Data Mining Homework 3

Shan Jiang

10/27/2019

Contents

Problem 2

Load packages

```
library(tidyverse)
library(broom)
library(kernlab)       # SVM methodology
library(e1071)         # SVM methodology
library(glmnet)
require(ggplot2)
```

Read in data

```
## 250 obs. train data
train_data <- read.delim("./synth.tr", header = T, sep = "")
## 1000 obs. test data
test_data <- read.delim("./synth.te", header = T, sep = "")
```

There are in total in the train dataframe, containing two predictors and a binary outcome.

(a) Construct a linear support vector classifier

Model

We must encode the response as a factor variable.

```
train_data = train_data %>%
             mutate(yc= as.factor(yc))
test_data = test_data %>%
             mutate(yc= as.factor(yc))
```

using `support vector classifier`, in this case, we try to tune the best parameter:

```
## Using scaled data
set.seed (1)
tune.out = tune(svm, yc~.,
                data= train_data,
                kernel="linear",
                ranges= list(cost=c(0.1,1,10,100,1000),
                gamma= c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.14
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.5 0.152 0.09003703
## 2  1e+00   0.5 0.140 0.07831560
## 3  1e+01   0.5 0.140 0.07831560
## 4  1e+02   0.5 0.140 0.07831560
## 5  1e+03   0.5 0.140 0.07831560
## 6  1e-01   1.0 0.152 0.09003703
## 7  1e+00   1.0 0.140 0.07831560
## 8  1e+01   1.0 0.140 0.07831560
## 9  1e+02   1.0 0.140 0.07831560
## 10 1e+03   1.0 0.140 0.07831560
## 11 1e-01   2.0 0.152 0.09003703
## 12 1e+00   2.0 0.140 0.07831560
## 13 1e+01   2.0 0.140 0.07831560
## 14 1e+02   2.0 0.140 0.07831560
## 15 1e+03   2.0 0.140 0.07831560
## 16 1e-01   3.0 0.152 0.09003703
## 17 1e+00   3.0 0.140 0.07831560
## 18 1e+01   3.0 0.140 0.07831560
## 19 1e+02   3.0 0.140 0.07831560
## 20 1e+03   3.0 0.140 0.07831560
## 21 1e-01   4.0 0.152 0.09003703
```

```
## 22 1e+00   4.0 0.140 0.07831560
## 23 1e+01   4.0 0.140 0.07831560
## 24 1e+02   4.0 0.140 0.07831560
## 25 1e+03   4.0 0.140 0.07831560
```

```r
svmfit1 = tune.out$best.model
#summary(svmfit1)
```

We see that `cost = 1` results in the lowest cross-validation error rate. The `tune()`function stores the best model obtained.

Test error and its standard error

```r
ypred1 = predict(svmfit1,test_data, type = "response")

test.error1 <- 1- sum(ypred1 == test_data$yc)/nrow(test_data)
test.error1
```

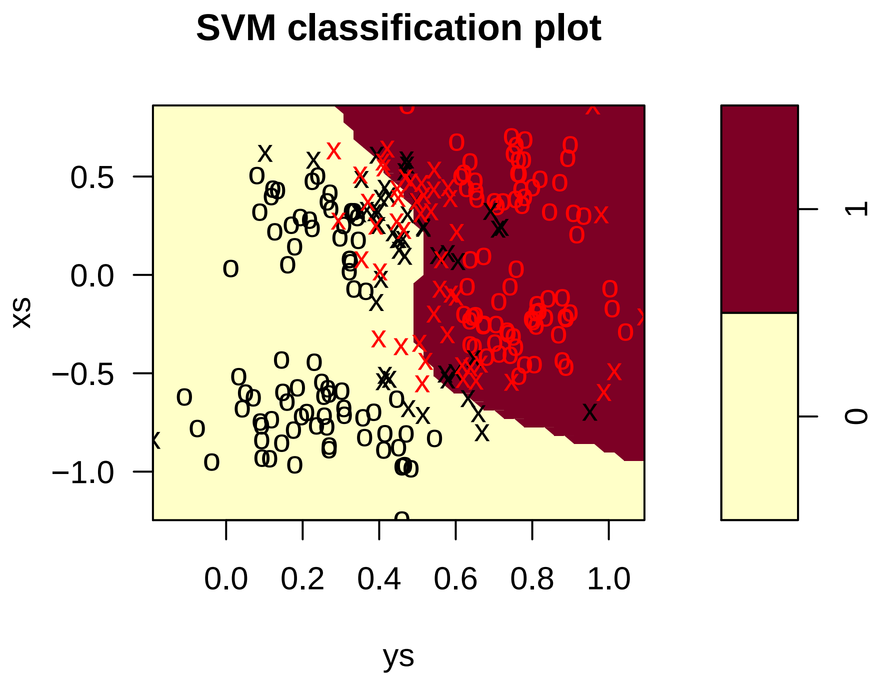
```
## [1] 0.107
```

```r
std.error1 <- sqrt((test.error1 * (1- test.error1))/(nrow(test_data)))
std.error1
```

```
## [1] 0.009775019
```

```r
## use the best-fit parameter to fit the model
svmfit1_model = svm(yc~., data = train_data,
           kernel="radial", gamma=0.5,cost = 1)

# summary(svmfit1_model)
plot(svmfit1_model, train_data)
```



(b) Construct a support vector classifier with Radial kernel

Tune parameter

Perform cross-validation using `tune()` to select the best choice.

```
set.seed (1)
tune.out2 = tune(svm, ys~., data=train_data,
                 kernel="radial",
            ranges= list(cost=c(0.1,1,10,100,1000),
            gamma= c(0.5,1,2,3,4)))
#summary(tune.out2)
```

Therefore, the best choice of parameters involves `cost=1` and `gamma=0.5`.

```
## use the best-fit parameter to fit the model
svmfit2 = svm(yc~., data = train_data,
           kernel="radial", gamma=0.5,cost =1)

#summary(svmfit2)
```

This tells us, for instance, that a linear kernel was used with cost = 1, and that there were 95 support vectors, 47 in one class and 48 in the other.

Test error and its standard error

```r
ypred2 = predict(svmfit2,
                 newdata = test_data, type = "response")

ypred2 = predict(svmfit2,test_data, type = "response")

test.error2 <- 1- sum(ypred2 == test_data$yc)/nrow(test_data)
test.error2
```

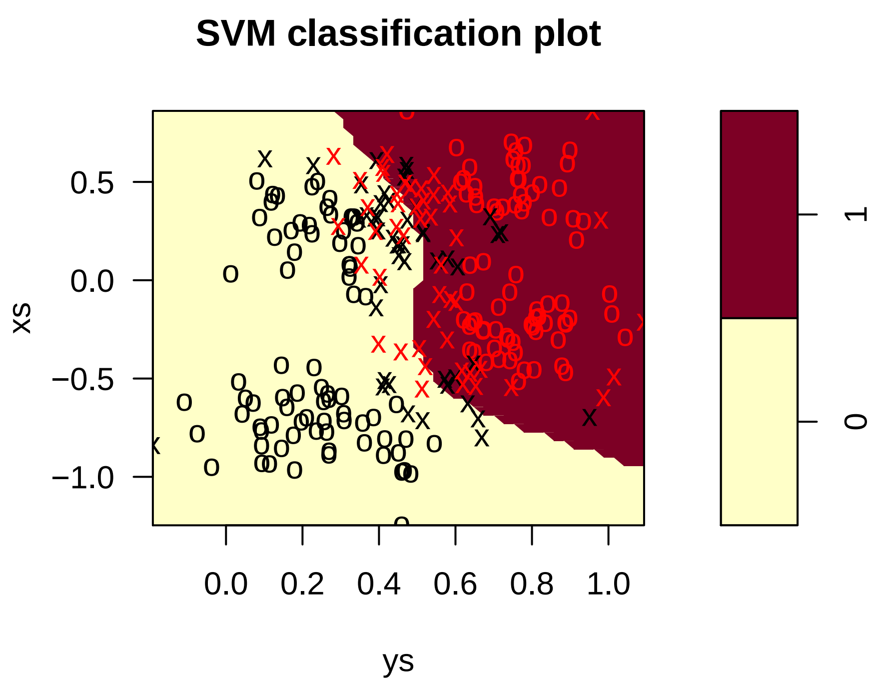
```
## [1] 0.095
```

```r
std.error2 <- sqrt((test.error2 * (1- test.error2))/(nrow(test_data)))
std.error2
```

```
## [1] 0.00927227
```

Plots

```r
plot(svmfit2, train_data)
```



(c) AdaBoost algorithm-classifier

Boosting is another technique which assumes a weak classifier technique at the begining, while it attains the true classification.

```r
#install.packages("gbm")
library(ada)
```

```
## Loading required package: rpart
```

```r
## use of gradient/generalized boosting models in "gbm".
set.seed(1118)
fit.adaboost <- ada(yc ~., data=train_data,
                    iter=50, nu = 0.1, bag.frac=0.5)
summary(fit.adaboost)
```

```
## Call:
## ada(yc ~ ., data = train_data, iter = 50, nu = 0.1, bag.frac = 0.5)
##
## Loss: exponential Method: discrete   Iteration: 50
##
## Training Results
##
## Accuracy: 0.912 Kappa: 0.824
```

Test Error and std.error

```r
ypred3 = predict(fit.adaboost,newdata = test_data)

test.error3 <- 1- sum(ypred3 == test_data$yc)/nrow(test_data)
test.error3
```

```
## [1] 0.098
```

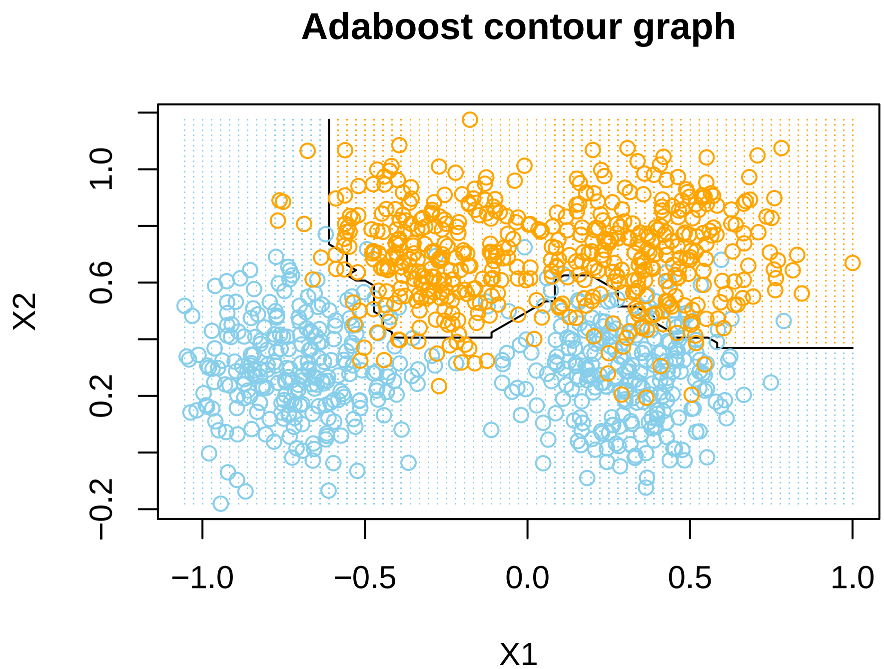
```r
std.error3 <- sqrt((test.error3 * (1- test.error3))/(nrow(test_data)))
std.error3
```

```
## [1] 0.009401915
```

```r
## Plotting for adaboost
plot2 = function(model,df_train){
  train = df_train[,1:2]
  L=75
  X=seq(min(train[,1]),max(train[,1]),length=L)
  Y=seq(min(train[,2]),max(train[,2]),length=L)
  XY=expand.grid(X,Y) %>% rename(xs=Var1,ys=Var2)
  yTrain = df_train$yc
  yhat=predict(model,XY)
  colors <- c("SkyBlue", "Orange")
  yhat1 <- colors[as.numeric(yhat)]
  yTrain <- colors[as.numeric(yTrain)]
  plot(train, xlab="X1", ylab="X2",
       xlim = range(train[,1]),
       ylim = range(train[,2]), type="n")
  points(XY,col=yhat1, pch=15,cex=0.1)
```

```
  contour(X, Y, matrix(as.numeric(yhat),L,L),
          levels=c(1,2), add=TRUE, drawlabels=FALSE)
  points(train, col=yTrain)
  title("Adaboost contour graph")
}

plot2(fit.adaboost, test_data)
```



Summary

```
test.error = c(test.error1, test.error2, test.error3)
sd = c(std.error1, std.error2, std.error3)
method = c("linear support vector classifier",
           "Radial Kernel support vector classifier",
           "Adaboost alogorithm")
summary = data.frame(method,test.error, sd)
knitr::kable(summary)
```

method	test.error	sd
linear support vector classifier	0.107	0.0097750
Radial Kernel support vector classifier	0.095	0.0092723
Adaboost alogorithm	0.098	0.0094019

Comment:

- Three models have slightly different classification test error results, while they vary only a little bit, the linear SVM is of less good fit compared with SVM machine with Kernel model, while Adaboost alogorithm perform best.
- The test error and standard deviation are lowest for Adaboost classifier, but we need to be cautious of the overfitting risk.