Homework 1 - Classification

Jeff Tilton

8/20/2018

Question 2.1

The US midterm elections are approaching and the Cambridge Analytica scandal is an example of the extent to which data science and machine learning have influenced politics in the United States. Get out the vote efforts are important in tight elections when control of a Congressional chamber is in play. The ability to classify voters into categories of supporter, non-supporter and likely or unlikely voter is a powerful tool for campaigns. The analytics company Cambridge Analytica used data from Facebook to target voters and likely had hundreds, if not thousands of predictors to choose from to categorize voters. Example predictors to catagorize a potential supporter are:

1. Religious beliefs
2. Education level
3. Sex
4. Income
5. Geographic location

Question 2.2

SVM

```
library("kernlab")
```

```
## Warning: package 'kernlab' was built under R version 3.2.5
```

```
data = as.matrix(read.table('credit_card_data-headers.txt', header = TRUE, sep = '\t'))
head(data)
```

```
##      A1    A2    A3   A8 A9 A10 A11 A12 A14 A15 R1
## [1,]  1 30.83 0.000 1.25  1   0   1   1 202   0  1
## [2,]  0 58.67 4.460 3.04  1   0   6   1  43 560  1
## [3,]  0 24.50 0.500 1.50  1   1   0   1 280 824  1
## [4,]  1 27.83 1.540 3.75  1   0   5   0 100   3  1
## [5,]  1 20.17 5.625 1.71  1   1   0   1 120   0  1
## [6,]  1 32.08 4.000 2.50  1   1   0   0 360   0  1
```

Split data into train and test

```
## the below is from https://stackoverflow.com/a/17200430/4296857
## 75% of the sample size
smp_size = floor(0.75 * nrow(data))

## set the seed to make your partition reproducible
set.seed(123)
train_ind = sample(seq_len(nrow(data)), size = smp_size)

train = data[train_ind, ]
```

```
test = data[-train_ind, ]


range01 = function(x){(x-min(x))/(max(x)-min(x))}

x_train = matrix(c(range01(train[,5]),
                   range01(train[,10])),
                 nrow = nrow(train), ncol = 2)
y_train = train[,11]


x_test = matrix(c(range01(test[,5]),
                  range01(test[,10])),
                nrow = nrow(test), ncol =2)
y_test = test[,11]
```

Create model and coefficients

```
model = ksvm(x=x_train,y=y_train,type="C-svc",kernel="vanilladot",C=1,scaled=FALSE)
```

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

```
y_hat = predict(model, x_test)
a = colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##        X1        X2
## 2.0047657 0.6354243
```

```
a0 = -model@b
a0
```

```
## [1] -1.004766
```

Determine model accuracy

```
sum(y_hat == test[,11]) / nrow(test)
```

```
## [1] 0.8963415
```

A9 and A15 were the only significant weights so I filtered the data and rerun the SVM model. The model accuracy was equal to the earlier model using all attributes. C was stable for a large range in the first model using all of the predictors and maintained an accuracy of .896 between 10^-2 and 10^4. I decided this was the better model because of its simplicity. I plotted the data below for visualization.

```
library("ggplot2")
library("data.table")

data = as.data.table(train)
data = data.table(A9 = range01(data[,A9]),
                  A15 = range01(data[,A15]), R1 = as.factor(data[,R1]))


slope = a[1]/-a[2]
```

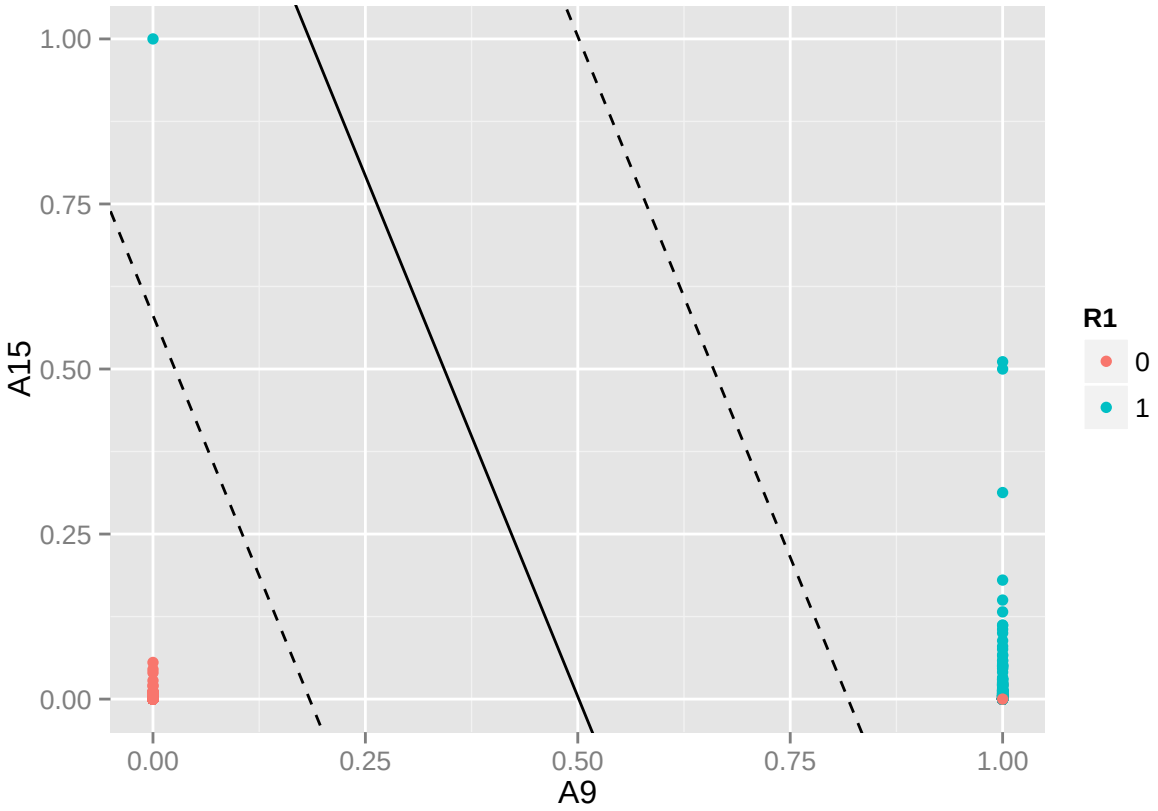
```
intercept = -a0/a[2]
gg = ggplot(data, aes(x=A9, y=A15, color = R1)) +
  geom_point()

gg=gg + geom_abline(intercept = intercept, slope = slope) +

  geom_abline(intercept = 1 + intercept,
              slope = slope,
              linetype = 'dashed') +

  geom_abline(intercept = -1 + intercept,
              slope = slope,
              linetype = 'dashed')

plot(gg)
```



I see from the above plot that A9 is the only significant predictor and A15 could be eliminated as well. The equation for my model is `y = -3.1550034x + 1.5812517`

KNN

```
library("kknn")
```

```
## Warning: package 'kknn' was built under R version 3.2.4
```

```
model = kknn(R1~.,
             as.data.frame(train),
```

```
             as.data.frame(test),
             k = 10, distance = 1,
             kernel = "triangular", scale = TRUE)
summary(model)
```

```
##
## Call:
## kknn(formula = R1 ~ ., train = as.data.frame(train), test = as.data.frame(test),     k = 10, distanc
##
## Response: "continuous"
```

```
y_hat = fitted(model)
```

Determine model accuracy

```
sum(round(y_hat) == test[,11]) / nrow(test)
```

```
## [1] 0.8597561
```