ISYE 6501 - Homework 10

Justin Lewis

March 14, 2019

Start by clearing the environment and importing the relevant packages.

```
rm(list=ls())
library(kernlab, quietly=TRUE)
library(knitr, quietly=TRUE)
library(ggplot2, quietly=TRUE)
```

Question 14.1

Q:

The breast cancer data set has missing values.

1. Use the mean/mode imputation method to impute values for the missing data.
2. Use regression to impute values for the missing data.
3. Use regression with perturbation to impute values for the missing data.
4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using:
    1. the data sets from questions 1,2,3
    2. the data that remains after data points with missing values are removed
    3. the data set when a binary variable is introduced to indicate missing values

A:

Lets begin by importing our data and taking a look at the form.

```
#Import data
bcdata <- data.frame(read.csv('breast-cancer-wisconsin.data.txt'), stringsAsFactors=FALSE)

#Rename columns based on provided attribute info. Messed with to reduce table widths.
colnames(bcdata) <- c('id', 'clump.thickn', 'uni.cl.size', 'uni.cl.shape', 'marg.adhes', 'sing.epith.cl.size', 'bare.nuclei'
, 'bland.c', 'norm.nucl', 'mito', 'class')
kable(bcdata[1:5, ], caption='First 5 Rows of Breast Cancer Data')
```

First 5 Rows of Breast Cancer Data

id	clump.thickn	uni.cl.size	uni.cl.shape	marg.adhes	sing.epith.cl.size	bare.nuclei	bland.c	norm.nucl	mito	class
1002945	5	4	4	5	7	10	3	2	1	2
1015425	3	1	1	1	2	2	3	1	1	2
1016277	6	8	8	1	3	4	3	7	1	2
1017023	4	1	1	3	2	1	3	1	1	2
1017122	8	10	10	8	7	10	9	7	1	4

```
#Make copies to use for parts 2 and 3.
bc2data <- cbind(bcdata)
bc3data <- cbind(bcdata)

#Store the column names
cols <- colnames(bcdata)
```

So we have our data and have the table cleaned up with more informative column names. Now we can look into how much missing data is in each column. Going to check NA, NULL, and 'other' values.

```
#Check for any NA data in the dataframe
print(bcdata[is.na(bcdata)==TRUE])
```

```
## character(0)
```

```
#Check for NULLs
lapply(bcdata, is.null)
```

```
## $id
## [1] FALSE
##
## $clump.thickn
## [1] FALSE
##
## $uni.cl.size
## [1] FALSE
##
## $uni.cl.shape
## [1] FALSE
##
## $marg.adhes
## [1] FALSE
##
## $sing.epith.cl.size
## [1] FALSE
##
## $bare.nuclei
## [1] FALSE
##
## $bland.c
## [1] FALSE
##
## $norm.nucl
## [1] FALSE
##
## $mito
## [1] FALSE
##
## $class
## [1] FALSE
```

```
#This shows us the summary for each column
summary(bcdata)
```

```
##        id            clump.thickn     uni.cl.size      uni.cl.shape
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870258   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071807   Mean   : 4.417   Mean   : 3.138   Mean   : 3.211
##  3rd Qu.: 1238354   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##    marg.adhes     sing.epith.cl.size  bare.nuclei     bland.c
##  Min.   : 1.000   Min.   : 1.000     1      :401   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000     10     :132   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000     2      : 30   Median : 3.000
##  Mean   : 2.809   Mean   : 3.218     5      : 30   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000     3      : 28   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000     8      : 21   Max.   :10.000
##                                      (Other): 56
##    norm.nucl          mito           class
##  Min.   : 1.00   Min.   : 1.00   Min.   :2.000
##  1st Qu.: 1.00   1st Qu.: 1.00   1st Qu.:2.000
##  Median : 1.00   Median : 1.00   Median :2.000
##  Mean   : 2.87   Mean   : 1.59   Mean   :2.691
##  3rd Qu.: 4.00   3rd Qu.: 1.00   3rd Qu.:4.000
##  Max.   :10.00   Max.   :10.00   Max.   :4.000
##
```

```
#Since 'bare.nuclei' has 'other' listed, lets see what else there is in the column
unique(bcdata$bare.nuclei)
```

```
##  [1] 10 2  4  1  3  9  7  ?  5  8  6
## Levels: ? 1 10 2 3 4 5 6 7 8 9
```

Based on this it looks like only the 'bare.nuclei' column has any 'missing' values. 'Missing' because instead of NA or Null, we have question marks in some rows. From what I can find here these look like the only missing values. Lets look at these rows specifically.

```
missing <- subset(bcdata, bare.nuclei == '?')
missing_rows <- rownames(missing)
kable(missing, caption='Subset of BCData with Missing Values')
```

Subset of BCData with Missing Values

	id	clump.thickn	uni.cl.size	uni.cl.shape	marg.adhes	sing.epith.cl.size	bare.nuclei	bland.c	norm.nucl	mito	class
23	1057013	8	4	5	1	2	?	7	3	1	4
40	1096800	6	6	6	9	6	?	7	8	1	2
139	1183246	1	1	1	1	1	?	2	1	1	2
145	1184840	1	1	3	1	2	?	2	1	1	2
158	1193683	1	1	2	1	3	?	1	1	1	2
164	1197510	5	1	1	1	2	?	3	1	1	2
235	1241232	3	1	4	1	2	?	3	1	1	2
249	169356	3	1	1	1	2	?	3	1	1	2
275	432809	3	1	3	1	2	?	2	1	1	2
292	563649	8	8	8	1	2	?	6	10	1	4
294	606140	1	1	1	1	2	?	2	1	1	2
297	61634	5	4	3	1	2	?	2	3	1	2
315	704168	4	6	5	6	7	?	4	9	1	2
321	733639	3	1	1	1	2	?	3	1	1	2

	id	clump.thickn	uni.cl.size	uni.cl.shape	marg.adhes	sing.epith.cl.size	bare.nuclei	bland.c	norm.nucl	mito	class
411	1238464	1	1	1	1	1	?	2	1	1	2
617	1057067	1	1	1	1	1	?	1	1	1	2

```
#Create the mean value
bnmean <- round(mean(as.numeric(bcdata[['bare.nuclei']])), 4)

#Since there was missing data, bare.nuclei column type is a factor. Need to use levels.
levels(bcdata$bare.nuclei)[levels(bcdata$bare.nuclei)=='?'] <- as.character(bnmean)

#Now check the rows that previously had missing values
kable(bcdata[missing_rows, ], caption='Previously Missing Rows of BCData')
```

Previously Missing Rows of BCData

	id	clump.thickn	uni.cl.size	uni.cl.shape	marg.adhes	sing.epith.cl.size	bare.nuclei	bland.c	norm.nucl	mito	class
23	1057013	8	4	5	1	2	3.1676	7	3	1	4
40	1096800	6	6	6	9	6	3.1676	7	8	1	2
139	1183246	1	1	1	1	1	3.1676	2	1	1	2
145	1184840	1	1	3	1	2	3.1676	2	1	1	2
158	1193683	1	1	2	1	3	3.1676	1	1	1	2
164	1197510	5	1	1	1	2	3.1676	3	1	1	2
235	1241232	3	1	4	1	2	3.1676	3	1	1	2
249	169356	3	1	1	1	2	3.1676	3	1	1	2
275	432809	3	1	3	1	2	3.1676	2	1	1	2
292	563649	8	8	8	1	2	3.1676	6	10	1	4
294	606140	1	1	1	1	2	3.1676	2	1	1	2
297	61634	5	4	3	1	2	3.1676	2	3	1	2
315	704168	4	6	5	6	7	3.1676	4	9	1	2
321	733639	3	1	1	1	2	3.1676	3	1	1	2
411	1238464	1	1	1	1	1	3.1676	2	1	1	2
617	1057067	1	1	1	1	1	3.1676	1	1	1	2

Looks like we have finished filling in the ‘missing’ data using mean/mode imputation.

Moving on to the regression portion, I am going to cheat a bit by reusing the missing_rows to make it a bit easier to split the data into what is basically training data, and data to be predicted on.

```
#Define what will be our predictors, and the target. Basically using all columns other than bare.nuclei to predict the value
in bare.nuclei.
predictors <- bc2data[, -7]
target <- bc2data[, 7]

#Split the data up
train_X <- predictors[-c(as.numeric(missing_rows)),]
test_X <- predictors[c(as.numeric(missing_rows)), ]
train_y <- target[-c(as.numeric(missing_rows))]

#We don't really need this since we will not be comparing the predictions to anything
test_y <- target[-c(as.numeric(missing_rows))]

#Make sure there are no question marks in train_y
'?' %in% train_y
```

```
## [1] FALSE
```

```
#Now need to convert the training y-values from factors to something interpretable by the model
train_y <- as.character(levels(train_y))[train_y]

#With our training data ready, we can make a regression model
imputelm <- lm(train_y~., data=train_X)

#And predict the imputations
imputedvalues <- predict.lm(imputelm, newdata=test_X)

print(imputedvalues)
```

```
##       23       40      139      145      158      164      235      249
## 7.192749 3.421250 1.189314 1.579971 1.260317 1.431066 1.944819 1.564262
##      275      292      294      297      315      321      411      617
## 1.741927 6.432896 1.303737 1.187606 2.083916 1.470705 1.180159 1.059397
```

We can see these values are significantly different than the mean value used in the first portion. Now want to fill in the indices in bc2data with the new values.

```
#Convert to a matrix to simplify assigning new values in specific locations
bc2mat <- as.matrix(bc2data)

#Go through each value and assign the predicted value at the location
for (ind in seq_along(as.numeric(missing_rows))){
  bc2mat[as.numeric(missing_rows[ind]),'bare.nuclei'] <- as.numeric(imputedvalues[[ind]])
}
#Convert back to a dataframe
bc2data <- data.frame(bc2mat)

#Now check the rows that previously had missing values
kable(bc2data[missing_rows, ], caption='Previously Missing Rows of BCData - Filled with Regression')
```

Previously Missing Rows of BCData - Filled with Regression

	id	clump.thickn	uni.cl.size	uni.cl.shape	marg.adhes	sing.epith.cl.size	bare.nuclei	bland.c	norm.nucl	mito	class
23	1057013	8	4	5	1	2	7.19274857477521	7	3	1	4
40	1096800	6	6	6	9	6	3.42124950766359	7	8	1	2
139	1183246	1	1	1	1	1	1.18931379687537	2	1	1	2
145	1184840	1	1	3	1	2	1.57997116347566	2	1	1	2
158	1193683	1	1	2	1	3	1.26031691014478	1	1	1	2
164	1197510	5	1	1	1	2	1.43106582093328	3	1	1	2
235	1241232	3	1	4	1	2	1.94481947383316	3	1	1	2
249	169356	3	1	1	1	2	1.56426216863242	3	1	1	2
275	432809	3	1	3	1	2	1.74192720991471	2	1	1	2
292	563649	8	8	8	1	2	6.43289571447995	6	10	1	4
294	606140	1	1	1	1	2	1.30373690009053	2	1	1	2
297	61634	5	4	3	1	2	1.18760582921301	2	3	1	2
315	704168	4	6	5	6	7	2.0839164991736	4	9	1	2
321	733639	3	1	1	1	2	1.47070476958598	3	1	1	2
411	1238464	1	1	1	1	1	1.18015872316139	2	1	1	2
617	1057067	1	1	1	1	1	1.05939734240347	1	1	1	2

With this table we see that the previously ‘missing’ values have been filled in with our predicted values from regression.

For regression with perturbation we have a lot of options to choose from, and we can just build on top of the previous section. We can take the predicted values and add some perturbation to them before we replace the missing values. Maybe we want to produce some random value between +10% of the value and -10% of the value. We cannot just use the same range for every value, because for larger numbers we will end up with a very small perturbation and for smaller values we end up changing the number completely. The +/-10% is an arbitrary limit, realistically it could be much more or much less.

```
#Loop through the predicted values and add a random value in the range of -10% and +10% of the current value.
perturbedimputed <- list()
for (i in seq_along(imputedvalues)){
  val <- imputedvalues[i]
  pertval <- val + runif(1, min=-0.1*val, max=0.1*val)
  perturbedimputed[i] <- pertval
}

#Now place these new values into bc3data to replace the missing values
bc3mat <- as.matrix(bc3data)
for (ind in seq_along(as.numeric(missing_rows))){
  bc3mat[as.numeric(missing_rows[ind]),'bare.nuclei'] <- as.numeric(perturbedimputed[[ind]])
}
bc3data <- data.frame(bc3mat)

#And confirm they've been replaced
kable(bc3data[missing_rows, ], caption='Previously Missing Rows of BCData - Filled with Regression and Perturbation')
```

Previously Missing Rows of BCData - Filled with Regression and Perturbation

	id	clump.thickn	uni.cl.size	uni.cl.shape	marg.adhes	sing.epith.cl.size	bare.nuclei	bland.c	norm.nucl	mito	class
23	1057013	8	4	5	1	2	7.29309263342474	7	3	1	4
40	1096800	6	6	6	9	6	3.64918795776647	7	8	1	2
139	1183246	1	1	1	1	1	1.07797042918553	2	1	1	2
145	1184840	1	1	3	1	2	1.72321995682558	2	1	1	2
158	1193683	1	1	2	1	3	1.30377735344928	1	1	1	2
164	1197510	5	1	1	1	2	1.35868636333324	3	1	1	2
235	1241232	3	1	4	1	2	1.80798166300757	3	1	1	2
249	169356	3	1	1	1	2	1.63096650496509	3	1	1	2
275	432809	3	1	3	1	2	1.86874967620596	2	1	1	2
292	563649	8	8	8	1	2	6.42966645262823	6	10	1	4
294	606140	1	1	1	1	2	1.18885662350754	2	1	1	2
297	61634	5	4	3	1	2	1.19085699098422	2	3	1	2
315	704168	4	6	5	6	7	1.92919167188235	4	9	1	2
321	733639	3	1	1	1	2	1.40822628013701	3	1	1	2
411	1238464	1	1	1	1	1	1.22338136505733	2	1	1	2
617	1057067	1	1	1	1	1	0.965234017100669	1	1	1	2

We can compare the differences between the regression values and the regression+perturbation values as well, just to see how much of an adjustment was made.

```
compdf <- data.frame(imputedvalues)
compdf$perturbedvalues <- perturbedimputed
kable(compdf, caption='Imputed vs Perturbed Imputed Values')
```

Imputed vs Perturbed Imputed Values

	imputedvalues	perturbedvalues
23	7.192749	7.29309263342474

	imputedvalues	perturbedvalues
40	3.421250	3.64918795776647
139	1.189314	1.07797042918553
145	1.579971	1.72321995682558
158	1.260317	1.30377735344928
164	1.431066	1.35868636333324
235	1.944819	1.80798166300757
249	1.564262	1.63096650496509
275	1.741927	1.86874967620596
292	6.432896	6.42966645262823
294	1.303737	1.18885662350754
297	1.187606	1.19085699098422
315	2.083916	1.92919167188235
321	1.470705	1.40822628013701
411	1.180159	1.22338136505733
617	1.059397	0.965234017100669

So while there is not a huge difference in this set, by adding the perturbation we ensure that if there was more missing data with the same set of predictor values we would not predict the exact same value. Effectively adding a bit of ‘jitter’ to the missing value predictions.

Question 15.1

Q:

Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

A:

At work I deal with an accelerator beamline. Tuning the beamline is an optimization process. One needs to know the variables (beamline element settings, beam mass/charge ratio, beam energy), the constraints (keep them close to theory values, some cannot be negative, cannot be more than the maximum slider setting), and the cost function (measuring beam transmission as well as how close to an ideal gaussian beam profile we have at a specific location as read by a position monitor).