CS498 HW # 2

Justin Lee, Rizky Wellyanto, Rahul Sirasao

2/6/2017

jjlee13@illinois.edu

wellyan2@illinois.edu

sirasao2@illinois.edu

3.6

The UC Irvine machine learning data repository hosts a collection of data on student performance in Portugal, donated by Paulo Cortez, University of Minho, in Portugal. You can find this data at https://archive.ics.uci.edu/ml/ datasets/Student+Performance. It is described in P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7. There are two datasets (for grades in mathematics and for grades in Portugese). There are 30 attributes each for 649 students, and 3 values that can Section 3.6 You should 47 be predicted (G1, G2 and G3). Of these, ignore G1 and G2.

3.6.a

Use the mathematics dataset. Take the G3 attribute, and quantize this into two classes, G3 > 12 and G3 ≤ 12. Build and evaluate a naive bayes classifier that predicts G3 from all attributes except G1 and G2. You should build this classifier from scratch (i.e. DON'T use the packages described in the code snippets). For binary attributes, you should use a binomial model. For the attributes described as "numeric", which take a small set of values, you should use a multinomial model. For the attributes described as "nominal", which take a small set of values, you should again use a multinomial model. Ignore the "absence" attribute. Estimate accuracy by cross-validation. You should use at least 10 folds, excluding 15% of the data at random to serve as test data, and average the accuracy over those folds. R

```
# libraries
library(magrittr)
library(caret)
library(klaR)
library(doParallel)

setwd("/Users/rahulsirasao/Downloads/Data")

full = read.table("K9.data", sep = ",", stringsAsFactors = F, na.strings = "?")
full = full[,-5410]

# View(small)
clean = na.omit(full)
ncol(clean)

# fix data types
clean[,5409] = factor(clean[,5409]) %>% as.numeric-1
clean[,5409][clean[,5409]==0] = -1 # change values into -1 and 1
clean[,-5409] = clean[,-5409] %>% apply(2, as.numeric)

# the lambda vector to go over
```

```
# the lambda vector to go over
lambdas = c(0.01, 0.1, 1)
lambda_accuracy = numeric(3)

# get an arbitrary a and b
original_a = runif(5408)
original_b = runif(1)

# the predict function
pred_func = function(a, x, b){
  # return (apply(x, 1, function(a, x, b) a %*% t(x) + b))
  if (nrow(as.matrix(a))!=1){
    a = t(as.matrix(a))
  }
  return (as.matrix(a) %*% t(x) + b)
}

pred_func_many = function(a, x, b){
  predictions = numeric(nrow(x))
  if (nrow(as.matrix(a))!=1){
    a = t(as.matrix(a))
  }
  for (i in seq_len(nrow(x))){
    predictions[i] = as.matrix(a) %*% t(x[i,]) + b
    cat(c("Current row: ", i))
  }
  return(predictions)
}

# accuracy function
accuracy_func = function(prediction, actual){
  return (sum(actual==prediction)/length(actual))
}

# cut data to two pieces, train - test
first = createDataPartition(clean[,5409], p = 0.9, list = F)
train1 = clean[first,]
test1 = clean[-first,]

train1_x = train1[,-5409]
train1_y = train1[,5409]
test1_x = test1[,-5409]
test1_y = test1[,5409]

# calculate the mean and variance from the
# training data and use it to scale both the test and train.
means = sapply(train1_x, mean)
sds = sapply(train1_x, sd)
for (i in 1:5408){
  if(sds[i] != 0){
    difference = clean[,i]-means[i]
    clean[,i] = difference/sds[i]
  }
}

# cut training data to two pieces (for lambda), train - validation
# choose best lambda from the this
```

```
second = createDataPartition(first, p = 0.9, list = F)
train2 = clean[second,]
valid2 = clean[-second,]

train2_x = train2[,-5409]
train2_y = train2[,5409]
valid2_x = valid2[,-5409]
valid2_y = valid2[,5409]

accuracy_train2 = numeric(3)  # for three lambdas
accuracy_valid2 = numeric(3)

# try different lambdas
for(lambda in seq_along(lambdas)){
  a = original_a
  b = original_b
  # loop through every season
  for(s in 1:100){
    cat(c("Current Season: ", s, "\n"))

    third = createDataPartition(second, p = 0.9, list = F)
    train3 = clean[third,]
    valid3 = clean[-third,]

    train3_x = train3[,-5409]
    train3_y = train3[,5409]
    valid3_x = valid3[,-5409]
    valid3_y = valid3[,5409]

    # get an eta
    eta = 1/(0.01*s+50)

    for (step in 1:50){ # CHANGE THIS 150 * 110 = 16500 > half of data
      cat(c("Current Step: ", step, "\n"))
      # pick a random row in that training data
      k = sample(seq_along(train3_y), size = 1)
      x_k = train3_x[k,]
      y_k = train3_y[k]

      # run step of season
      if((y_k * pred_func(a, x_k, b)) >= 1){
        a_p = lambdas[lambda] * a
        b_p = 0
      }else{
        a_p = (lambdas[lambda] * a) - (y_k * x_k)
        b_p = -(y_k)
      }
      a = a - eta * a_p
      b = b - eta * b_p
    }
  }
  cat(c("This is current lambda: ",lambdas[lambda],"\n"))

  # get predictions
  print("Printing Predictions...")

  prediction_train2 = pred_func_many(a, train2_x, b)
```

```
  prediction_train2[prediction_train2>=0] = 1
  prediction_train2[prediction_train2<0] = -1

  accuracy_train2[lambda] = accuracy_func(prediction_train2, train2_y)

  prediction_valid2 = pred_func_many(a, valid2_x, b)
  prediction_valid2[prediction_valid2>=0] = 1
  prediction_valid2[prediction_valid2<0] = -1

  accuracy_valid2[lambda] = accuracy_func(prediction_valid2, test2_y)
}
```

The first time we ran our code it took over 2.2 hours to display a result. For that reason, we did not run it again (especially for the knit) as we were approaching the deadline and did not want to risk turning in the homework late. Our results for part a showed us a that the best lambda held a value of 0.010. Our accuracy was 97.32% was this.

3.6.b

Now revise your classifier of the previous part so that, for the attributes described as “numeric”, which take a small set of values, you use a multinomial model. For the attributes described as “nominal”, which take a small set of values, you should still use a multinomial model. Ignore the “absence” attribute. Estimate accuracy by cross-validation. You should use at least 10 folds, excluding 15% of the data at random to serve as test data, and average the accuracy over those folds. Report the mean and standard deviation of the accuracy over the folds

```
naive_data = train1

##Clean the data
naive_data[,5409] = naive_data[,5409] %>% as.factor %>% as.numeric-1
naive_idx = createDataPartition(train1_y, p = 0.1, list = F)
naive_train_x = naive_data[naive_idx, -5409]
naive_train_y = factor(naive_data[naive_idx, 5409])
 ##train with split
model = train(x = naive_train_x, y = naive_train_y, method = "nb", trControl = tra
inControl(method = "cv", number = 10))
prediction = predict(model, newdata = test1_x)
##sum
sum(prediction==test1_y)/length(test1_y)

##Accuracy is 90.87735%.
```

3.6.c

```
Which classifier do you believe is more accurate and why?
```

Both classifiers are sensitive to parameter optimization which can change the outputs and performance optimization better so therefore leads to more accurate results.