WEEK 1 HOMEWORK – SAMPLE SOLUTIONS

IMPORTANT NOTE

These homework solutions show multiple approaches and some optional extensions for most of the questions in the assignment. You don't need to submit all this in your assignments; they're included here just to help you learn more – because remember, the main goal of the homework assignments, and of the entire course, is to help you learn as much as you can, and develop your analytics skills as much as possible!

Question 1

Describe a situation or problem from your job, everyday life, current events, etc., for which a classification model would be appropriate. List some (up to 5) predictors that you might use.

One possible answer:

Being students at Georgia Tech, the Teaching Assistants for the course suggested the following example. A college admissions officer has a large pool of applicants must decide who will make up the next incoming class. The applicants must be put into different categories – admit, waitlist, and deny – so a classification model is appropriate. Some common factors used in college admissions classification are high school GPA, rank in high school class, SAT and/or ACT score, number of advanced placement courses taken, quality of written essay(s), quality of letters of recommendation, and quantity and depth of extracurricular activities.

If the goal of the model was to automate a process to make decisions that are similar to those made in the past, then previous admit/waitlist/deny decisions could be used as the response. Alternatively, if the goal of the model was to make *better* admissions decisions, then a different

measure could be used as the response – for example, if the goal is to maximize the academic success of students, then each admitted student's college GPA could be the response; if the goal is to maximize the post-graduation success of admitted students, then some measure of career success could be the response; etc.

Question 2

The file credit_card_data.txt contains 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 categorial 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 a_0 and $a_1 \ldots a_m$. 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]

```
# call ksvm. Vanilladot is a simple linear kernel.
model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type="C-
svc",kernel="vanilladot",C=100,scaled=TRUE)
# calculate a1...am
# a <- colSums(data[model@SVindex,1:10] * model@coef[[1]]) # for unscaled data
a <- colSums(data[model@xmatrix[[1]]] * model@coef[[1]]) # for scaled data
```

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)

SOLUTION:

There are multiple possible answers. See file HW1-Q2-1-fall.R for the R code for one answer. Please note that a good solution doesn't have to try both of the possibilities in the code; they're both shown to help you learn, but they're not necessary.

One possible linear classifier you can use, for scaled data z, is

$-0.0010065348z_1 - 0.0011729048z_2 - 0.0016261967z_3 + 0.0030064203z_4 + 1.0049405641z_5 - 0.0028259432z_6 + 0.0002600295z_7 - 0.0005349551z_8 - 0.0012283758z_9 + 0.1063633995z_{10} + 0.08158492 = 0.$

It predicts 565 points (about 86.4%) correctly. (Note that this is its performance on the training data; as you saw in Module 3, that's not a reliable estimate of its true predictive ability.) This quality of linear classifier can be found for a wide range of values of C (from 0.01 to 1000, and beyond). Using unscaled data, it's a lot harder to find a C that does this well.

It's also possible to find a better nonlinear classifier using a different kernel; kudos to those of you who went even deeper and tried this!

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.

- *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).*

SOLUTION:

Here's one possible solution. See file HW1-Q2-2-fall.R for code. Please note that a good solution doesn't have to try all of the possibilities in the code.

As detailed in the code, we observe maximum accuracy for k=12 and k=15. (Again, as above, we're reporting performance on the training data, which is generally not good practice, as you saw in Module 3.) A summary of the number of correct predictions for different values of k is shown below (using scaled data).

Value of k (scaled data)	Correct predictions	Percent correct predictions
1-4	533	81.50%
6	553	84.56%
7,9	554	84.71%
8	555	84.86%
10,19-20	556	85.02%
5,11,13-14,16-18	557	85.17%
12,15	558	85.32%

As the table shows, the key (in the training data) is to use k ≥ 5; smaller values of k are significantly inferior. Although k=12 and k=15 look slightly better than the rest, it's not a statistically significant difference.

- Note that if we're just looking at fraction of correct predictions, it might be easy to get caught up in finding the very highest amount we can find. Don't lose sight of the fact that these differences might just be 1 data point out of 654 – which is not statistically significant.

We could do the same using unscaled data, by changing one word in the R code; replace

model=kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,data[-i,],data[i,],k=X, scale = TRUE)

with

model=kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,data[-i,],data[i,],k=X, scale = FALSE)

Using unscaled data, the results are significantly worse.

Value of k (unscaled data)	**Correct predictions**	**Percent correct predictions**
1-4	434	66.36%
10	443	67.74%
11	445	68.04%
12	447	68.35%
9,13	449	68.65%
14-15	450	68.81%
5,20	452	69.11%
7-8,16-19	453	69.27%
6	455	69.57%