PRACTICA 1 APR

Adrian Tendero Lara

1. Ejercicio de entrenamiento Completo

En este ejercicio básicamente tenemos que hacer todo paso a paso para dos conjuntos de entrenamiento en dos dimensiones que son *mini/TrSep* y *mini/Tr* de estos dos solo el primero es linealmente separable.

El ejercicio propone realizar varias tareas para los dos conjuntos, y unas tareas extra para el conjunto no-separable. La Primera cosa que tenemos que hacer es sacar el SVM con kernel de tipo lineal de ambos conjuntos.

Para ello se tiene que realizar el siguiente proceso:

```
load data/mini/tr.dat

load data/mini/trlabels.dat

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

Con esto ya tenemos cargado y entrenado uno de los conjuntos de entrenamiento, este es el no separable, el otro se carga igual pero cambiando el nombre del archivo ha cargar trsep.data y trseplabels.data.

Dentro de la variable res ahora se encontrará tanto los multiplicadores de lagrance, los vectores soporte y el umbral (w0):

```
lagrange = res.sv_coef;

vector_soporte = tr(res.sv_indices,:)

w0 = -res.rho;
```

Y esto nos permite calcular fácilmente el vector de pesos (w) y el Margen

```
w = lagrange' * vector_soporte;

margen = 1 / (w * w');
```

Con lo cual nos permite calcular la recta de separación (Frontera de separación) y los márgenes fácilmente:

```
m = - w(1)/w(2);

b = - w0 / w(2); (recta)

b1 = - (w0 - 1) / w(2); (margen 1)

b2 = - (w0 + 1) / w(2) (Margen 2)

X = [1:0.001:7]; (Espacio de representación)

Y = m * X + b ; (Y de la recta)

Y1 = m * X + b1; (Y del margen 1)

Y2 = m * X + b2; (Y del margen 2)
```

Y lo representamos

```
plot(X, Y, X, Y1, X, Y2, tr(trlabels==1, 1), tr(trlabels==1, 2), 'o',
tr(trlabels==2, 1), tr(trlabels==2, 2), 'x', tr(res.sv_indices, 1),
tr(res.sv_indices, 2), '+');
```

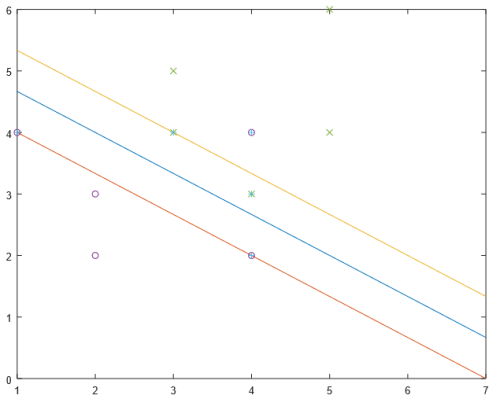
Con esto ya tenemos todo lo que se nos pide para el conjunto de datos separable pero para el conjunto no separable nos piden además determinar los valores de tolerancia del margen para cada dato de entrenamiento e indicar en la representación grafica los vectores de soporte erróneos asi como mostrar los resultados entrenando las muestras con C diferentes, esto lo realizo de la siguiente manera:

```
tolerancia_margen_sv = (abs(lagrange) == C) .* (1 - sign(lagrange) .*
(vector_soporte * w' + w0));

tolerancia_margen = zeros(size(trlabels));

tolerancia_margen(res.sv_indices) = tolerancia_margen_sv;
```

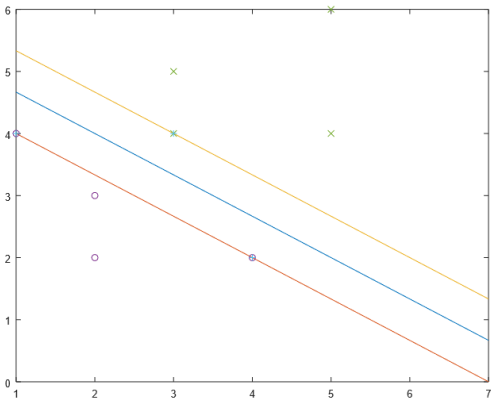
Y lo represento asi:

```
plot(X, Y, 'g', X, Y1, 'b', X, Y2, 'r', tr(trlabels==1,1), tr(trlabels==1,2),
'sr', tr(trlabels==2,1), tr(trlabels==2,2), '.b', tr(tolerancia_margen!=0,1),
tr(tolerancia_margen!=0,2), "xk",
res.SVs(tolerancia_margen_sv==0,1),res.SVs(tolerancia_margen_sv==0,2), "+k");
```

Estos son algunos de los resultados obtenidos para representar gráficamente los vectores de entrenamiento, marcando los que son vectores soporte, y la recta separadora correspondiente.

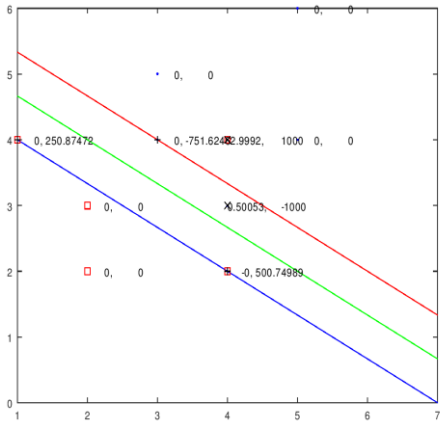


C = 1000 no separable

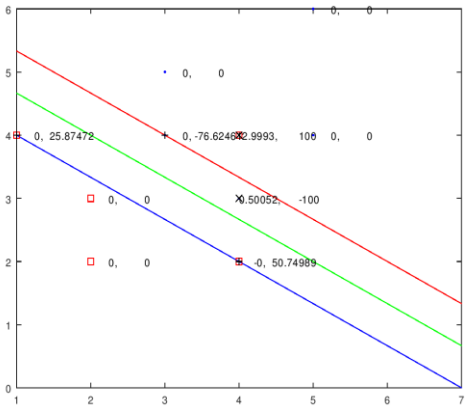


C=10 conjunto separable

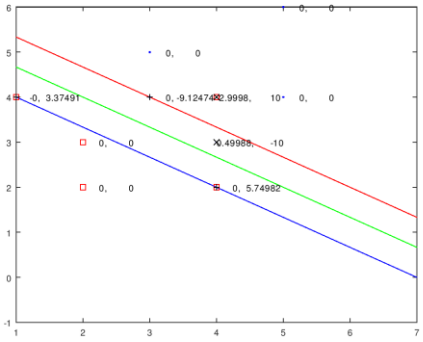
Y estos son varias pruebas para el conjunto de datos no separables que van del rango [1 – 1000] de la variable C



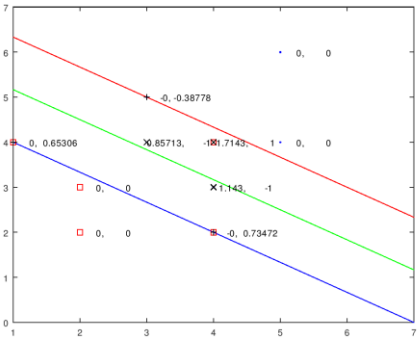
C= 1000



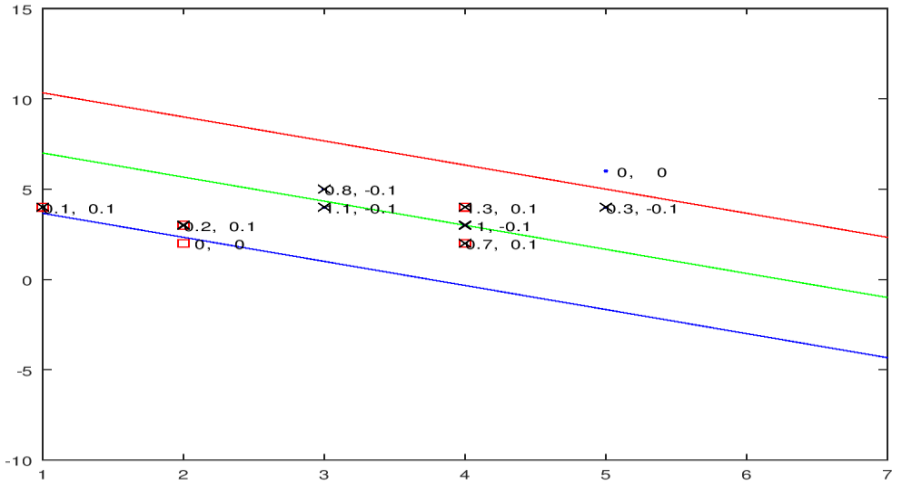
C=100



C = 10



C=1



C= 0.1

En estas graficas anteriores están representados la frontera de decisión, sus márgenes y los datos de la muestra indicando de que clase son si son vectores soporte o si están mal clasificadas y también he añadido la tolerancia del margen y los multiplicadores de lagrace asociados. Como se puede observar en los resultados al incrementar la C los márgenes disminuyen considerablemente.

Ejercicios con un corpus de dos clases

Para Realizar los resultados he utilizado la función de entrenamiento y de predicción de la siguiente forma:

```
model = svmtrain(trlabels, tr, ["-q -t ", num2str(kernel), " -c ", num2str(C), " -d
", num2str(pot)]);

model = svmtrain(trlabels, tr, ["-q -t ", num2str(kernel), " -c ", num2str(C)]);

[prediction, accuracy, decision_values] = svmpredict(tslabels, ts, model, "-q");

p = accuracy(1) / 100;

confidence = 1.96 * sqrt(p * (1-p) / N);
```

Como se puede ver hay 2 model uno para el caso especial de polinómico que necesita el argumento -d que yo he representado en la variable pot y otro para los casos lineales y radial, asi mismo el calculo de la precisión y el intervalo de confianza al 95% siguiendo las formulas es un calculo trivial.

Entrenamiento con kernel lineal

C	precisión	Intervalo de confianza
0.01	1	0
0.1	0.999276	0.001419
1	0.997104	0.002834
10	0.997104	0.002834
100	0.997104	0.002834
1000	0.997104	0.002834

Entrenamiento polinómico de P=2

C	precisión	Intervalo de confianza
0.01	0.982621	0.006892
0.1	0.992035	0.004688
1	0.995655	0.003469
10	0.997828	0.002456
100	0.997828	0.002456
1000	0.997828	0.002456

Entrenamiento polinómico de P=3

C	precisión	Intervalo de confianza
0.01	0.987690	0.005816
0.1	0.990587	0.005093
1	0.992035	0.004688
10	0.996379	0.003168
100	0.997828	0.002456
1000	0.997828	0.002456

Entrenamiento radial

C	precisión	Intervalo de confianza
0.01	0.606083	0.025771
0.1	0.737147	0.023216
1	0.867487	0.017882
10	0.908038	0.015241
100	0.907314	0.015295
1000	0.907314	0.015295

Como se pueden apreciar en los resultados de las tablas anteriores se puede comprobar como el kernel lineal puede alcanzar la mayor precisión de todas, con C=0.1 la precisión es de 1, pero su intervalo de confianza no es muy bueno y contra mas alta la C mas perdida de precisión incurre, eso si cuando C = 1 la precisión y la confianza se estabilizan y por mas incrementos que se aplique ha C estos no cambian sus valores.

los entrenamientos polinómicos demuestran que incurren en una precisión menor que el entrenamiento lineal pero en cambio tienen mejor intervalo de confianza y la precisión aumenta conforme aumenta C hasta llegar a un punto de saturación que en polinómico de orden 2 es con C = 10 y con orden 3 es de C= 100, asi mismo se puede comprobar que los resultados del polinómico de orden 3 son ligeramente peores ya que en la saturación de C la precisión y el intervalo de confianza son iguales en ambos entrenamientos pero el entrenamiento polinómico de orden 2 se satura con un C menor al del polinómico de orden 3.

Por último tenemos el entrenamiento Radial en el cual podemos observar que la precisión es mucho menor que en los otros casos, C se satura en el valor C=100 con una precisión del 90% esto es claramente inferior al 99% aproximado que pueden obtener el resto de entrenamientos, la única ventaja es que su intervalo de confianza al 95% es bastante mayor que en el resto de casos.

En conclusión, parece que es mejor utilizar el modelo de entrenamiento polinomial de orden 2 es el que mejor se ajusta a este problema salvo que usemos C menores que 1 en cuyo caso seria mas recomendable utilizar el entrenamiento lineal ya que puede conseguir una precisión perfecta de 1 con C muy bajos.

Ejercicios con un corpus multi-clase

En este ejercicio utilizo el mismo código que en el ejercicio 4 con la salvedad de cambiar los load para cargar los nuevos archivos.

Entrenamiento Lineal

C	precisión	Intervalo de confianza
0.01	0.931739	0.011034
0.1	0.928251	0.011291
1	0.926258	0.011434
10	0.925760	0.011470
100	0.925760	0.011470
1000	0.925760	0.011470

Entrenamiento Polinómico orden 2

C	precisión	Intervalo de confianza
0.01	0.725461	0.019525
0.1	0.900349	0.013105
1	0.935227	0.010768
10	0.945690	0.009915
100	0.943697	0.010085
1000	0.944694	0.010000

Entrenamiento Polinómico orden 3

C	precisión	Intervalo de confianza
0.01	0.556054	0.021737

0.1	0.854011	0.015448
1	0.928749	0.011254
10	0.938216	0.010533
100	0.941206	0.010292
1000	0.940708	0.010333

Entrenamiento Radial

C	precisión	Intervalo de confianza
0.01	0.746886	0.019023
0.1	0.914798	0.012214
1	0.942202	0.010210
10	0.950174	0.009519
100	0.949178	0.009609
1000	0.949178	0.009609

En este caso el método de entrenamiento que obtiene mayor precisión es el entrenamiento radial aunque el entrenamiento polinómico de orden 2 se acerca muy notablemente a la precisión del radial sigue siendo inferior asi como el polinómico de orden 3 es algo inferior al de orden 2 se puede afirmar que el entrenamiento radial consigue casi un 95% de precisión mientras que los polinómicos de orden 2 y 3 consiguen el 94.4% y 94% respectivamente, en cambio en esta ocasión el entrenamiento lineal es bastante mas malo que los anteriores ya que con una C alta la precisión se reduce a un 92.5% mientras que con una C del 0.01 obtiene su máximo con un 93% siendo esta muy superior a lo que pueden conseguir los otros entrenamientos con C<1 con la excepción del entrenamiento radial que c = 1 ya consigue una precisión superior a todas las demás y con C = 0.1 obtiene una precisión ligeramente inferior a la de la lineal.

En conclusión en este ejercicio parece que el mejor modelo de entrenamiento es el radial ya que consigue unas precisiones sensiblemente mas grandes que los otros modelos exceptuando si vamos ha hacer un entrenamiento con una C menor que 1 en cuyo caso seria mejor utilizar un modelo lineal.