Práctica 1: Clasificación

Tratamiento de Datos
Curso 2013–2014

6 de diciembre de 2013

Introducción

En esta práctica analizaremos los métodos de regresion logística y las máquinas de vectores soporte utilizando datos generados artificialmente. A continuación, analizaremos el comportamiento de estos clasificadores utilizando una base de datos reales.

1. Análisis de un problema sintético bidimensional

Cargue los datos contenidos en el fichero `'datosP1.mat'`. El fichero contiene dos variables, `x` e `y`. La primera es una matriz de observaciones. Cada fila de la matriz es una observación bidimensional. La segunda contiene la categoría a la que pertenece cada observación (0 ó 1).

1.1. Regresión logística: clasificador lineal

En primer lugar, analizaremos el comportamiento de un modelo de regresión logística.

1. Visualice las observaciones sobre el plano, utilizando un marcador diferente para los datos de cada categoría.
2. Utilizando la función `glmfit`, ajuste un modelo de regresión logística a los datos.

```
w =
    1.5003
   -2.4573
    0.1496
```

3. Utilizando la función `contourf`, visualice el mapa de probabilidades a posteriori estimadas para la categoría 1.
4. Determine las tasas de error de entrenamiento y validación.

```
Error rates:
eTrain =
    0.2975
eVal =
    0.3525
```

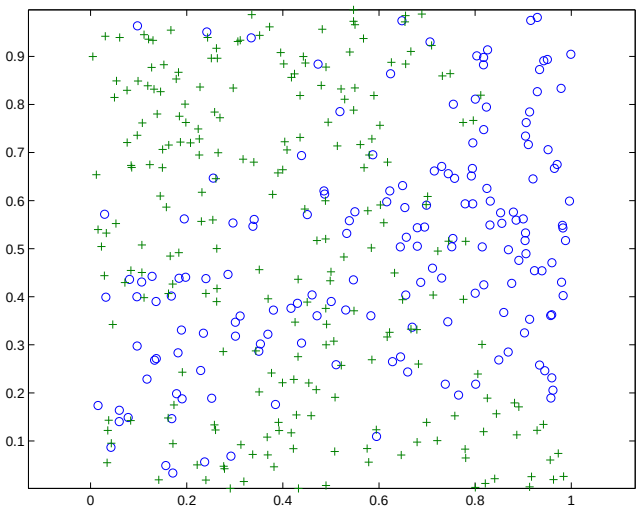

Figura 1: Datos de entrenamiento

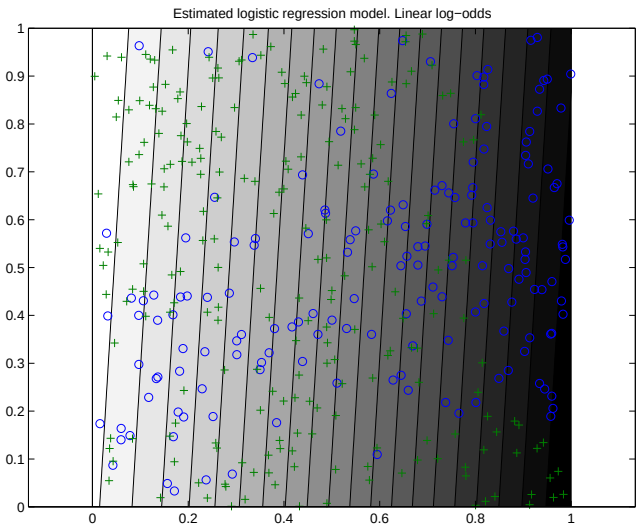

Figura 2: Probabilidad a posteriori

5. Determine las verosimilitud de los datos de entrenamiento y test para el modelo obtenido.

```
Likelihoods for training and test data:
lTrain =
  250.0087
lTest =
  239.3246
```

1.2. Regresión logística: clasificador no lineal

A continuación, analizaremos el comportamiento de un clasificador no lineal basado en un modelo de regresión logística con términos polinómicos.

1. Utilizando la función `glmfit`, ajuste un modelo de regresión logística con términos polinómicos de grado 1, 2 y 3 a los datos.

```
w2 =
   -0.5516
   11.4479
   -5.7585
   -8.0336
  -14.2474
   13.5061


w2 =
    1.0855
   -1.4407
  -16.5035
   42.6038
  -42.7352
   55.5733
  -34.9939
   -7.4124
   34.3358
  -37.8443
```

2. Utilizando la función `contourf`, visualice el mapa de probabilidades a posteriori estimadas para la categoría 1.

3. Calcule las tasas de error de entrenamiento y validación.

```
Training and validation error rates for d=1:
eTrain =
    0.2975
eVal =
    0.3525

Training, validation and test error rates for d=2:
eTrain =
```

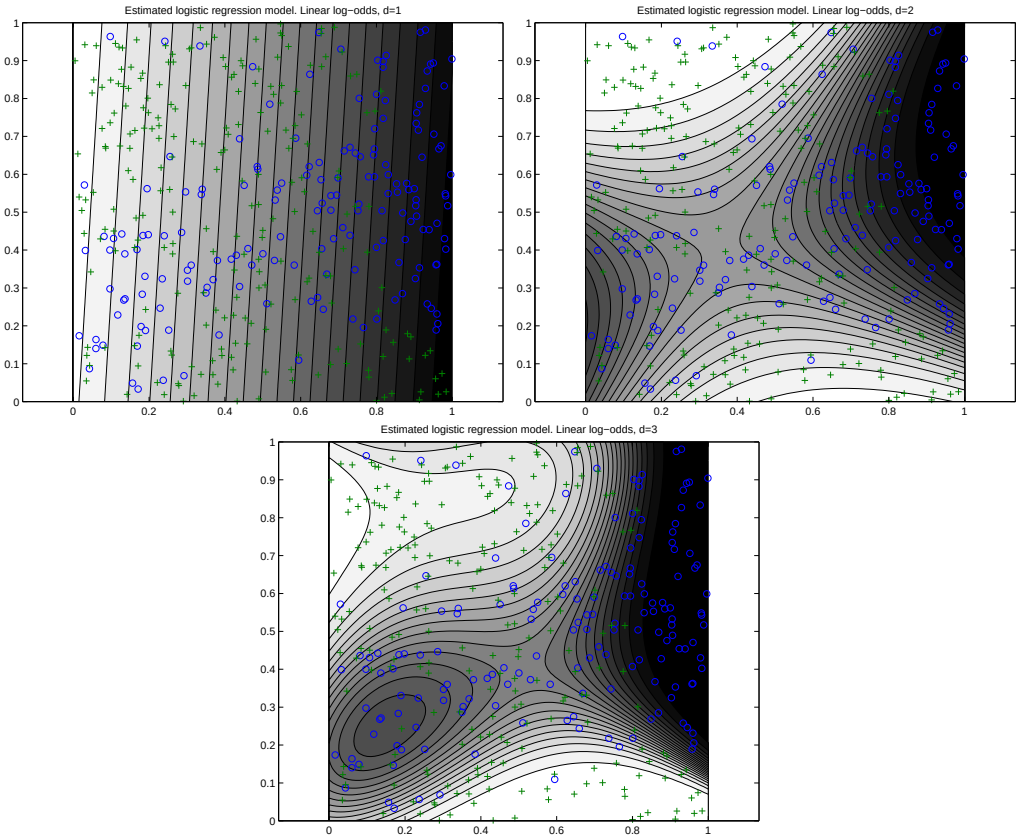

Figura 3: Probabilidad a posteriori con modelos de regresión logística con términos polinómicos de grado 1 (primera fila izda), 2 (primera fila dcha) y 3 (segunda fila) a los datos.

```
    0.2325
eVal =
    0.2625


Training and validation error rates  for d=3:
eTrain =
    0.2075
eTest =
    0.2275
```

4. Calcule las verosimilitud de los datos de entrenamiento y test para el modelo obtenido.

```
Likelihoods for training and test data for d=1:
lTrain =
  250.0087
lTest =
  239.3246

Likelihoods for training and test data for d=2:
lTrain =
  189.2652
```

```
lTest =
  171.9362

Likelihoods for training and test data  for d=3:
lTrain =
  173.5047
lTest =
  171.3453
```

1.3. Máquina de vectores soporte

Evaluaremos a continuación las prestaciones de un clasificador basado en máquinas de vectores soporte (SVM, *Support Vector Machine*).

Utilizaremos, para ello, una SVM con núcleos gaussianos de desviación típica 0.5.

1. Utilizando la función `svmtrain`, entrene una SVM con los datos de entrenamiento, visualizando la frontera de decisión, tomando $C = 0.1$. (Nótese que sólo se puede visualizar la frontera de decisión de forma automática usando la función svmtrain del statistic toolbox, no con el uso de la función de svmtrain de las librerías provistas).

   El comando para usar la función svmtrain del statistic toolbox es la siguiente:

   ```
   figure
   svmStruct0 = svmtrain(xTrain,yTrain,     'kernel_function','rbf',...
                         'rbf_sigma',sigma, 'showplot','true',...
                         'boxconstraint',C, 'method','LS');
   dVal   = svmclassify(svmStruct0,xVal,  'showplot',false);
   dTrain = svmclassify(svmStruct0,xTrain,'showplot',false);
   ```

   y se obtienen los siguientes errores

   ```
   eTrain =
       0.2200
   eVal =
       0.2950
   ```

   El comando para usar la función svmtrain de las librerías provistas es la siguiente:

   ```
   svmStruct0 = svmtrain(yTrain+1,xTrain, ['-t 2 -c ' num2str(C) ' -g ' num2str(1/sigma)]);
   [dVal, accuracyVal] = svmpredict(yVal+1, xVal, svmStruct0);
   ```

   nótese que como la tercera variable de entrada es una cadena de texto, hay que dejar espacios entre las opciones elegidas.

2. Haciendo un barrido sobre el parámetro C, visualice las tasas de error de entrenamiento y validación, en función de C, y elija un valor de acuerdo con el error de validación. (Resultados correspondientes a un barrido de C

   ```
   C  = 10.^(-4:0.1:2);
   ```

)

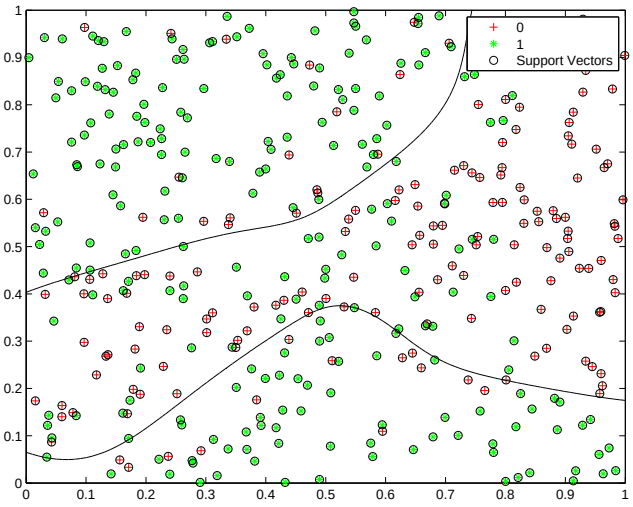

Figura 4: Datos de entrenamiento (cruces rojas y verdes), frontera de decisión (línea negra) y vectores soporte (círculos negros)

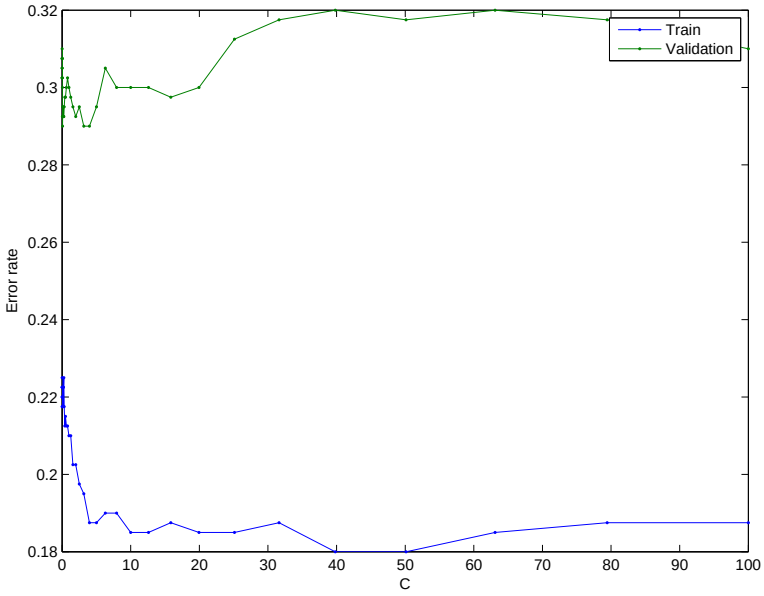

Figura 5: Tasas de error de entrenamiento (azul) y validación (verde).

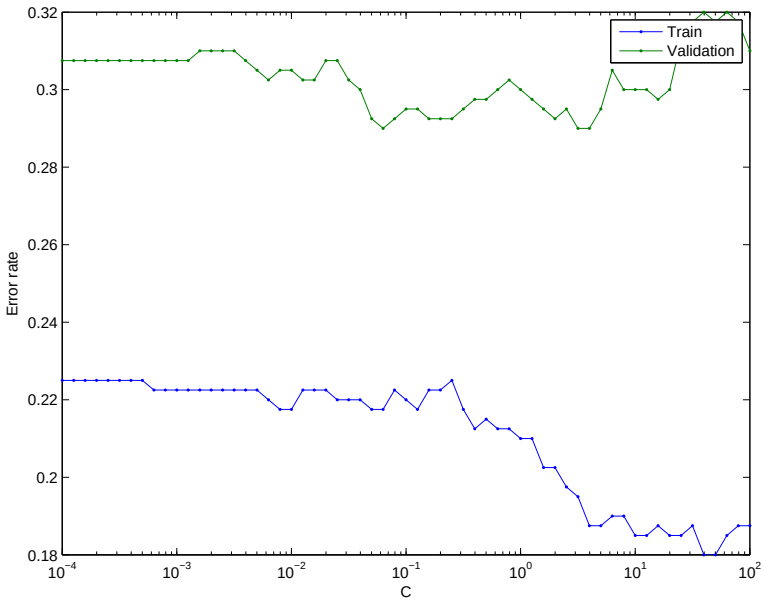

Figura 6: Tasas de error de entrenamiento y validación con C en escala logarítmica para mejor visualización.

1.4. Evaluación final

Seleccione, de acuerdo con las tasas de error de validación, el clasificador que mejores prestaciones ha obtenido, de entre todos los analizados en las secciones anteriores, y determine su tasa de error con los datos de test.

```
[eMin,iMin] = min(eVal);
 C(iMin)
ans =
    0.0631
```

Usando el clasificado que mejores prestaciones ha obtenido en error de validación se error de test es

```
eTest =
    0.2375
```

2. Clasificación con datos reales

Aplicaremos en esta sección los clasificadores anteriores a una base de datos reales multidimensionales. La visualización de los datos en el espacio de entrada ya no es posible, pero podemos evaluar las prestaciones utilizando las tasas de error.

Trabajaremos con el `cancer_dataset`.

2.1. Preparación de datos

Antes de entrenar el clasificador, conviene normalizarlos, para que la escala de todas las variables sea similar.

Las siguientes líneas de código realizan una transformación lineal que hace que los datos de entrenamiento tengan media nula y varianza unidad. Exactamente la misma transformación ha de aplicarse a los datos de validación y de test:

```
mx = mean(xTrain); stdx = std(xTrain);
xTrain = (xTrain - ones(nTrain,1)*mx)./(ones(nTrain, 1)*stdx);
xVal = (xVal - ones(nVal,1)*mx)./(ones(nVal,1)*stdx);
xTest = (xTest - ones(nTest,1)*mx)./(ones(nTest,1)*stdx);
```

Realizaremos todo el resto de la práctica con los datos normalizados.

2.2. Clasificación con máquinas de vectores soporte

En este apartado entrenaremos una máquina de vectores soporte con núcleos gaussianos. Utilizaremos el conjunto de validación para determinar los valores de los hiperparámetros *sigma* y C.

1. Entrene la SVM para diferentes valores de σ entre -0.02 y 8 y valores de C entre 0.001 y 1000. Determine las tasas de error de entrenamiento y validación en cada caso, y represente gráficamente (por ejemplo, mediante la función `contourf`), dichas tasas de error.

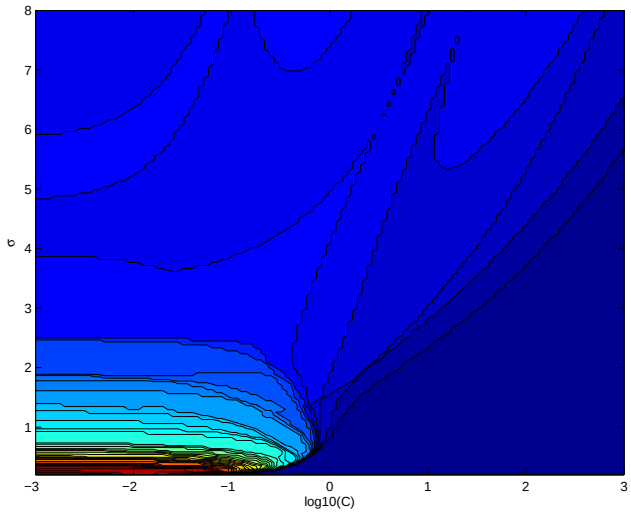

Figura 7: Resultados de error con datos de entrenamiento para diferentes valores de σ entre -0.02 y 8 y valores de C entre 0.001 y 1000.

2. Determine los valores de anchura del núcleo, σ, y del parámetro C que minimizan la tasa de error de validación.

```
sigma(jMin)
ans =
    1.4500
K>> C(kMin)
ans =
    0.4467
```

3. Evalúe las prestaciones del clasificador utilizando los datos de test, para los valores de los hiperparámetros seleccionados en el apartado anterior.

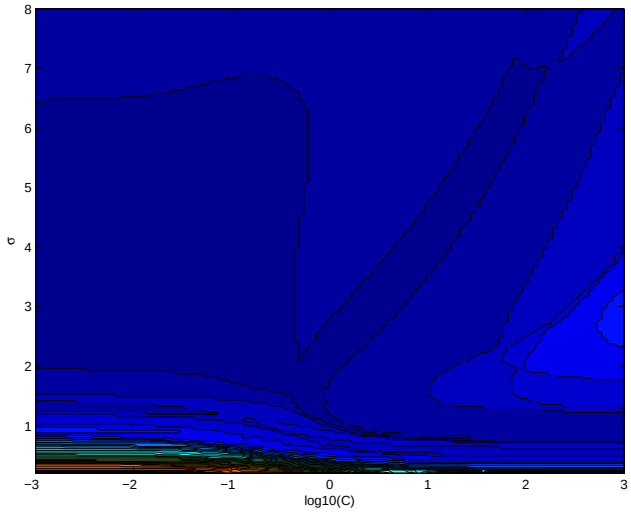

Figura 8: Resultados de error con datos de validación para diferentes valores de σ entre -0.02 y 8 y valores de C entre 0.001 y 1000.

Usando los los valores de C y sigma que minimizan el error de validación, se clasifica nuevamente obteniendo un error de test de

```
eTest =
    0.0258
```