Lab_09R

36-290 – Statistical Research Methodology

Week 9 Thursday – Fall 2021

Data

Below we read in the pulsar dataset we used last time. Except…SVM is **slow**. Very slow. Order n-cubed slow. It is not an algorithm made for big data. So we will do what we did last time: construct a smaller dataset, this time with sample size 1638 that has the response variable evenly split between classes.

```
rm(list=ls())
file.path = "http://www.stat.cmu.edu/~pfreeman/pulsar.Rdata"
load(url(file.path))
rm(file.path)
set.seed(406)
w.0 = which(response$X9==0)
w.1 = which(response$X9==1)
s = sample(length(w.0),length(w.1)/2)
predictors = predictors[c(w.0[s],w.1[1:length(s)]),]
response = as.character(response$X9)[c(w.0[s],w.1[1:length(s)])]
predictors = scale(predictors)
predictors = data.frame(predictors)
cat("Number of predictor variables: ",ncol(predictors),"\n")
```

```
## Number of predictor variables:  8
```

```
cat("Sample size:                   ",nrow(predictors),"\n")
```

```
## Sample size:                    1638
```

```
response = factor(response,labels=c("NO","YES"))
contrasts(response)
```

```
##     YES
## NO    0
## YES   1
```

```
#No is 0, Yes 1
```

The eight predictors are summary statistics that describe the distribution of brightness measurements of a pulsar candidate (mean, standard deviation, skewness, kurtosis) as well as the distribution of “dispersion measure” readings (also mean, standard deviation, skewness, kurtosis).

The response is either “NO” (the candidate is *not* a pulsar) or “YES”.

Questions

Question 1

Split the data and perform a basic logistic regression analysis. (Yes, logistic regression…you are establishing a baseline and seeing if SVM can beat it.) You just need to output the test-set MCR and the confusion matrix.

```
response = data.frame(response)

set.seed(100)
fraction=.7
sp = sample(nrow(predictors), round(fraction*nrow(predictors)))
pred.train = predictors[sp ,]
pred.test = predictors[-sp ,]

resp.train = response[sp, ]
resp.test = response[-sp ,]

#logistic regression model using training data
glm.fit = glm(resp.train~.,data=pred.train,family="binomial")
summary(glm.fit)
```

```
##
## Call:
## glm(formula = resp.train ~ ., family = "binomial", data = pred.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2026  -0.2769   0.0026   0.0306   3.0902
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.7225     0.7258   2.373 0.017630 *
## profile.mean     0.2588     0.5610   0.461 0.644586
## profile.stddev   0.1348     0.2452   0.550 0.582413
## profile.skew    12.9962     1.7297   7.514 5.75e-14 ***
## profile.kurt    -7.5717     2.2379  -3.383 0.000716 ***
## dm.mean         -1.1242     0.3952  -2.845 0.004446 **
## dm.stddev        1.7453     0.4925   3.543 0.000395 ***
## dm.skew          2.6744     1.3175   2.030 0.042369 *
## dm.kurt         -2.9270     1.1454  -2.555 0.010606 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1589.76  on 1146  degrees of freedom
## Residual deviance:  377.02  on 1138  degrees of freedom
## AIC: 395.02
##
## Number of Fisher Scoring iterations: 11
```

```
predicted = predict(glm.fit, pred.test, type="response")

glm.pred=rep("NO", length(response))
glm.pred[predicted >.5]="YES"
tab = table(glm.pred,resp.test)
tab
```

```
##         resp.test
## glm.pred  NO YES
##      NO    1   0
##      YES  10 215
```

```
(10)/(215+1+10)
```

```
## [1] 0.04424779
```

```
#MCR: 4.424
```

Question 2

We will work with the `e1071` package. (Its name comes from the coding for the Institute of Statistics and Probability Theory at the Technische Universitat Wien, in Vienna. It's like us calling a package `36-290`. Which we should.)

Here, code a support vector classifier (meaning, use `kernel="linear"`): use the `tune()` function with a representative sequence of potential costs C, then extract the best model. If the optimum value of C occurs at or very near the end of your sequence of potential costs, alter the sequence. The variable `best.parameters`, embedded in the output, provides the optimal value for C. Provide that value. Use the best model to generate predictions, a test-set MCR, and a confusion matrix. Does the support vector classifier "beat" logistic regression? How do the results differ?

Note: `e1071` is prickly about wanting the response vector to be part of the predictor data frame. To join the predictors and response together, do the following: `pred.train = cbind(pred.train,resp.train)`. `cbind()` means "column bind."

Note that `tune()` does cross-validation on the training set to estimate the optimum value of C. Which means that the training data are randomly assigned to folds (by default, 10…to change this, you'd make a call like `tune.control(cross=5)`). Which means you should set a random number seed before calling `tune()`. For reproducibility n'at.

See the third code block of page 364 of `ISLR` for an example of how to specify ranges of tuning parameters. Note there is only one here: `cost`. As for prediction: `tune()` will return an object that includes `best.model`. Pass this to `predict()` along with the argument `newdata=` whatever you call the test predictors data frame. By default, `predict()` will output a vector of class predictions, so there is no need to round off to determine classes.

```
response = data.frame(response)

set.seed(100)
fraction=.7
sp = sample(nrow(predictors), round(fraction*nrow(predictors)))
pred.train = predictors[sp ,]
pred.test = predictors[-sp ,]

resp.train = response[sp, ]
resp.test = response[-sp ,]

pred.train = cbind(pred.train,resp.train)


library(e1071)
set.seed(202) # reproducible cross-validation
tune.out = tune(svm,resp.train~.,data=pred.train,kernel="linear",ranges=list(cost=10^seq(-2,2,by=0.2)))
cat("The estimated optimal value for C is ",as.numeric(tune.out$best.parameters),"\n")
```

```
## The estimated optimal value for C is  100
```

```
names(tune.out)
```

```
## [1] "best.parameters"  "best.performance" "method"           "nparcomb"
## [5] "train.ind"        "sampling"         "performances"     "best.model"
```

```
resp.pred = predict(tune.out$best.model,newdata=pred.test)


mean(resp.pred!=resp.test)
```

```
## [1] 0.05702648
```

```
table(resp.pred,resp.test)
```

```
##          resp.test
## resp.pred  NO YES
##       NO  247  20
##       YES   8 216
```

```
#Use the best model to generate predictions, a test-set MCR, and a confusion matrix. Does the support vector classi
fier "beat" logistic regression? How do the results differ?
```

```
The estimated optimal value for C is  100


MCR 5.7%
(8+20)/(247+20+8+216)


The support vector classifier doesn't "beat" logistic regression in the sense that the MCR iss higher
```

Question 3

Now code a support vector machine with a polynomial kernel. In addition to tuning `cost`, you also have to tune the polynomial `degree`. Try integers from 2 up to some maximum number (not too large, like 4). How do the results change? (Note: if you get the warning `WARNING: reaching max number of iterations`, do not worry about it.)

```
response = data.frame(response)

set.seed(100)
fraction=.7
sp = sample(nrow(predictors), round(fraction*nrow(predictors)))
pred.train = predictors[sp ,]
pred.test = predictors[-sp ,]

resp.train = response[sp, ]
resp.test = response[-sp ,]


pred.train = cbind(pred.train,resp.train)


set.seed(202)
tune.out = tune(svm,resp.train~.,data=pred.train,kernel="polynomial",
                ranges=list(cost=10^seq(0,4,by=0.5),degree=2:4))
cat("The estimated optimal values for C and degree are ",as.numeric(tune.out$best.parameters),"\n")
```

```
## The estimated optimal values for C and degree are  10 3
```

```
resp.pred = predict(tune.out$best.model,newdata=pred.test)
(svm.poly.mcr = mean(resp.pred!=resp.test))
```

```
## [1] 0.05702648
```

```
table(resp.pred,resp.test)
```

```
##          resp.test
## resp.pred  NO YES
##       NO  247  20
##       YES   8 216
```

```
MCR is the same 0.05702648
```

Question 4

Now code a support vector machine with a radial kernel. In addition to tuning `cost`, you also have to tune the parameter `gamma`. Try a base-10 logarithmic sequence of values that includes -3 (for $10^{-3} = 0.001$). How do the results change?

```
response = data.frame(response)

set.seed(100)
fraction=.7
sp = sample(nrow(predictors), round(fraction*nrow(predictors)))
pred.train = predictors[sp ,]
pred.test = predictors[-sp ,]

resp.train = response[sp, ]
resp.test = response[-sp ,]

pred.train = cbind(pred.train,resp.train)


set.seed(202)
tune.out = tune(svm,resp.train~.,data=pred.train,kernel="radial",
                ranges=list(cost=10^seq(-1,1,by=0.5),gamma=10^seq(-1,1,by=0.4)))
cat("The estimated optimal values for C and gamma are ",as.numeric(tune.out$best.parameters),"\n")
```

```
## The estimated optimal values for C and gamma are  10 0.6309573
```

```
resp.pred = predict(tune.out$best.model,newdata=pred.test)
(svm.poly.mcr = mean(resp.pred!=resp.test))
```

```
## [1] 0.06517312
```

```
table(resp.pred,resp.test)
```

```
##          resp.test
## resp.pred  NO YES
##       NO  243  20
##       YES  12 216
```

```
MCR:0.06517312, higher than the previous ones
```