Support Vector Machines - IV

Prof. Dan A. Simovici

UMB

1. Building an SVM Classifier for the Iris data set

2. Data Preparation for IRIS

3. Other available kernels in kernlab

Letter Image Recognition Data

Data was created by D. J. Slate and used in P. W. Frey and D. J. Slate (Machine Learning Vol 6 #2 March 91) "Letter Recognition Using Holland-style Adaptive Classifiers".

The Structure of Data

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet.

- The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli.
- Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15.
- Training is made for the first 16000 items and then use the resulting model to predict the letter category for the remaining 4000.

Attribute Information

1.	lettr	capital letter	(26 values from A to Z)
2.	x-box	horizontal position of box	(integer)
3.	y-box	vertical position of box	(integer)
4.	width	width of box	(integer)
5.	high	height of box	(integer)
6.	onpix	total # on pixels	(integer)
7.	x-bar	mean x of on pixels in box	(integer)
8.	y-bar	mean y of on pixels in box	(integer)
9.	x2bar	mean x variance	(integer)
10.	y2bar	mean y variance	(integer)
11.	xybar	mean x y correlation	(integer)
12.	x2ybr	mean of x * x * y	(integer)
13.	xy2br	mean of x * y * y	(integer)
14.	x-ege	mean edge count left to right	(integer)
15.	xegvy	correlation of x-ege with y	(integer)
16.	y-ege	mean edge count bottom to top	(integer)
17.	yegvx	correlation of y-ege with x	(integer)

Class Distribution

789 A	766 B	736 C	805 D	768 E	775 F	773 G
734 H	755 I	747 J	739 K	761 L	792 M	783 N
753 O	803 P	783 Q	758 R	748 S	796 T	813 U
764 V	752 W	787 X	786 Y	734 Z		

Data Structure of the object letters

```
> letters <- read.csv("letter-recognition.csv",header=TRUE,sep=",")
> str(letters)
’data.frame’: 20000 obs. of 17 variables:
$ lettr : Factor with 26 levels "A","B","C","D",..: 20 9 4 14 7 19 2 1 10 13 ...
$ x.box : int 2 5 4 7 2 4 4 1 2 11 ...
$ y.box : int 8 12 11 11 1 11 2 1 2 15 ...
$ width : int 3 3 6 6 3 5 5 3 4 13 ...
$ high : int 5 7 8 6 1 8 4 2 4 9 ...
$ onpix : int 1 2 6 3 1 3 4 1 2 7 ...
$ x.bar : int 8 10 10 5 8 8 8 8 10 13 ...
$ y.bar : int 13 5 6 9 6 8 7 2 6 2 ...
$ x2bar : int 0 5 2 4 6 6 6 2 2 6 ...
$ y2bar : int 6 4 6 6 6 9 6 2 6 2 ...
$ xybar : int 6 13 10 4 6 5 7 8 12 12 ...
$ x2ybr : int 10 3 3 4 5 6 6 2 4 1 ...
$ xy2br : int 8 9 7 10 9 6 6 8 8 9 ...
$ xletters.ede: int 0 2 3 6 1 0 2 1 1 8 ...
$ xegvy : int 8 8 7 10 7 8 8 6 6 1 ...
$ y.ege : int 0 4 3 2 5 9 7 2 1 1 ...
$ yegvx : int 8 10 9 8 10 7 10 7 7 8 ...
```

R Packages specialized in SVMs:

- kernlab
- svmlight
- libsvm
- e1071

We shall use kernlab.

```
>
> local(pkg <- select.list(sort(.packages(all.available = TRUE)),graphics=TRUE)
+ if(nchar(pkg)) library(pkg, character.only=TRUE))
Warning message:
package "kernlab" was built under R version 3.0.2
```

Construction of Training Set and Test Set

```
>
> letters_train ← letters[1:16000, ]
> letters_test ← letters[16001:20000, ]
>
```

```
> letter_classifier → ksvm(lettr ∼ .,data= letters_train,kernel = "vanilladot")
Setting default kernel parameters
> letter_classifier
Support Vector Machine object of class "ksvm"
SV type: C-svc (classification)
parameter : cost C = 1
Linear (vanilla) kernel function.
Number of Support Vectors : 7037
Objective Function Value : -14.1746 -20.0072 -23.5628 -6.2009 -7.5524 -32.7694 -49.9786 -18.1824 -62.1111 -32.7
-16.2209 -32.2837 -28.9777 -51.2195 -13.276 -35.6217 -30.8612 -16.5256 -14.6811 -32.7475 -30.3219 -7.7956 -11.8
-32.3463 -13.1262 -9.2692 -153.1654 -52.9678 -76.7744 -119.2067
...
Training error : 0.130062
```

```
> letter_prediction ← predict(letter_classifier,letters_test)
> head(letter_prediction)
[1] U N V X N H
Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
> table(letter_prediction,letters_test$lettr)
letter_prediction A B C D E F G
A 144 0 0 0 0 0 0
B 0 121 0 5 2 0 1
C 0 0 120 0 4 0 10
D 2 2 0 156 0 1 3
E 0 0 5 0 127 3 1
F 0 0 0 0 0 138 2
G 1 1 2 1 9 2 123
(in abbreviated form)
```

```
> agreement ← letter_prediction == letters_test$lettr
> table(agreement)
agreement
FALSE TRUE
643 3357
>
```

Data Set Description

Attribute Information:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
  - Iris Setosa
  - Iris Versicolour
  - Iris Virginica

Data Presentation

Data contains 150 records: 50 records for each class value: *setosa*, *versicolor*, and *virginica*.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
.
.
.
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
.
.
.
6.3,2.5,5.0,1.9,Iris-virginica
6.5,3.0,5.2,2.0,Iris-virginica
6.2,3.4,5.4,2.3,Iris-virginica
5.9,3.0,5.1,1.8,Iris-virginica
```

The IRIS data set is already grouped on class values; this requires a random rearrangement of the records in order to extract the training set and the test set.

Uniform distribution generation

The function runif generates n values of a random variable uniformly distributed in the interval $[m, M]$.

It is called using

$$> \text{runif}(n, m, M)$$

```
> runif(10,12, 20)
[1] 14.81854 13.33863 17.58722 15.75252 17.11880 13.99228 19.8
12.95395 [9] 18.50042 12.46879
```

If called with one argument n it produces n random values in the interval $[0, 1]$.

Ordering Permutation

The function order returns a permutation which rearranges its first argument into ascending or descending order, breaking ties by further arguments.

```
> iris_rand < − iris[order(runif(150)), ]
```

Classifier Generation

```
> iris_train < − iris_rand[1:120,]
> iris_test < − iris_rand[121:150,]
> iris_classifier < − ksvm(class ∼ .,
+ data = iris_train, kernel = "vanilladot")
> iris_prediction < − predict(iris_classifier,iris_test)
> table(iris_prediction,iris_test$class)
```

Kernels available in kernlab

- The linear `vanilladot` is the simplest and is given by $K(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}'\boldsymbol{v}$; this is useful when dealing with large sparse data vectors (typically text cathegorization).
- the Gaussian radial basis kernel `rbfdot` is $K(\boldsymbol{u}, \boldsymbol{v}) = e^{-\sigma\|\boldsymbol{u}-\boldsymbol{v}\|^2}$; a typical invocation is

  $$\text{rbf} \; < -\text{rbfdot}(\text{sigma} = 0.05)$$

  This is a general kernel and is used when no further prior knowledge exists about data.
- The polynomial kernel `polydot` $K(\boldsymbol{u}, \boldsymbol{v}) = (k\boldsymbol{u}'\boldsymbol{v} + c)^d$ frequently used in image classification.

- The hyperbolic tangent kernel `tanhdot` is

$$K(\boldsymbol{u},\boldsymbol{v}) = \tanh(k\boldsymbol{u}'\boldsymbol{v} + c)$$

  mainly used as an alternative to neural networks.
- The Laplace radial basis kernel `laplacedot`

$$K(\boldsymbol{u},\boldsymbol{v}) = e^{-\sigma\|\boldsymbol{u}-\boldsymbol{v}\|}$$

  is a general purpose kernel.
- the ANOVA radial basis kernel `anovadot`

$$K(\boldsymbol{u},\boldsymbol{v}) = \left(\sum_{i=1}^{n} e^{-\sigma(u_i-v_i)^2}\right)^d$$

  used in multidimensional regression problems.

Example

```
> letter < −read.csv("letter-recognition.csv",header=TRUE,sep=",")
> letters_train < − letter[1:16000,]
> letters_test < − letter[16001:20000,]
> letter_classifier < − ksvm(lettr   .,data = letters_train,kernel="rbfdot")
Using automatic sigma estimation (sigest) for RBF or Laplace kernel
> letter_classifier
Support Vector Machine object of class "ksvm"
SV type: C-svc (classification)
parameter : cost C = 1
Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.0474609039404198
Number of Support Vectors : 8680
Objective Function Value : -43.1068 -33.8779 -59.0838 -27.2155 -34.6708 -46.8762 ....
Training error : 0.051625
```

Example

```
> letter_classifier < − ksvm(lettr  .,data = letters_train,kernel="polydot")
Setting default kernel parameters
> letter_classifier
Support Vector Machine object of class "ksvm"
SV type: C-svc (classification)
parameter : cost C = 1
Polynomial kernel function.
Hyperparameters : degree = 1 scale = 1 offset = 1
Number of Support Vectors : 7035
Objective Function Value : -14.1746 -20.0072 -23.5628 -6.2009 -7.5524 -32.7694 ....
Training error : 0.130125
```

Example

```
> letter_classifier < − ksvm(lettr   .,data = letters_train,kernel= "tanhdot")
Setting default kernel parameters
> letter_classifier
Support Vector Machine object of class "ksvm"
SV type: C-svc (classification)
parameter : cost C = 1
Hyperbolic Tangent kernel function.
Hyperparameters : scale = 1 offset = 1
Number of Support Vectors : 15696
Objective Function Value : -15157.29 -1786.306 -15642.6 -5531.012 -1218.474 -14029.91 ...
Training error : 0.910875
```

Example

```
> letter_classifier < − ksvm(lettr   .,data = letters_train,kernel="laplacedot")
Using automatic sigma estimation (sigest) for RBF or laplace kernel
> letter_classifier
Support Vector Machine object of class "ksvm"
SV type: C-svc (classification)
parameter : cost C = 1
Laplace kernel function.
Hyperparameter : sigma = 0.0477332265453678
Number of Support Vectors : 11331
Objective Function Value : -101.5121 -67.578 -131.9846 -70.7183 -77.3382 -109.682 ...
Training error : 0.084875
```

Example

```
> letter_classifier < − ksvm(lettr   .,data = letters_train,kernel="anovadot")
Setting default kernel parameters
> letter_classifier
Support Vector Machine object of class "ksvm"
SV type: C-svc (classification)
parameter : cost C = 1
Anova RBF kernel function.
Hyperparameter : sigma = 1 degree = 1
Number of Support Vectors : 6636
Objective Function Value : -8.7926 -9.3741 -12.0187 -6.6614 -5.8274 -16.8295 ...
Training error : 0.032687
```