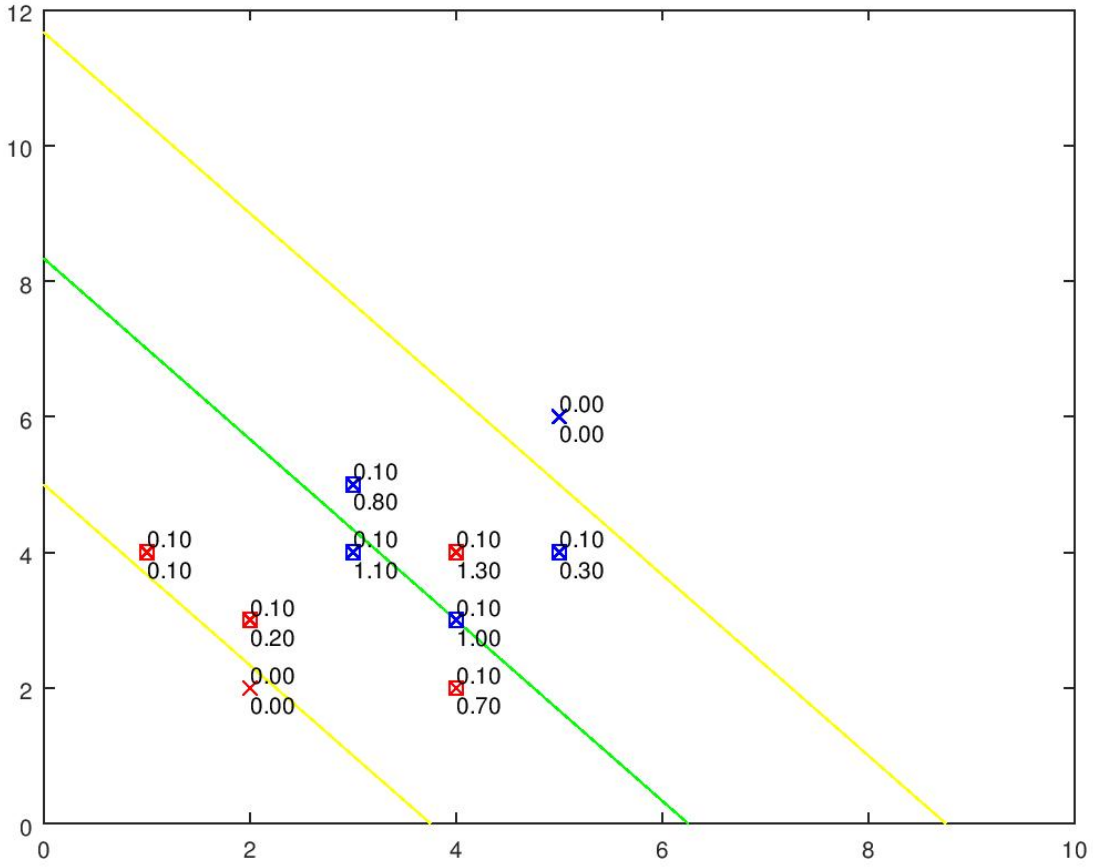
Cortes con los ejes para definir la frontera:
(8.00166  0) - (0  5.33283)

4.1 En la representación gráfica, marcar los vectores soporte \ erróneos";
4.2 Obtener los resultados (gráficas) indicados anteriormente para diversos valores relevantes de C.

En la gráfica mostrada a continuación se pueden ver la frontera de decisión (Linea verde), los margenes de dicha frontera (lineas amarillas), los puntos clasificados en sus clases (X rojas y X azules) y los puntos que son vectores soporte (Cuadrado alrededor del punto), además podemos ver los vectores soporte que son erróneos (Tachados con un asterisco negro) y sus valores de tolerancia asociados a cada punto.



A continuación se muestra la tabla con un c = 0.1, como se puede observar, el resultado es peor que la tabla anterior (c = 1000)




## Ejercicio 4

Se propone la realización de un experimento de clasificación para determinar los parámetros del clasificador, al menos el tipo de kernel y el valor de C, que obtienen mejores resultados. Se deberá presentar una tabla y/o gráfica donde se muestre no solo el mejor resultado obtenido, sino también otros resultados relevantes que permitan poner de manifiesto la tendencia a mejorar o empeorar del modelo según varían los parámetros considerados.

Usaremos el script Ejercicio4.m el cual contiene 2 bucles para variar tanto la t (kernel usado) como la C, como hemos usado a lo largo de

esta práctica c = 1000 ($10^3$), usaremos como una C muy baja, c= 0.0001 (10^-3).
También se han ploteado los datos (mediante plot_ej4.plt, generando Ej4.dat). Se ha decidido no usar dicha grafica ya que a la vista de los datos, hay varias pruebas que coinciden en resultados y por tanto los datos en la gráfica no quedaban nada claros.
A continuación se mostrarán los datos conseguidos mediante el experimento y se procederá a dar una valoración de las observaciones:

Kernel lineal

| Valores C | % de acierto | Valor de confianza superior | Valor de confianza inferior |
|---|---|---|---|
| 0.0001 | 93.33 | 0.07 | 0.06 |
| 1 | 83.97 | 0.15 | 0.15 |
| 1000 | 83.97 | 0.15 | 0.15 |

Kernel polinomial

| Valores C | % de acierto | Valor de confianza superior | Valor de confianza inferior |
|---|---|---|---|
| 0.0001 | 97.91 | 0.02 | 0.01 |
| 1 | 97.91 | 0.02 | 0.01 |
| 1000 | 97.91 | 0.02 | 0.01 |

Teniendo en cuenta que por defecto el kernel polinomial usa un polinomio de grado 3, hemos hecho varias pruebas variando el parámetro -d, para ver cómo se comporta dicho kernel con más grados y con menos grados de polinomio. Los datos de la siguiente tabla se recogen en el archivo Ej4t1.dat. Concretamente hemos probado los grados polinomiales 1, 5 y 8:

| Valores C | % de acierto | | | Valor de confianza superior | | | Valor de confianza inferior | | |
|---|---|---|---|---|---|---|---|---|---|
| | grado (-d) | | | grado (-d) | | | grado (-d) | | |
| | 1 | 5 | 8 | 1 | 5 | 8 | 1 | 5 | 8 |
| 0.0001 | 94.25 | 96.65 | 93.64 | 0.06 | 0.04 | 0.07 | 0.05 | 0.03 | 0.06 |
| 1 | 94.25 | 96.65 | 93.64 | 0.08 | 0.03 | 0.07 | 0.07 | 0.003 | 0.06 |
| 1000 | 94.25 | 96.65 | 93.64 | 0.06 | 0.04 | 0.07 | 0.05 | 0.02 | 0.06 |

## Kernel RBF

| Valores C | % de acierto | Valor de confianza superior | Valor de confianza inferior |
|---|---|---|---|
| 0.0001 | 11.35 | 0.9 | 0.9 |
| 1 | 11.35 | 0.9 | 0.9 |
| 1000 | 11.35 | 0.9 | 0.9 |

## Kernel Sigmoide

| Valores C | % de acierto | Valor de confianza superior | Valor de confianza inferior |
|---|---|---|---|
| 0.0001 | 11.35 | 0.9 | 0.9 |
| 1 | 11.35 | 0.9 | 0.9 |
| 1000 | 11.35 | 0.9 | 0.9 |

El mejor clasificador es el que usa kernel polinomial por defecto (grado 3). Variando el grado del polinomio del kernel polinomial conseguimos también muy buenos resultados, aunque no tan buenos como el kernel polinomial de grado 3.
El segundo mejor clasificador será el kernel lineal con la menor c posible.
El kernel RBF y el kernel Sigmoide se comportan igual, siendo su comportamiento muy malo y nada aceptable.