**Question 2.2**

The files credit\_card\_data.txt (without headers) and credit\_card\_data-headers.txt (with headers) contain a dataset with 654 data points, 6 continuous and 4 binary predictor variables. It has anonymized credit card applications with a binary response variable (last column) indicating if the application was positive or negative. The dataset is the “Credit Approval Data Set” from the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/Credit+Approval>) without the categorical variables and without data points that have missing values.

1. Using the support vector machine function ksvm contained in the R package kernlab, find a good classifier for this data. Show the equation of your classifier, and how well it classifies the data points in the full data set. (Don’t worry about test/validation data yet; we’ll cover that topic soon.)

Notes on ksvm

* You can use scaled=TRUE to get ksvm to scale the data as part of calculating a classifier.
* The term λ we used in the SVM lesson to trade off the two components of correctness and margin is called C in ksvm. One of the challenges of this homework is to find a value of C that works well; for many values of C, almost all predictions will be “yes” or almost all predictions will be “no”.
* ksvm does not directly return the coefficients a0 and a1…am. Instead, you need to do the last step of the calculation yourself. Here’s an example of the steps to take (assuming your data is stored in a matrix called data):[[1]](#footnote-1)

# # call ksvm. Vanilladot is a simple linear kernel.

model <- ksvm(data[,1:10],data[,11],type=”C-svc”,kernel=”vanilladot”,C=100,scaled=TRUE)

# calculate a1…am

# a <- colSums(model@xmatrix[[1]] \* model@coef[[1]])

a

# calculate a0

# a0 <- –model@b

a0

# see what the model predicts

# pred <- predict(model,data[,1:10])

pred

# see what fraction of the model’s predictions match the actual classification

# sum(pred == data[,11]) / nrow(data)

Hint: You might want to view the predictions your model makes; if C is too large or too small, they’ll almost all be the same (all zero or all one) and the predictive value of the model will be poor. Even finding the right order of magnitude for C might take a little trial-and-error.

Note: If you get the error “Error in vanilladot(length = 4, lambda = 0.5) : unused arguments (length = 4, lambda = 0.5)”, it means you need to convert data into matrix format:

model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type=”C-svc”,kernel=”vanilladot”,C=100,scaled=TRUE)

# read data

data <- read.delim2("credit\_card\_data\_headers.txt", header = TRUE, sep = "\t", dec = ",")

# access library kernlab

library(kernlab)

# Convert data into a numerical matrix

dataNumericalMatrix <- data.matrix(data[,1:10])

# Review data with boxplot

BoxPlot = boxplot(scale(dataNumericalMatrix, center = TRUE, scale = TRUE))

BoxPlot

# call ksvm. Vanilladot is a simple linear kernel.

model <- ksvm(dataNumericalMatrix,as.factor(data[,11]),type="C-svc",

kernel="vanilladot",C=100000,scaled=TRUE)

# calculate a1…am

a <- colSums(model@xmatrix[[1]] \* model@coef[[1]])

a

# calculate a0

a0 <- -model@b

a0

# see what the model predicts

pred <- predict(model,dataNumericalMatrix)

pred

# see what fraction of the model’s predictions match the actual classification

sum(pred == data[,11]) / nrow(data)

1. You are welcome, but not required, to try other (nonlinear) kernels as well; we’re not covering them in this course, but they can sometimes be useful and might provide better predictions than vanilladot.
2. Using the k-nearest-neighbors classification function kknn contained in the R kknn package, suggest a good value of k, and show how well it classifies that data points in the full data set. Don’t forget to scale the data (scale=TRUE in kknn).

# Notes on kknn

* You need to be a little careful. If you give it the whole data set to find the closest points to i, it’ll use i itself (which is in the data set) as one of the nearest neighbors. A helpful feature of R is the index –i, which means “all indices except i”. For example, data[-i,] is all the data except for the ith data point. For our data file where the first 10 columns are predictors and the 11th column is the response, data[-i,11] is the response for all but the ith data point, and data[-i,1:10] are the predictors for all but the ith data point.

(There are other, easier ways to get around this problem, but I want you to get practice doing some basic data manipulation and extraction, and maybe some looping too.)

* **Note** that kknn will read the responses as continuous, and return the fraction of the k closest responses that are 1 (rather than the most common response, 1 or 0).

# read data

data <- read.delim2("credit\_card\_data\_headers.txt", header = TRUE, sep = "\t", dec = ",")

# access library kknn

library(kknn)

# Convert data into a scaled numerical matrix

#dataNumericalMatrix <- scale(data.matrix(data[,1:10]), center = TRUE, scale = TRUE)

#Setting seed to produce reproducible results

set.seed(1)

# create a function knnModel and pass the number K

knnModel = function(K){

# predictions: start with a vector of all zeros

predicted <- rep(0,(nrow(data)))

# Loop for all data points 1 --> total number of rows in dataset

for (i in 1:nrow(data)){

# data[-i] means we remove row i of the data when finding nearest neighbors...

#...otherwise, it'll be its own nearest neighbor!

model = kknn(R1~.,data[-i,],data[i,],k=K, scale = TRUE)

# record whether the prediction is at least 0.5 (round to 1) or less than 0.5 (round to 0)

predicted[i] <- as.integer(fitted(model)+0.5) # round off to 0 or 1

}

# calculate accuracy of predictions

acc = sum(predicted == data[,11]) / nrow(data)

return(acc)

}

# set up a vector of 30 zeros to start

accurracy=rep(0,30)

# call knnModel for values of k from 1 to 30

for (K in 1:30){

accurracy[K] = knnModel(K)

}

# review accuracies

plot(accurracy)

title("K-Nearest-Neighbors")

1. I know I said I wouldn’t give you exact R code to copy, because I want you to learn for yourself. In general, that’s definitely true – but in this case, because it’s your first R assignment and because the ksvm function leaves you in the middle of a mathematical calculation that we haven’t gotten into in this course, I’m giving you the code. [↑](#footnote-ref-1)