Weight Lifting Class - Prediction Model (Coursera - Practical Machine Learning)

Mike Kuklinski

February 11, 2016

Introduction and Description of Data

This report deals with the data sets collected for the **Human Activity Recognition** project, more specifically, the **Weight Lifting Exercises Dataset**. (more information on this generous source is at the bottom). The Weight Lifting Exercise data set was generated by placing various instruments (accelerometers, gyroscopes, etc.) on the forearm, arm, belt, and dumbbell of 6 participants while they performed the same exercise both correctly (A class) and incorrectly (B-E classes). See description of exercise below:

One set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:
- Class A - exactly according to the specification
- Class B - throwing the elbows to the front
- Class C - lifting the dumbbell only halfway
- Class D - lowering the dumbbell only halfway
- Class E - and throwing the hips to the front

Additional Notes:

- Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate.
- The 6 participants were male, aged between 20-28 years, with little weight lifting experience.
- A light dumbbell (1.25kg) was used to ensure all participants could perform the exercises easily and safely.

The purpose of the research was to see if the manner (class) in which the exercise was performed could be determined purely by the readings on the instruments given a single point in time.

The purpose of this report is to walk through the steps I took in creating a model for predicting the class in which an exercise was performed. The training set received consisted of 19,622 observations and 160 variables. The test set had the same variables, but 20 observations. In the end, I used Principal Component Analysis and created a random forest model, utilizing 4 k-folds cross validation and 500 trees. Below is the process I took for building the model.

Loading and Cleaning Data

To start I first imported the necessary libraries, saved the training data set to my working directory, and loaded.

```
# import necessary packages
suppressMessages(suppressWarnings(library(caret)))
suppressMessages(suppressWarnings(library(ggplot2)))

# create folder to store and work with data
setwd('~/R Scripts/Coursera/Practical Machine Learning/Course Project/Weight_Lifting_Class_Prediction_M
if (!dir.exists('rawdata')){
```

```
    dir.create('rawdata')
    url <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
    download.file(url,destfile='./rawdata/TrainExer.csv')
    }

Tdata <- read.csv('./rawdata/TrainExer.csv')
str(Tdata)
```

Taking a glimpse at the data table, I noticed there were many 'NA' values, missing values, and '#DIV/0' values. These values dealt with variables which appeared to be summary statistics for a specific time interval and did not exist for the majority of the observations. To get a better idea of how important they were, I decided to do a nearZeroCheck.

```
zero_check <- nearZeroVar(Tdata, saveMetrics = TRUE)
head(subset(zero_check, nzv == T), 15)
```

```
##                        freqRatio percentUnique zeroVar  nzv
## new_window              47.33005    0.01019264   FALSE TRUE
## kurtosis_roll_belt    1921.60000    2.02323922   FALSE TRUE
## kurtosis_picth_belt    600.50000    1.61553358   FALSE TRUE
## kurtosis_yaw_belt       47.33005    0.01019264   FALSE TRUE
## skewness_roll_belt    2135.11111    2.01304658   FALSE TRUE
## skewness_roll_belt.1   600.50000    1.72255631   FALSE TRUE
## skewness_yaw_belt       47.33005    0.01019264   FALSE TRUE
## max_yaw_belt           640.53333    0.34654979   FALSE TRUE
## min_yaw_belt           640.53333    0.34654979   FALSE TRUE
## amplitude_yaw_belt      50.04167    0.02038528   FALSE TRUE
## avg_roll_arm            77.00000    1.68178575   FALSE TRUE
## stddev_roll_arm         77.00000    1.68178575   FALSE TRUE
## var_roll_arm            77.00000    1.68178575   FALSE TRUE
## avg_pitch_arm           77.00000    1.68178575   FALSE TRUE
## stddev_pitch_arm        77.00000    1.68178575   FALSE TRUE
```

The check deemed 60 variables to be unimportant. Examining the type of variables that were identified, they were all primarily summary statistic variables such as average (ave), standard deviation (stddev), min and max, etc. Therefore, I decided to disregard all summary variables in the dataset. In addition, I made the assumption that the date/time information should have little impact on the measurements from the instruments so I disregarded those as well. Finally, given that the model was intended to make predictions based solely on instrument readings, I removed the participant identifier as well.

```
# Identify columns containing summary statistics
kurt <- grep('^kurtosis',names(Tdata), value = F)
skew <- grep('^skewness',names(Tdata), value = F)
maxi <- grep('^max',names(Tdata), value = F)
mini <- grep('^min',names(Tdata), value = F)
amp <- grep('^amplitude',names(Tdata), value = F)
vari <- grep('^var',names(Tdata), value = F)
avg <- grep('^avg',names(Tdata), value = F)
std <- grep('^stddev',names(Tdata), value = F)
time <- c(1, 2, 3, 4, 5, 6, 7)
remove_index<-sort(c(time, kurt, skew, maxi, mini, amp, vari, avg, std))
```

```
# Remove columns from data and delete temp variables
subTdata <- Tdata[,-remove_index]
remove(time, kurt, skew, maxi, mini, amp, vari, avg, std)
dim(subTdata)
```

```
## [1] 19622    53
```

In order to continue with my analysis, I converted all the non-factor variables to a numeric type, in lieu integer, for processing.

```
integers <- sapply(subTdata, is.integer)
tonum <- which(integers == T)

for(i in seq_along(tonum)){
    subTdata[,tonum[i]] <- as.numeric(subTdata[,tonum[i]])
}
```

Pre-Processing

My next step was to check for correlation between variables, since having too many correlated variables would increase the variance. To do this I calculated the correlation matrix and found that 38 of the 53 variables had a correlation of over 0.8 with another variable.

```
# Check correlation between variables
cor_subTdata <- abs(cor(subTdata[,-53]))
diag(cor_subTdata) <- 0
highcor <- which(cor_subTdata > .8, arr.ind = T)
nrow(highcor)
```

```
## [1] 38
```

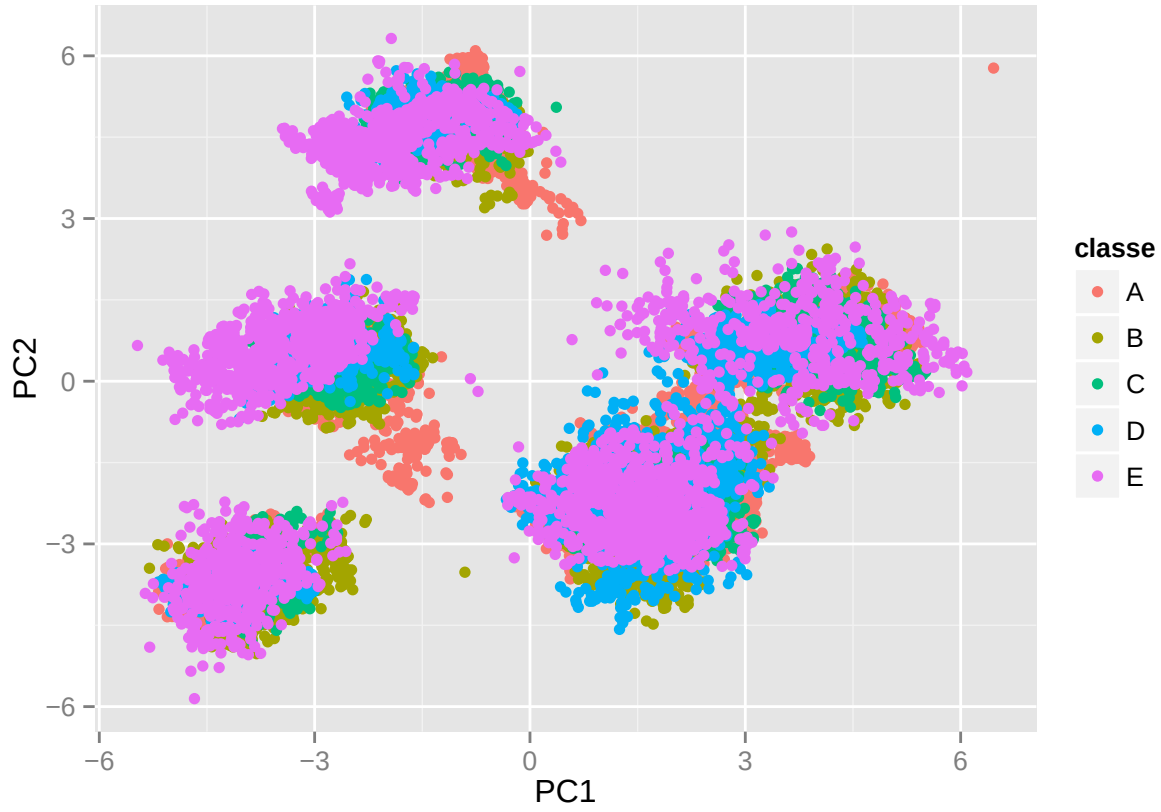
To adjust for this, I ran a *Principal Component Analysis* on the numeric data and found that 99% of the variation could be explained in only 36 variables. I adjusted accordingly.

```
# Perform PCA with 95% variance
pp <- preProcess(subTdata[,-53], method = 'pca', thresh = 0.99)
pp_subTdata <- predict(pp, subTdata[,-53])
pp_subTdata <- cbind(pp_subTdata, classe = subTdata$classe)
pp
```

```
##
## Call:
## preProcess.default(x = subTdata[, -53], method = "pca", thresh = 0.99)
##
## Created from 19622 samples and 52 variables
## Pre-processing: principal component signal extraction, scaled, centered
##
## PCA needed 36 components to capture 99 percent of the variance
```

Below is a look at some of the PCA variables.

```
g2 <- ggplot(data = pp_subTdata, aes(x = PC1, y = PC2, col = classe))
g2 <- g2 + geom_point()
g2
```



Cross Validation and Model Fitting

With my data cleaned and pre-processed, I moved forward with strategizing my model. Given that the variable I was trying to predict was a factor/class and not continuous, I concluded linear models would not be the best model to target. Instead, I started experimenting with CART (classification and regression trees) models, specifically regression trees (rpart), random forest(rf), and boosting(gbm).

Since running a model on such a large dataset takes a long time, I decided to take a sub sample of the data and run some test models to get an idea of which model to target.

```
# Subset Data
set.seed(45)
subsub <- sample(1:nrow(pp_subTdata), 1000)
T1 <- pp_subTdata[subsub,]
```

Due to the amount of time required to re-run the model, I have just listed the code used to create it.

```
# Regression trees
Model_rpart <- train(classe ~ . , data = T1, method = 'rpart')
Model_rf <- train(classe ~ . , data = T1, method = 'rf')
Model_gbm <- train(classe ~ . , data = T1, method = 'gbm')
```

```
Model_rf <- readRDS("./Models/Model_rf.rds")
Model_rf
```

```
## Random Forest
##
## 1000 samples
##   36 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.7770729  0.7151111  0.02286672   0.02839696
##   19    0.7601744  0.6938355  0.02417802   0.03017773
##   36    0.7406988  0.6690209  0.02411395   0.03002325
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

With default settings, random forest was the best choice with an initial accuracy of 77.7%. Regression Trees and boosting yielded accuracies of 38.8% and 73.5% respectively.

With a random forest model target, I began experimenting with the different cross validation methods, number of folds, and number of trees. For CV, I looked at cross validation and leave one out. After various trials of running different combinations of parameters, I settled on a cross validation with k = 4 folds, and 1000 trees since they yielded the best values in accuracy and error for my subset. I then applied these model parameters to the entire data set to get my final model.

Due to the amount of time required to re-run the model, I have just listed the code used to create it.

```
# Run Final Model
modelTrain <- trainControl(method = 'cv', number = 4)
best_model <- train(classe ~ . , data = pp_subTdata, method = 'rf', ntree = 1000, trControl = modelTrai

saveRDS(best_model, "Models/bestmodel.rds")
```

Summary Results

Below is the loaded final model with results.

```
best_model <- readRDS("Models/bestmodel.rds")
best_model
```

```
## Random Forest
##
## 19622 samples
##    36 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 14717, 14716, 14716, 14717
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9817551  0.9769139  0.001980461  0.002508378
##   19    0.9778817  0.9720143  0.003349860  0.004238615
##   36    0.9723266  0.9649866  0.003999118  0.005053411
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Reviewing the final model created, the expected 'out of sample'/'out of bag'(OOB) error was 1.39%. After applying the same clean up and pre-processing procedures on the test data set, this model successfully predicted 20/20 classes in the test data set for the project.

Data Source: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Collaborators:

- Wallace Ugulino (wugulino at inf dot puc-rio dot br)
- Eduardo Velloso
- Hugo Fuks

Read more: http://groupware.les.inf.puc-rio.br/har#wle_paper_section#ixzz3pEynaPBx