Advances Computational Econometrics. Chapter 6: Kernel-based classification and regression

Thierry Denoeux

5/9/2022

Exercise 1

We first load package `kernlab` and read the data file:

```
library(kernlab)
credit<-read.csv('/Users/Thierry/Documents/R/Data/Economics/Greene/TableF7-3.csv',
                 sep=",",header=TRUE)
```

We then split the data and declare the response variables as factors:

```
set.seed(9052022)
n<-nrow(credit)
ntrain<-10000
ntest<-n-ntrain
train<-sample(n,ntrain)
credit$CARDHLDR<-as.factor(credit$CARDHLDR)
credit$DEFAULT<-as.factor(credit$DEFAULT)
```

Prediction of `CARDHLDR`

We create a new data frame without the four variables to be excluded:

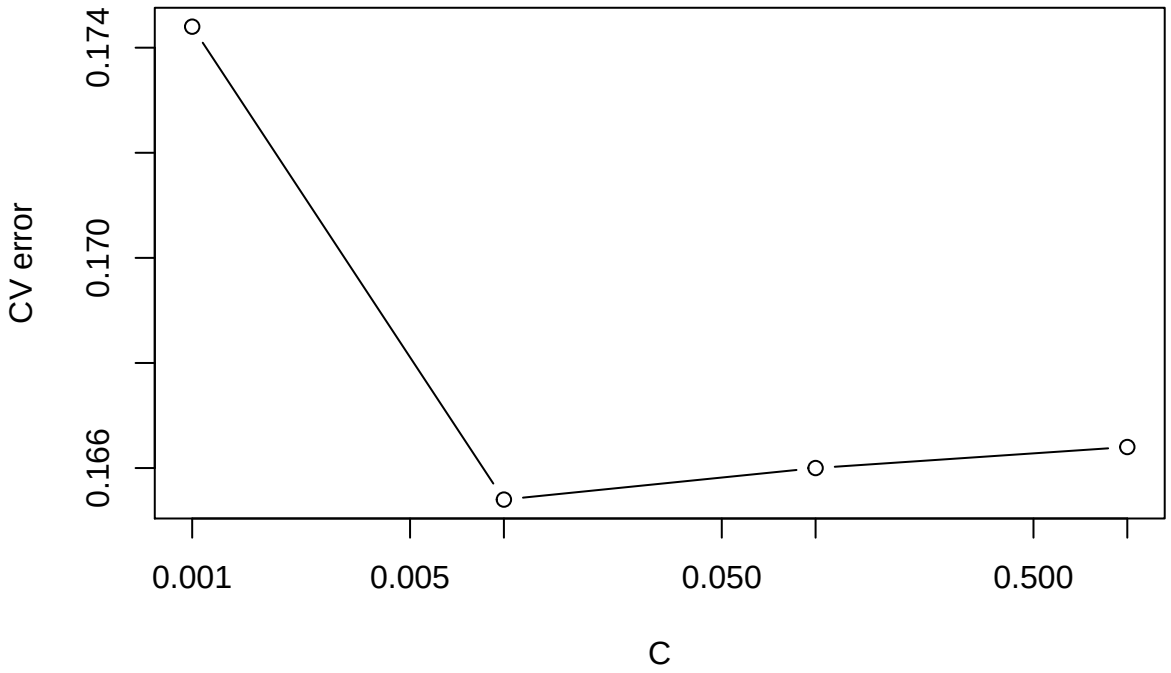
```
credit1<-credit[,-c(2,12:14)]
credit1.train<-credit1[train,]
credit1.test<-credit1[-train,]
y.test<-credit1.test$CARDHLDR
```

Let us first consider a linear classifier. We tune hyper-parameter C by 5-fold cross-validation:

```
CC<-c(0.001,0.01,0.1,1)
N<-length(CC)
err<-rep(0,N)
for(i in 1:N) err[i] <- cross(ksvm(CARDHLDR~.,data=credit1.train,scaled=TRUE,
                          kernel="vanilladot",C=CC[i],cross=5))
```

```
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
```

```
plot(CC,err,type="b",log="x",xlab="C",ylab="CV error")
```



We then fit the model with the minimum-error value of C, and compute the test error rate:

```
Cmin<-CC[which.min(err)]
fit<-ksvm(CARDHLDR~.,data=credit1.train,scaled=TRUE,kernel="vanilladot",C=Cmin)
```

```
##  Setting default kernel parameters
```

```
pred<-predict(fit,newdata=credit1.test)
1-mean(y.test==pred)
```

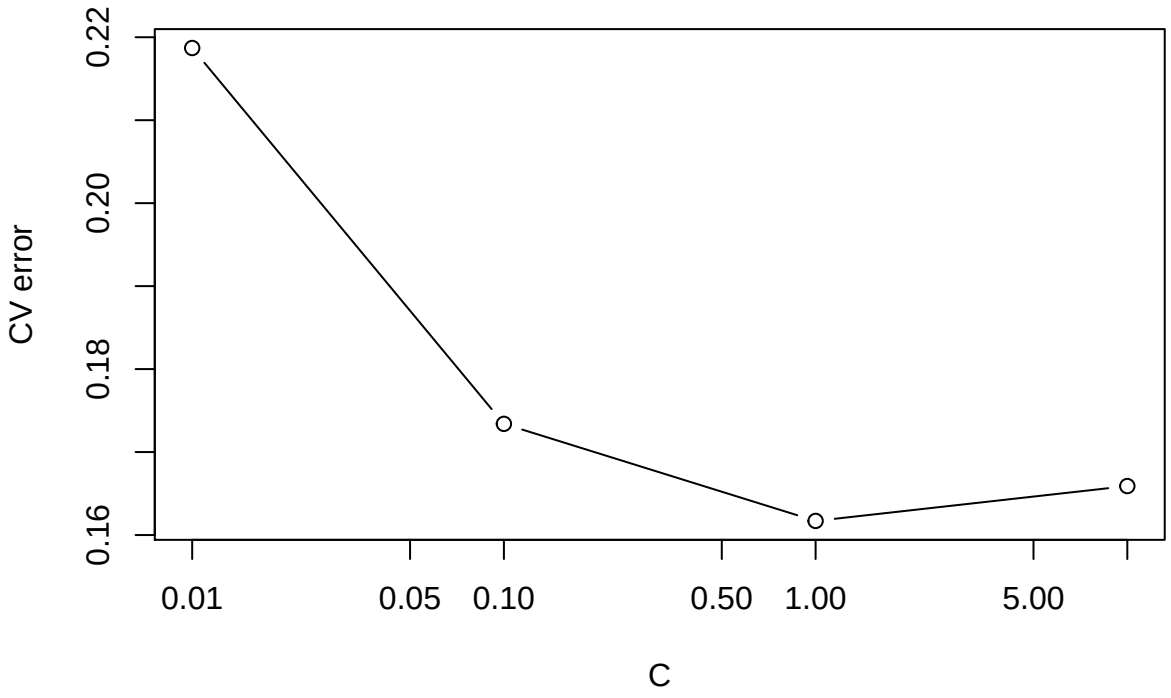
```
## [1] 0.170151
```

We now try a nonlinear SVM with an RBF kernel. Hyperparameter σ for the kernel is determined automatically by function `ksvm` using a heuristic rule:

```
CC<-c(0.01,0.1,1,10)
N<-length(CC)
err<-rep(0,N)
for(i in 1:N){
  print(i)
  err[i]<-cross(ksvm(CARDHLDR~.,data=credit1.train,scaled=TRUE,
                     kernel="rbfdot",C=CC[i],cross=5))
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

```
plot(CC,err,type="b",log="x",xlab="C",ylab="CV error")
```



Again, we fit the model on the whole training data with the best value of C:

```
fit <- ksvm(CARDHLDR~.,data=credit1.train,scaled=TRUE,kernel="rbfdot",C=1)
pred<-predict(fit,newdata=credit1.test)
1-mean(y.test==pred)
```

```
## [1] 0.1626016
```

The test error rate is only slightly less than that of the linear SVM. As a comparison, we can try a logistic regression classifier:

```
fit.lr<-glm(CARDHLDR~.,data=credit1.train,family="binomial")
pred.lr=predict(fit.lr,newdata=credit1.test,type="response")
yhat<-as.numeric(pred.lr>0.5)
1-mean(y.test==yhat)
```

```
## [1] 0.1649245
```

The performance of logististic regression is similar in this case.

Prediction of `DEFAULT`

We create a new data frame `credit2` containing only the observations for which variable `CARDHLDR` has value `TRUE`, and without variable 'CARDHLDR'. We then split the data:

```
ii<-which(credit$CARDHLDR==1)
credit2<-credit[ii,-1]
n<-nrow(credit2)
ntrain<-5000
ntest<-n-ntrain
train<-sample(n,ntrain)
credit2.train<-credit2[train,]
credit2.test<-credit2[-train,]
y.test<-credit2.test$DEFAULT
```

We can see that there are about ten times more observations from the negative class than observations from the positive class:

```
table(credit2$DEFAULT)
```

```
##
##    0    1
## 9503  996
```

We say that the data are imbalanced. A classifier always predictive the negative class (no default) will have an error rate as low as 996/(9503+996)=0.0949. A nonlinear linear SVM with $C = 1$ does not do better than that:

```
fit<-ksvm(DEFAULT~.,data=credit2.train,scaled=TRUE,kernel="rbfdot",C=1)
pred<-predict(fit,newdata=credit2.test)
table(y.test,pred)
```

```
##       pred
## y.test    0    1
##      0 4967    0
##      1  532    0
```

```
1-mean(y.test==pred)
```

```
## [1] 0.09674486
```

We $C = 100$, we get a higher error rate, but a better balance between false positive and false negative rate:

```
fit<-ksvm(DEFAULT~.,data=credit2.train,scaled=TRUE,kernel="rbfdot",C=1000)
pred<-predict(fit,newdata=credit2.test)
table(y.test,pred)
```

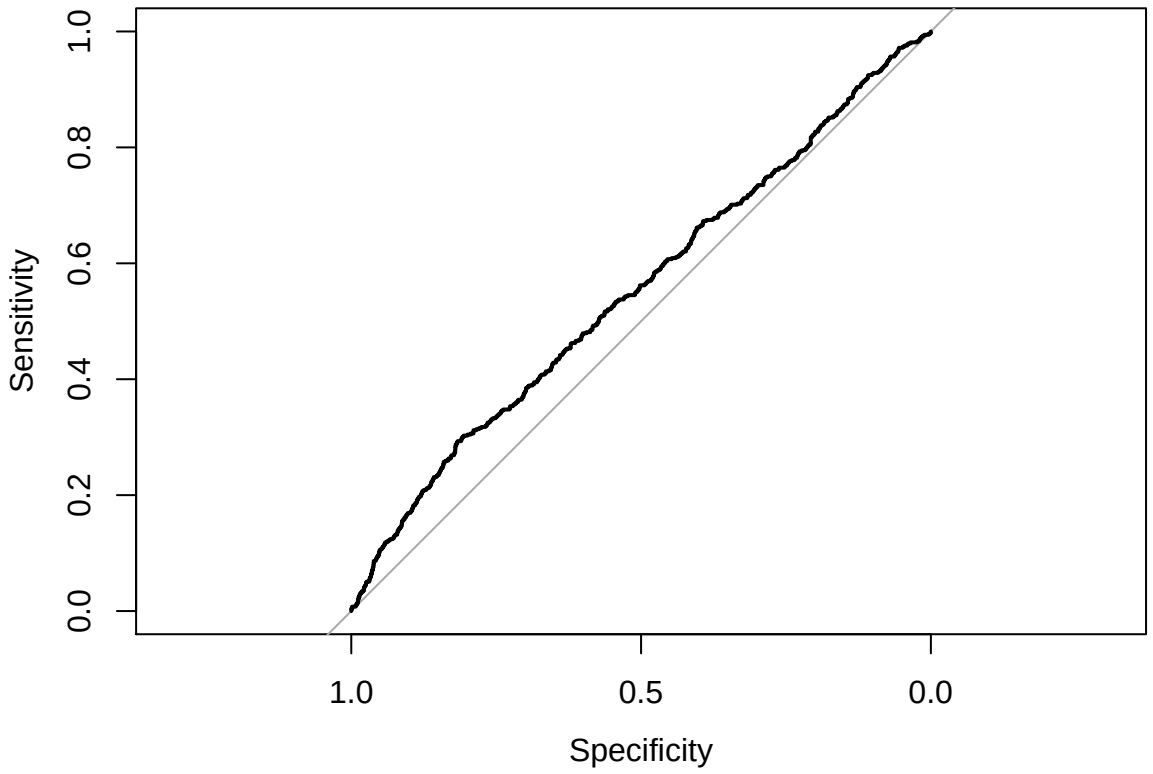
```
##       pred
## y.test    0    1
##      0 4718  249
##      1  476   56
```

```
1-mean(y.test==pred)
```

```
## [1] 0.1318422
```

We can plot the ROC cirve of this classifier:

```
library(pROC)
pred<-predict(fit,newdata=credit2.test,type="decision")
roc.svm<-roc(response=y.test,predictor=pred)
plot(roc.svm)
```



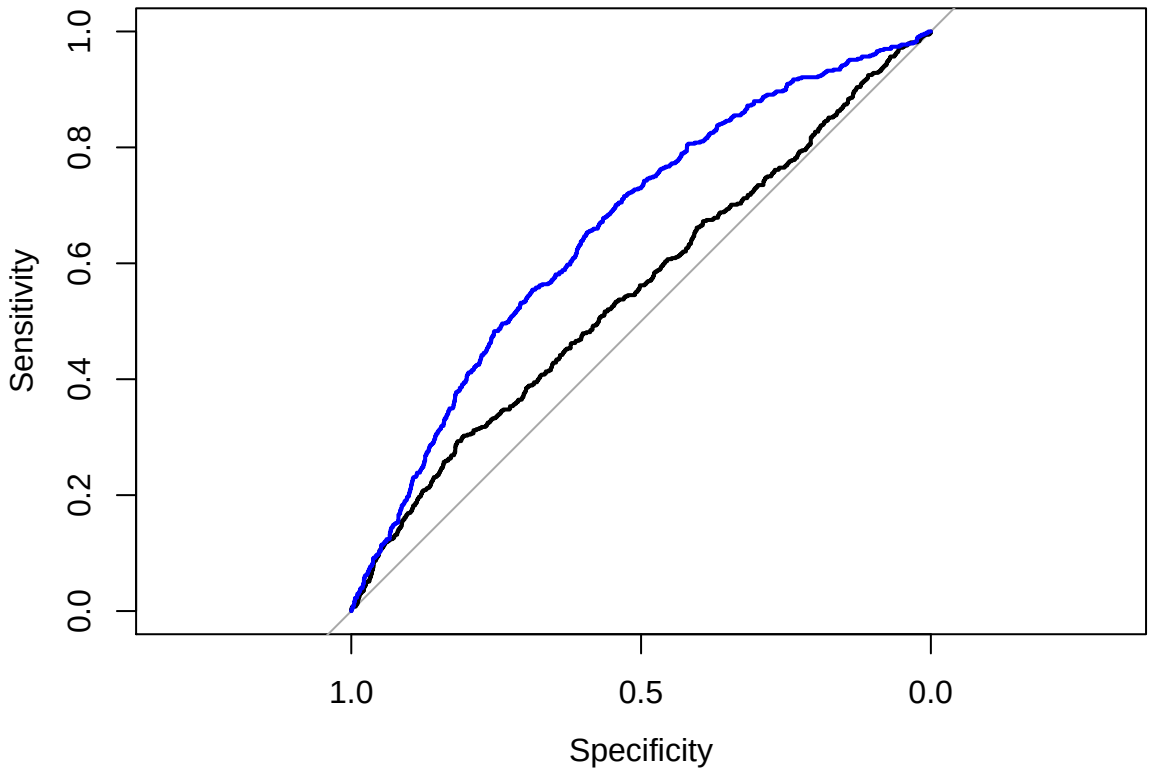
This curve can be compared to that of a logistic regression classifier:

```
fit.lr<-glm(DEFAULT~.,data=credit2.train,family="binomial")
pred.lr=predict(fit.lr,newdata=credit2.test,type="response")
roc.log<-roc(response=y.test,predictor=pred.lr)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc.svm)
plot(roc.log,add=TRUE,col="blue")
```



Logistic regression is clearly superior in this case.

Exercise 2

Preprocessing:

```
library(MASS)
n<-nrow(Boston)
ntrain<-round(2*n/3)
ntest<-n-ntrain
train<-sample(n,ntrain)
Boston.train<-Boston[train,]
Boston.test<-Boston[-train,]
y.test<-Boston.test$medv
```

Linear SVR:

```
fit<-ksvm(medv~.,data=Boston.train,scaled=TRUE,kernel="vanilladot",C=10)
```

```
##  Setting default kernel parameters
```

```
pred.svr1<-predict(fit,newdata=Boston.test)
mse.svr1<-mean((y.test-pred.svr1)^2)
print(mse.svr1)
```

```
## [1] 24.41289
```

Nonlinear SVR:

```
fit<-ksvm(medv~.,data=Boston.train,scaled=TRUE,kernel="rbfdot",C=10)
pred.svr2<-predict(fit,newdata=Boston.test)
```

```
mse.svr2<-mean((y.test-pred.svr2)^2)
print(mse.svr2)
```

```
## [1] 11.02041
```

LS linear regression:

```
reg<- lm(medv ~. ,data=Boston.train)
pred<-predict(reg,newdata=Boston.test)
mse.ls<-mean((y.test-pred)^2)
print(mse.ls)
```

```
## [1] 19.49426
```

Random forest:

```
library(randomForest)
fit.rf<-randomForest(medv~.,data=Boston.train)
pred.rf<-predict(fit.rf,newdata=Boston.test)
mse.rf<-mean((y.test-pred.rf)^2)
print(mse.rf)
```

```
## [1] 8.933704
```

The nonlinear SVR model performs much better than the linear models, but it is still outperformed by the random forest regressor.