Aprendizaje Automático

Máquinas de Vectores Soporte

curso 2016-2017

Enrique Vidal, Vicente Bosch, Francisco Casacuberta
Departamento de Sisatemas Informáticos y Computación
Universidad Politécnica de Valencia

1. Introducción

El objetivo de esta práctica es experimentar con las máquinas de vectores soporte *Support Vector Machine* (SVM) dentro de `octave`. Para ello se utiliza la librería `LibSVM` que se puede descargar en:

`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Para poder emplear dicha librería desde `octave` se deben tener compilados los ejecutables: `svm-train` y `svm-predict`. También se necesitan tener los ficheros de `octave` `svmtrain.mex` y `svmpredict.mex`. Todo ello en el directorio de trabajo.

Todos estos requerimientos se han tenido en cuenta en la elaboración del toolkit `apr_toolkit.tgz`, que está en el mismo directorio de esta práctica. Para la realización de esta práctica se aconseja que se copie a un directorio de trabajo del usuario y se descomprima para empezar a trabajar en dicho directorio.

2. Ejemplo de entrenamiento y clasificación con SVM

2.1. Corpus de ejemplo

Para introducir la funcionalidad básica de la librería dentro de `octave` se propone emplear el corpus de `hart` que se encuentra dentro del directorio `data`. Es un corpus de dos clases donde los puntos se representan en dos dimensiones. Los pasos a seguir podrían ser:

1. Ejecutamos `octave` en el directorio `apr_toolkit`.
2. Una vez en `octave`, cargamos los ficheros con los datos y las etiquetas de clase tanto del corpus de entrenamiento como del de prueba:

```
octave:1> load data/hart/tr.dat
octave:2> load data/hart/trlabels.dat
octave:3> load data/hart/ts.dat
octave:4> load data/hart/tslabels.dat
```

Una vez cargados los datos se pueden, por ejemplo, visualizar las muestras de entrenamiento con:

```
octave:5> plot (tr(:,1),tr(:,2),"x")
```

El problema es que no se distinguen las muestras de cada clase. Mejor dibujar las de la clase 1 y la clase 2 de manera separada:

```
octave:6> plot(tr(trlabels==1,1),tr(trlabels==1,2),"x",tr(trlabels==2,1),
tr(trlabels==2,2),"s")
```

2.2. Entrenamiento

Realizamos un entrenamiento de SVM con kernel tipo RBF.

```
octave:7> res=svmtrain(trlabels,tr,'-t 2 -c 1');
.*.*
optimization finished, #iter = 2298
nu = 0.174579
obj = -108.403744, rho = -0.096434
nSV = 398, nBSV = 91
Total nSV = 398
```

Para ver todas las opciones de `svmtrain` escribir en `octave` `svmtrain` y darle al `enter`. Podemos ver los diferentes parámetros que le podemos usar para el aprendizaje de la SVM: kernel, parámetro $\mathcal{C}$, grado del kernel polinomial, etc.

Observamos como el resultado del proceso de entrenamiento lo guardamos en la variable `res` que es un tipo estructurado donde se almacenará entre otros: los parámetros del modelo, los índices de los vectores que han resultado de soporte, los multiplicadores de Lagrange α_i asociados a cada SV, etc.

Ver para este ejemplo el contenido de `res`:

```
octave:8> res
```

La forma de acceder a cada uno de los campos de `res` es con el operador ".". Por ejemplo para ver los índices de los SV's podemos hacer:

```
octave:9> res.sv_indices
```

Ejercicio propuesto: Dibuja los SV's superponiéndolos a las muestras de entrenamiento.

2.3. Clasificación y estimación de la tasa de acierto

A partir del modelo entrenado en la etapa anterior (variable `res`), clasificamos un conjunto de prueba mediante la orden `svmpredict`:

```
octave:10> svmpredict(tslabels,ts, res,' ');
Accuracy = 98.1% (981/1000) (classification)
```

Como en el caso de `svmtrain`, para ver todas las opciones de `svmpredict` escribir en `octave svmpredict` y darle al `enter`. Tanto en `svmtrain` como en `svmpredict` podremos poner entre ' ' las opciones que admite la librería `svm`.

2.4. Un pequeño ejercicio de aprendizaje completo (a entregar)

En el subdirectorio `datos/mini` se encuentran dos conjuntos de datos de entrenamiento en dos dimensiones: `(trSep.dat, trSeplabels.dat)` y `(tr.dat, trlabels.dat)`. El primero es linealmente separable y el segundo no. Para cada uno de estos conjuntos:

1. Obtener una SVM sin kernel (kernel tipo lineal). Para simular la optimización estándard del caso separable basta usar un valor grande de $\mathcal{C}$;
2. Determinar a) los multiplicadores de Lagrange, α, asociados a cada dato de entrenamiento, b) los vectores soporte, c) el vector de pesos y umbral de la función discriminante lineal, y c) el margen correspondiente;
3. Calcular los parámetros de la recta separadora;
4. Representar gráficamente los vectores de entrenamiento, marcando los que son vectores soporte, y la recta separadora correspondiente.

Además, para el conjunto no-separable:

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

4.1 En la representación gráfica, marcar los vectores soporte "erróneos".

3. Ejercicios con corpus en dos dimensiones

En esta sección proponemos la realización de diferentes pruebas con dos corpus bidimensionales. La elección de corpus en dos dimensiones se debe a que facilitan la tarea de visualizar los resultados. El primer corpus es linealmente separable (`2dseparable`) y el segundo corpus no es linealmente separables (`2dnoseparable`). Ambos corpus se encuentran dentro del directorio `data`.

Se pide: Diseñad una función que haga un barrido del parámetro $\mathcal{C} \in$ `[0.01,0.1,1,10,100]` con un kernel radial. La función debe cargar los ficheros `tr.dat` y `trlabels.dat` de los directorios adecuados.

A entregar: La función completa y la gráfica con los SV's, tanto en el caso separable como en el no separable, para el parámetro $\mathcal{C}$ con el que se obtengan el menor número de SV's.

4. Ejercicios con corpus dos clases

En esta sección proponemos la realización de un experimento de clasificación completo: entrenamiento y evaluación. Para ello se empleará un corpus, de dos clases, de detección de *Spam* compuesto por 3220 muestras de aprendizaje y 1381 de test. Las muestras se representan mediante 58 componentes. Los ficheros de entrenamiento y test se encuentran en el directorio `spam` de `data`.

Se pide: diseñad una función que haga un barrido de los diferentes parámetros del modelo, en concreto el tipo de kernel y el valor de $\mathcal{C}$.

A entregar: La función implementada así como la siguiente tabla de resultados de clasificación del conjunto de test para diferentes parámetros del entrenamiento:

Kernel	C=0.01	C=0.1	C=1	C=10	C=100
Lineal					
Polinomial $d = 2$					
Polinomial $d = 3$					
Radial					

5. Ejercicios con corpus multi-clase

En esta sección proponemos la realización de un experimento de clasificación completo para un corpus multiclase. Para ello se empleará un corpus de 10 clases de reconocimiento de dígitos manuscritos (`usps`) compuesto por 7291 muestras de aprendizaje y 2007 de test. Las muestras se representan mediante 256 componentes que se corresponden con los valores de gris de imágenes de 16×16 píxeles. Los ficheros de entrenamiento y test se encuentran en el directorio `usps` de `data`.

Se pide: diseñad una función que haga un barrido de los diferentes parámetros del modelo, en concreto el tipo de kernel y el valor de $\mathcal{C}$.

A entregar: La función implementada así como la siguiente tabla de resultados de clasificación del conjunto de test para diferentes parámetros del entrenamiento:

Kernel	C=0.01	C=0.1	C=1	C=10	C=100
Lineal					
Polinomial $d = 2$					
Polinomial $d = 3$					
Radial					

Ejercicio opcional: Visualizar algunas de las imágenes de aprendizaje. Como ayuda podéis ver los comandos `imshow` y `reshape` de `octave`.

Ejercicio opcional: Visualizar algunos de los SV's.