Module 8: Algorithms I
Tutorial on Basic Regression and Classification

Applied Data Science
Summer 2017

Introduction

This is a written explanation of the R script that is used for this module's tutorial on regression and classification in R. The document contains five sections: data loading, exploration and setup; linear modeling; k-means clustering; k-nn classification; and decision trees.

The dataset used for this tutorial can be found here at the UCI ML archive or on Canvas as the auto-mpg dataset.

Loading, exploration and setup

The following block of code sets up our data. First, we use `setwd` to set our working directory. We then read our data into a variable, I've named it `auto` in this instance. Next, we do some data exploration quickly to check what data we're working with. Horsepower is read in as a factor instead of an integer, which is wrong, so we'll correct that on the next line. Then, we'll set up a function `fe` that creates a class `fuelEfficient`, for which the vehicle can either fall into or not fall into.

```
# NOTE: The output for this code block has been hidden.
setwd("./")
auto <- read.table("auto-mpg.data",
                   col.names = c("MPG","cylinders","displacement",
                                 "horsepower","weight","acceleration",
                                 "year","origin","name"))
str(auto)
summary(auto)
auto$horsepower <- as.integer(auto$horsepower)

fe <- function(x){if(x>30){T}else{F}}
auto$fuelEfficient<-sapply(X=with(auto, MPG),FUN=fe)
```

After we've got our data in good shape, we'll split it into training and testing data sets. This will be useful to validate our supervised algorithms. We set a random seed (equal to the date in this case, it can be your favorite number or anything you'd like) to ensure others can repeat our work. We then get the sample size for our train set. In this case, I've decided that it will be 80-percent train and 20-percent test. We then sample the rows we're interested in and subset auto to contain all the columns of the rows we interested in (using the positive case for training and negative for testing.) Lastly, we validate that our output is as desired.

```
#NOTE: All the output for this code block is hidden

set.seed(31052017)
```

```r
#Select sample size and sample rows
smp_size <- floor(.8 * nrow(auto))
ti <- sample(seq_len(nrow(auto)),size=smp_size)

#Create train and test version by using positive and negative samples from above
train.auto <- auto[ti,]
test.auto <- auto[-ti,]

#Validate all is well
str(train.auto)
str(test.auto)
```

Linear modeling

In this section we use the `lm` function to create a linear model, or perform a linear regression of, our data. We attempt to predict the miles per gallon of each automobile by considering the weight, the year it was released, and the horsepower of the vehicle. We note from the results that the horsepower is not a reliable instrument in this case.

To do this, we pass texttttMPG weight+year+horsepower to the `lm` function. This represents the linear model we're suggesting. The `summary` function gives us a nice summary of our model. We can use `predict.lm` along with our model and the test data to test our model on new data. The last line of this code block summarizes the differences between the predicted value and our results.

We can see that on average, we missed the true MPG of the vehical by about 2.8 miles per gallon, and for the average vehicle, we missed by about 2.2 miles per gallon.

You'll note the entire regression analysis only requires four lines of (very abstract) code.

```r
#All the results for this code block appear at the end
mylm <- lm(MPG~weight+year+horsepower,dat=train.auto)
summary(mylm)
res <- predict.lm(mylm,test.auto)
summary(abs(res - test.auto$MPG))
```

```
##
## Call:
## lm(formula = MPG ~ weight + year + horsepower, data = train.auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1167 -2.4135 -0.1326  1.9467 14.3513
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.460e+01  4.330e+00  -3.372 0.000839 ***
## weight      -6.605e-03  2.758e-04 -23.950  < 2e-16 ***
## year         7.526e-01  5.209e-02  14.448  < 2e-16 ***
## horsepower   9.520e-03  7.244e-03   1.314 0.189741
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 3.328 on 314 degrees of freedom
## Multiple R-squared:  0.8176,Adjusted R-squared:  0.8158
## F-statistic:   469 on 3 and 314 DF,  p-value: < 2.2e-16
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1365  1.2010  2.2190  2.8430  3.7800 12.7400
```

K-Means clustering

In this section, we use K-means clustering to attempt to automatically classify our data. We use model year and weight again, as they were relatively successful with our regression model. We find that they are less successful here; however, we are able to capture all of the fuel efficient vehicles into a single cluster, suggesting there is a cutoff point, in terms of year and weight, that a vehicle must pass in order to be fuel efficient.

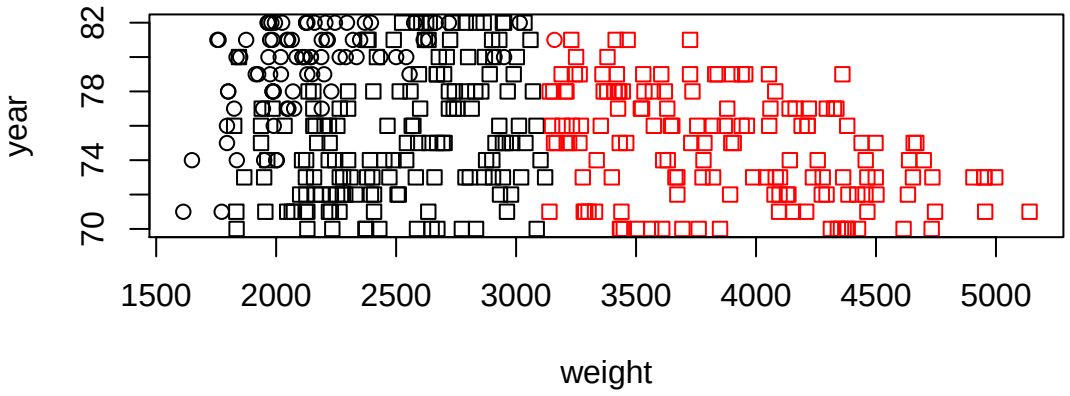
To do this, we use the `kmeans` fucnction on the columns of the data we wish to analyze, and pass as an integer the number of centers we would like. Optionally, we can pick the number of iterations or the algorithm used. To assess the effectiveness of this approach, I chose to use a table, comparing the cluster to the true fuel efficency status, as well as a color and shape coded scatter plot. The color indicates the class (black is 1, red is 2) and the shape indicates the fuel efficiency (square is false, circle is true).

From the table, we can see that the model was poor at distinguishing between non-fuel efficient vehicles, but had a very high recall of fuel efficient vehicles.

```
## Results for this section appear after the plot
mykm <- kmeans(auto[c("weight","year")],2)
mykm.df <- data.frame(auto,mykm$cluster)
with(mykm.df, table(fuelEfficient,mykm.cluster))

##              mykm.cluster
## fuelEfficient   1   2
##         FALSE 158 155
##         TRUE   84   1

plot(auto[c("weight","year")],col=mykm$cluster,pch=as.integer(auto$fuelEfficient))
```



K Nearest Neighbor

In this section, we use a K Nearest Neighbor method to evaluate our dataset. Here we use all the features again, instead of limiting ourselves to just weight and year.

To do this we load the class library with `library('class')` and use the `knn` function on our train and test data. We set the `cl` parameter equal to the class of interest and set the `k` parameter equal to the number of neighbors we would like to "vote". Below, you can see how this is done for k=1 and k=3.

We compare our results by appending the output of these functions to our dataframe and constructing a table of results.

We can see that this method is much better at predicting false cases (non-fuel efficient vehicles) than K-means; however, has only about 50 percent recall and precision for fuel efficient vehicles.

```
#The output in this code block is displayed after each command.
# K-NN for 1 and 3 neighbors
library('class')
k1.pred <- knn(train.auto[c(seq(2,8),10)],test.auto[c(seq(2,8),10)],cl=train.auto$fuelEfficient,k=1)
k3.pred <- knn(train.auto[c(seq(2,8),10)],test.auto[c(seq(2,8),10)],cl=train.auto$fuelEfficient,k=3)
kpred <- data.frame(test.auto,k1.pred,k3.pred)
with(kpred, table(fuelEfficient, k1.pred))

##              k1.pred
## fuelEfficient FALSE TRUE
##         FALSE    53   11
##         TRUE      8    8

with(kpred, table(fuelEfficient, k3.pred))

##              k3.pred
## fuelEfficient FALSE TRUE
##         FALSE    58    6
```

```
##         TRUE      9    7
```

Decision Trees

Finally, in this section, we demonstrate decision trees. We demonstrate two methods of decision trees: regression trees and classification trees. We use the R function `rpart` to achieve this. `rpart` stands for recursive partitioning, which is a reasonable description of the decision tree creation process.

Starting with the classification tree, to build this tree we pass the rpart function the model we want to use (similar to the regression above), here I chose to use `fuelEfficient  cylinders+horsepower+weight+acceleration+year` We also specify the data for our model and the method of partitioning, class based or anova. For classification, we choose `class`. We then use the `predict` function on the tree and our test data and create a new data frame that has these results as a column. We can find the validity of our model by summarizing the results in the new columns, filtering on the true class.

From this, we can see that on average, our fuel efficient vehicles are being sorted down into branches that are 60 percent true; while our non-fuel efficient vehicles are being sorted into branches that are 99 percent false. Again, it appears that finding fuel efficient vehicles is a much harder task.

```
# Classification Tree
library('rpart')
ctree <- rpart(fuelEfficient ~ cylinders+horsepower+weight+acceleration+year,
               data=train.auto,
               method = 'class')
ctree.test <- data.frame(test.auto,predict(ctree,test.auto))
summary(ctree.test[which(ctree.test$fuelEfficient==T),c(10,11,12)])

##  fuelEfficient      FALSE.            TRUE.
##  Mode:logical   Min.   :0.08163   Min.   :0.1613
##  TRUE:16        1st Qu.:0.08163   1st Qu.:0.4778
##  NA's:0         Median :0.33333   Median :0.6667
##                 Mean   :0.36518   Mean   :0.6348
##                 3rd Qu.:0.52218   3rd Qu.:0.9184
##                 Max.   :0.83871   Max.   :0.9184

summary(ctree.test[which(ctree.test$fuelEfficient==F),c(10,11,12)])

##  fuelEfficient       FALSE.            TRUE.
##  Mode :logical   Min.   :0.08163   Min.   :0.00000
##  FALSE:64        1st Qu.:0.98980   1st Qu.:0.01020
##  NA's :0         Median :0.98980   Median :0.01020
##                  Mean   :0.91915   Mean   :0.08085
##                  3rd Qu.:0.98980   3rd Qu.:0.01020
##                  Max.   :1.00000   Max.   :0.91837
```

We can also use a regression tree to predict the fuel efficiency of the vehicle. This uses a similar process, except we provide `anova` as the parameter for method and we compare our results not by adding a column to our data frame and constructing tables, but by summarzing the difference between the predicted MPG and the true MPG.

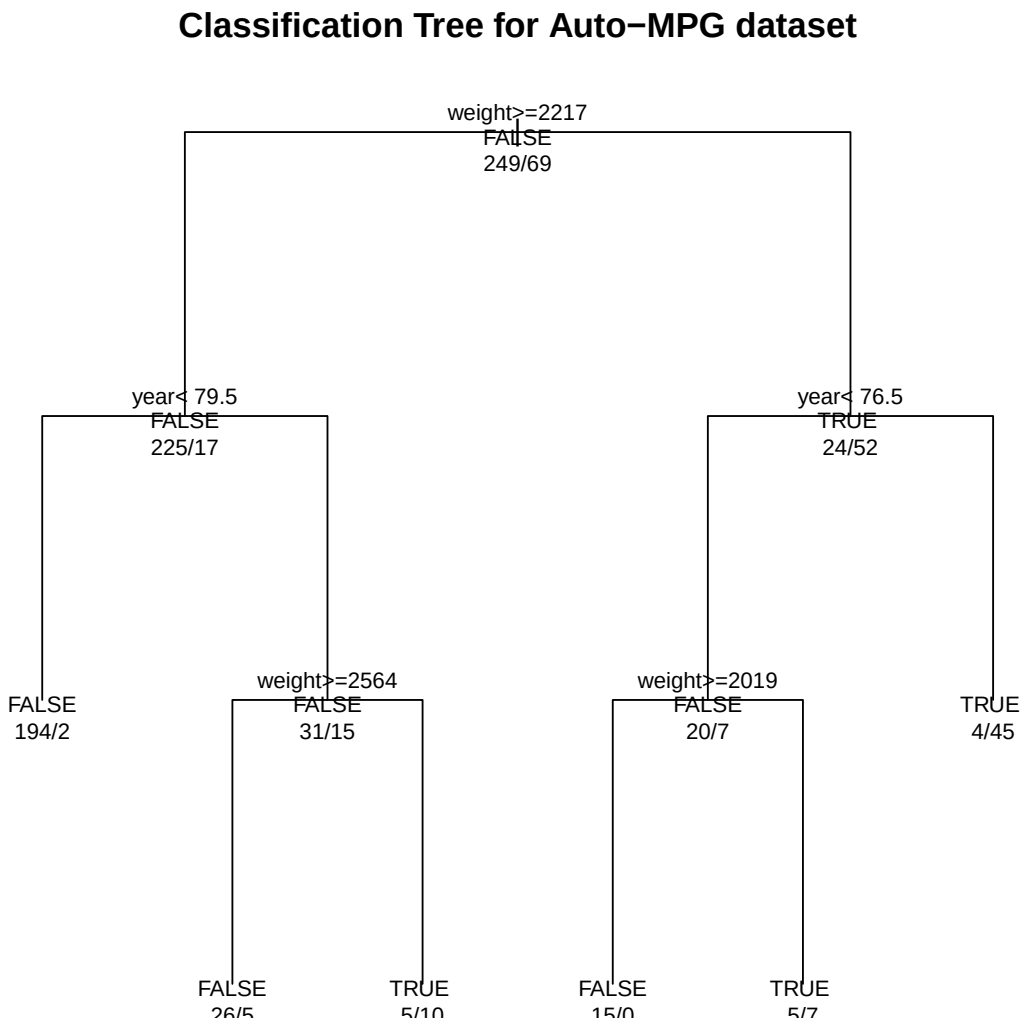
Our regression tree, you'll note, performs about as well as our linear regression; perhaps slightly better.

```r
# Regression Tree
rtree <- rpart(MPG ~ cylinders+horsepower+weight+acceleration+year,
               data=train.auto,
               method = 'anova')
rtree.test <- data.frame(test.auto,predict(rtree,test.auto))
summary(abs(rtree.test[1]-rtree.test[11]))
```

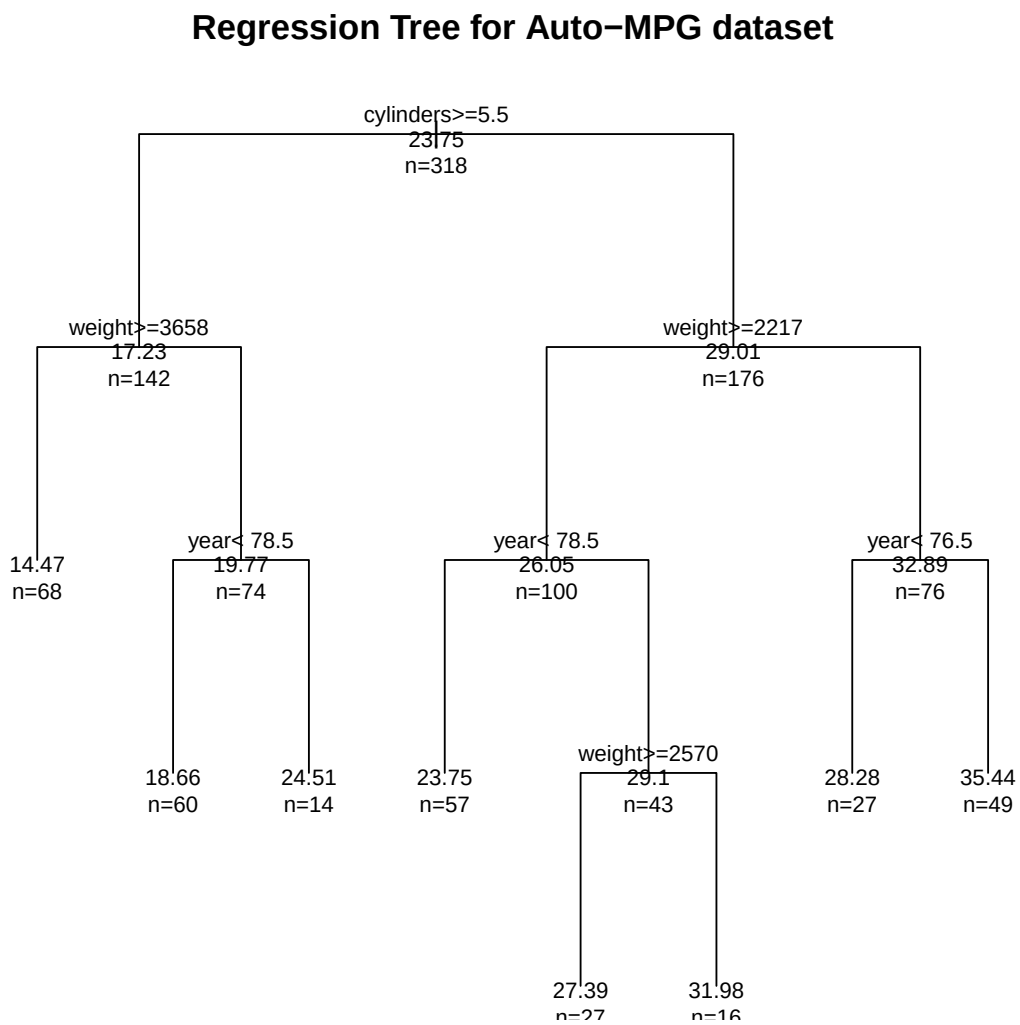
```
##       MPG
##  Min.   : 0.1509
##  1st Qu.: 1.0294
##  Median : 2.2509
##  Mean   : 2.6045
##  3rd Qu.: 3.4706
##  Max.   :12.1294
```

Further, we can use the standard `plot` function to plot of decision trees. We must add the text on after with the `text` function. Unfortunately, there is no easy way to plot the results of your prediction with the standard rpart library.

```
plot(ctree, uniform=TRUE, main="Classification Tree for Auto-MPG dataset")
text(ctree, use.n=TRUE, all=TRUE, cex=.75)
```



```
plot(rtree, uniform=TRUE, main="Regression Tree for Auto-MPG dataset")
text(rtree, use.n=TRUE, all=TRUE, cex=.75)
```



Conclusion

This tutorial demonstrated how to use linear models, k-means, k-nearest neighbor, and decision tree algorithms in R for classification and regression. You should be able to modify the code provided here to use these functions for any purpose. If you have questions about how to adopt this code for the homework, please use the Piazza discussion board and ask your classmates for help.

Other great resources are the documentation for the specific packages and Stackoverflow.