Máquinas de vectores soporte

Marcos Esteve Casademunt

Noviembre 2018

En el problema de la clasificación en C clases con SVM, comparar las técnicas basadas en votación y en DAGs (directed acyclic graphs) utilizando SVMs de dos clases con el subconjunto de los datos utilizados en clase de prácticas para las clases 0 a 3.

1. Extraer los datos de las clases 0 a 3

```
load("data/usps/tr.dat")
load("data/usps/trlabels.dat")
datos = tr(trlabels < 4,:);
trlabelss = trlabels(trlabels<4, :);
```

2. Separar un conjunto de entrenamiento y otro de test

```
[N,L] = size(datos);
rand("seed",23);
perm = randperm(N); #permutacion de filas
data=tr(perm,:); #aplicamos la permutacion a los datos
#aplicamos la permutacion a las etiquetas
datalabels = trlabelss(perm,:);
NTr=round(.7*N); #extraemos 70% para train
traindata = data(1:NTr,:);
trainlabels = datalabels(1:NTr,:);
tedata=data(NTr+1:N,:); # 30% para test
telabels = datalabels(NTr+1:N,:);
```

Para ello, realizaremos una permutación aleatoria de las filas y escogeremos el 70 % para Train y el 30 % para Test

3. implementar un clasificador de 4 clases mediante el método por votación utilizando clasificadores de dos clases (libsvm)

```
for i = 0:3
        for j = i+1: 3
                #clasificadores binarios clase i vs j
                localdata = [traindata(trainlabels == i,:);
                traindata(trainlabels ==j,:)];
                locallabels = [trainlabels(trainlabels ==i,:);
                trainlabels(trainlabels ==j,:)];
                res = svmtrain(locallabels,localdata,"-q -t 0 -c 1000");
                if i == 0
                        clasificadores(j)= res;
                else
                        clasificadores(i+j+1) = res;
        endif
        endfor
endfor
```

Se entrenan y almacenan los distintos clasificadores binarios asociados a cada par de clases i j. En cuanto a la evaluación del método por votaciones:

```
for i = 1:6
        #almacenamos la prediccion para cada clasificador
        prediction = svmpredict(telabels,tedata,clasificadores(i),"-q");
        prediccion(:,i) = prediction;
endfor
[Filas,Columnas] = size(prediccion);
contador = zeros(Filas,4);
#contamos cuantas veces aparece la clase j [0..3] en cada fila
for i = [1:Filas]
        for j = [1:4]
                contador(i,j) = sum(prediccion(i,:) ==j-1);
endfor
endfor
#La clase asignada sera la mas votada...
[value,pos] = max(contador');
mis_etiquetas = pos-1;
aciertoVotacion = sum(mis_etiquetas' == telabels)/length(telabels)
intervaloVotacion = 1.96* sqrt((aciertoVotacion * (1-aciertoVotacion))
/length(telabels))
```

Como podemos observar en el código, se busca aquella clase que se haya elegido por un mayor número de clasificadores y se comprueba cuantas coinciden con las asignadas en el conjunto de test.

4. implementar un clasificador de 4 clases mediante el método DAG

El DAG asociado a este problema es:

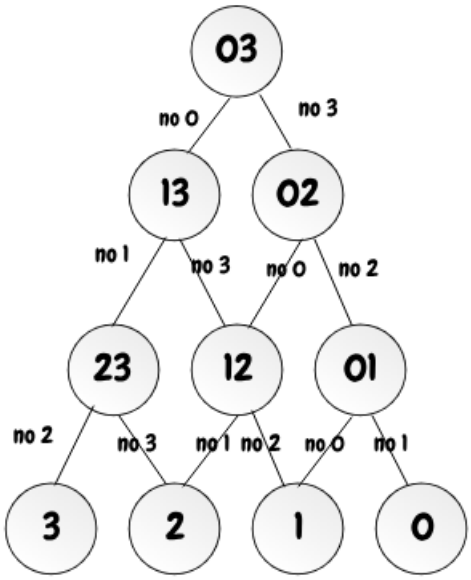

Figura 1: Grafo dirigido acíclico asociado a 4 clases

```
for d = [1:length(telabels)]
        inferior = 0;
        superior = 3;
        mi_clasificador = -1;
        while (inferior +1 != superior)
                if inferior == 0
                        mi_clasificador = superior;
                else
                        mi_clasificador = inferior+superior+1;
                endif
                prediccion =  svmpredict(telabels(d,:),tedata(d,:),
                clasificadores(mi_clasificador),"-q");
                if prediccion != superior
                        superior--;
                else
```

```
                inferior++;
            endif

        endwhile
        if inferior == 0
            mi_clasificador = superior;
        else
            mi_clasificador = inferior+superior+1;
        endif
        predicciones(d) = svmpredict(telabels(d,:),tedata(d,:),
        clasificadores(mi_clasificador),"-q");

endfor
aciertoDAG = sum(predicciones' == telabels)/length(telabels)
intervaloDAG = 1.96* sqrt((aciertoDAG * (1-aciertoDAG))/length(telabels))
```

Para cada dato se recorre el DAG para obtener la clase a a la que pertenece. Por último se evalua la precisión del modelo desarrollado

5. Comparar los mejores resultados obtenidos con el test

Kernel	% Train	Accuracy Votación	Accuracy DAGs	Accuracy original
Lineal	80 %	0.99304	0.99304	0.99304
Lineal	60 %	0.98955	0.99094	0.99024
Polinomial	80 %	0.99582	0.99582	0.99582
Polinomial	60 %	0.98258	0.98258	0.98258
Radial	80 %	1	1	1
Radial	60 %	0.99373	0.99373	0.99373

Cuadro 1: Evolución de la precisión al variar el conjunto de Train y el kernel

Kernel	% Train	Confidence Votación	Confidence DAGs	Confidence original
Lineal	80 %	0.0060827	0.0060827	0.0060827
Lineal	60 %	0.0052622	0.0049023	0.0050856
Polinomial	80 %	0.0047183	0.0047183	0.0047183
Polinomial	60 %	0.0067695	0.0067695	0.0067695
Radial	80 %	0	0	0
Radial	60 %	0.0040847	0.0040847	0.0040847

Cuadro 2: Evolución del intervalo de confianza al variar el conjunto de Train y el kernel

No se observan diferencias significativas entre los distintos métodos, pero si en el coste temporal de ambos algoritmos. Mientras que el método basado en votaciones tiene un coste temporal de $O(C^2)$ el método basado en DAG consigue reducir el coste a O(C). Esto es debido a que no es necesario explorar todos los clasificadores en el método de DAGs. Por último, se han comparado los resultados obtenidos de ambos métodos con el programa original proporcionado en la asignatura. Los resultados son prácticamente idénticos. El mejor resultado observado se consigue utilizando un kernel radial con un 80 % para Train y un 20 % para Test.

6. Anexo: Código completo

```
graphics_toolkit gnuplot;
load("data/usps/tr.dat") #tr
load("data/usps/trlabels.dat")#trlabels
#Extraemos los datos de las clases 0 a 3
datos = tr(trlabels < 4,:);
trlabelss = trlabels(trlabels <4, :);
#Barajamos las filas y cogemos un conjunto para train y otro para test
[N,L] = size(datos);
rand("seed",23);
perm = randperm(N); #permutación de filas
data=tr(perm,:); #aplicamos la permutación a los datos
#aplicamos la permutación a las etiquetas
datalabels = trlabelss(perm,:);
NTr=round(.4*N); #extraemos 70% para train
traindata = data(1:NTr,:);
trainlabels = datalabels(1:NTr,:);
tedata=data(NTr+1:N,:); # 30% para test
telabels = datalabels(NTr+1:N,:);
for i = 0:3
```

```
        for j = i+1: 3
                #clasificadores binarios clase i vs j
                localdata = [traindata(trainlabels == i,:);
                 traindata(trainlabels ==j,:)];
                locallabels = [trainlabels(trainlabels ==i,:);
                 trainlabels(trainlabels ==j,:)];
                res = svmtrain(locallabels,localdata,"-q -t 0 -c 1000");
                if i == 0
                        clasificadores(j)= res;
                else
                        clasificadores(i+j+1) = res;
                endif
        endfor
endfor
disp("Entrenado con exito modelo votacion")
for i = 1:6
        #almacenamos la prediccion para cada clasificador
        [prediction, accuracy, decision_values] = svmpredict(telabels,
                tedata,clasificadores(i),"-q");
        prediccion(:,i) = prediction;
endfor
[Filas,Columnas] = size(prediccion);
contador = zeros(Filas,4);
#contamos cuantas veces aparece la clase j [0..3] en cada fila
for i = [1:Filas]
        for j = [1:4]
                contador(i,j) = sum(prediccion(i,:) ==j-1);
endfor
endfor
#La clase asignada será la más votada...
[value,pos] = max(contador');
mis_etiquetas = pos-1;
aciertoVotacion = sum(mis_etiquetas' == telabels)/length(telabels)
intervaloVotacion = 1.96* sqrt((aciertoVotacion * (1-aciertoVotacion))/
length(telabels))
disp("DAGS")

for d = [1:length(telabels)]
        inferior = 0;
        superior = 3;
        mi_clasificador = -1;
        while (inferior +1 != superior)
                if inferior == 0
                        mi_clasificador = superior;
```

```
                else
                        mi_clasificador = inferior+superior+1;
                endif
                prediccion =  svmpredict(telabels(d,:),tedata(d,:),
                clasificadores(mi_clasificador),"-q");
                if prediccion != superior
                        superior--;
                else
                        inferior++;
                endif

        endwhile
        if inferior == 0
                mi_clasificador = superior;
        else
                mi_clasificador = inferior+superior+1;
        endif
        predicciones(d) = svmpredict(telabels(d,:),tedata(d,:),
        clasificadores(mi_clasificador),"-q");

endfor
aciertoDAG = sum(predicciones' == telabels)/length(telabels)
intervaloDAG = 1.96* sqrt((aciertoDAG * (1-aciertoDAG))/
length(telabels))
```