HW9

Cathy Su

8/11/2019

Q2

(a)

Use a support vector machine to classify emails as spam or ham. The e1071 package on CRAN provides a svm function that will be useful. Start by using the default parameters of the svm function and report your test-set accuracy.

```
# data
spam <- read.csv("../data/gam-spam.csv", header=FALSE)
spam$V58 <- as.factor(spam$V58)
training_rows <- sample.int(nrow(spam), round(nrow(spam) / 3))

spam.train <- spam[training_rows, ]
spam.test <- spam[-training_rows, ]

# apply log transform
cols <- names(spam.train)[1:(length(spam.train) - 1)]
for (col in cols) {
  spam.train[col] <- log(0.1 + spam.train[[col]])
  spam.test[col] <- log(0.1 + spam.test[[col]])
}
#
fit <- svm(V58 ~ ., data = spam.train)
#summary(fit)
pred <- predict(fit, newdata = spam.test, probability = TRUE)
table(spam.test$V58, pred)
```

```
##    pred
##        0    1
##   0    0 1844
##   1    0 1223
```

(b)

By default, svm uses a radial kernel, but it also supports linear, polynomial, and sigmoid kernels. Each has different tuning parameters, such as the degree of the polynomial kernel or gamma. perform cross-validation to select the best tuning parameters. Build the best classifier for each kernel, tuning over reasonable ranges of the tuning pa- rameters, and report your results. Which kernel performs best on this data? Which kernel performs worst, and why might it be the worst?

These are the tuning parameters for each kernel:

- radial: cost, gamma
- linear: cost
- polynomial: cost, coef0, degree, gamma
- sigmoid: cost, coef0, gamma

I found that tuning the parameters in different function calls vs together seemed to give the same results. Therefore I tuned once, and then printed the outputs as below. The best performing kernel seems to be the

```
type <- c("linear", "polynomial","sigmoid", "radial")

params <- tune.svm(V58 ~ ., data=spam.train,
                   cost=10^(0:4),
                   gamma=10^(-4:-1),
                   coef0=10^(-1:2),
                   degree=1:4)

####### polynomial
for(k in type){
  print(k)
  fit <- svm(V58 ~ ., data = spam.train,
             kernel = k,
             degree=params$best.parameter[[1]],
             cost=params$best.parameter[[4]],
             coef0=params$best.parameter[[3]],
             gamma=params$best.parameter[[2]],
             probability = TRUE)
  #summary(fit)
  pred <- predict(fit, newdata = spam.test)
  t <- table(spam.test$V58, pred)/length(pred)
  print(kable(t))
}
```

```
## [1] "linear"
##
##
##               0           1
## ---  ----------  ----------
## 0     0.5679817   0.0332573
## 1     0.0322791   0.3664819
## [1] "polynomial"
##
##
##               0           1
## ---  ----------  ----------
## 0     0.5758070   0.0254320
## 1     0.0384741   0.3602869
## [1] "sigmoid"
##
##
##               0           1
## ---  ----------  ----------
## 0     0.5735246   0.0277144
## 1     0.0388001   0.3599609
## [1] "radial"
##
##
##               0           1
## ---  ----------  ----------
## 0     0.5787414   0.0224976
## 1     0.0348875   0.3638735
```

(c)

Compare the accuracyyouhaveobtainedheretotheaccuracyyouobtainedonHomework 8 while using random forests

We see that based on the confusion matrix of each type, the

Q3. Degrees of freedom of a tree.

An Introduction to Statistical Learning, chapter 9, exercise 5 (pp. 369–370). For part (i), when commenting on your results, answer these questions: * In what space is the decision boundary linear? * If we can make logistic regression produce nonlinear decision boundaries by adding ap- propriate covariates, why is the kernel trick in SVMs useful?

We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.

a) Generate a data set with n = 500 and p = 2, such that the obser- vations belong to two classes with a quadratic decision boundary between them. For instance, you can do this as follows:

```
x1=runif(500)-0.5
x2=runif(500)-0.5
y=1*(x1^2-x2^2 > 0)
```

b) Generate 100 observations with predictors X1, X2, . . . , X10 as inde- pendent standard Gaussian variates and fix these values.

c) Generate response values also as standard Gaussian ($\sigma^2 = 1$), indepen- dent of the predictors. Fit regression trees to the data of fixed size 1,5 and 10 terminal nodes and hence estimate the degrees of freedom of each fit. [Do ten simulations of the response and average the results, to get a good estimate of degrees of freedom.]

d) Compare your estimates of degrees of freedom in (a) and (c) and discuss.

e) Suggest a way to compute an approximate S matrix for a regression tree, compute it and compare the resulting degrees of freedom to those in (a) and (c).

Q4

Should we set C large or small if we want our estimated boundary to have the minimum possible variance?

If the parameter C is large, then any wrong classification is penalized a lot. This would lead to higher variance since the classification boundary tries to eliminate every single outlier. Thus to have smaller variance we should set C to be small.

Q5

Unbalanced classes. Simulate a dataset with two variables, X1 and X2, and an outcome variable Y. Make the dataset have a linear decision boundary between Y = 1 and Y = 0. But put the code that simulates data into a function, and make one of the parameters of that function be the fraction of cases that should have Y = 1. This way, we can simulate what happens when the classes are imbalanced.

```
simulate <- function(fraction=0.5) {
  X <- matrix(runif(2 * N, min=0, max=10), nrow=N, ncol=2)
  Y <- ifelse((X[, 1] < 5 & X[, 2] < 5) | (X[, 1] > 5 & X[, 2] > 5), rbinom(N, 1, 0.1), rbinom(N, 1, 0.9

  stopifnot(nrow(X) == length(Y)) # sanity check!
  new <- cbind(as.data.frame(X), as.dsata.frame(Y))
  new$Y <- as.factor(new$Y)
  #colnames(new)[-1] <- "Y"
  return(new)
}
```

a)

Set the fraction to 0.5: perfect balance. Run a logistic regression and an SVM on the data to predict Y. Compare the confusion matrices on a test set (just run your function again to get a test set!). How do the error rates compare, and do the error rates differ for points with Y = 0 vs. those with Y = 1?

b)

Now repeat this analysis with fractions varying from 0.6 to 0.95. Look at the confusion matrices as the classes become less balanced.What happens to the errors for logistic regression? What kinds of errors become more common?

c)

What happens to the errors for SVMs in the same case? Show a graph or table summarizing the results and comparing SVMs to logistic regression.

d)

Describe, in words, why the difference you observed should exist. Give your explanation in terms of how SVMs and logistic regression find the decision boundary, and what loss functions they use.