Practical Machine Learning Course Project: HAR Exercise Quality Prediction

Scott Wallace

September 21, 2014

Executive Summary

The following study was performed using the Human Activity Recognition (HAR) Weight Lifting Exercises (WLE) data set. This study uses the WLE data set to create a machine learning model which will predict the quality of exercises as categorized with the `classe` variable. Using PCA and a random forest classification model we attain an approximate **97%** prediction accuracy.

Exploratory Data Analysis

First, we download the data from the web urls provided and load into `training` and `testing` data frames. Once downloaded we analyze the data using `head` and `summary` functions. Note: the output of `head` and `summary` are left out for the sake of brevity.

```
## Loading required package: lattice
## Loading required package: ggplot2
```

Preprocessing

The first thing we notice when looking at the data is that there are some variables that should have no bearing on our model's outcome. Looking at the data set the `X`, `user_name` and timestamp columns should not affect our outcome, so we'll remove them upfront. We also notice that our test cases include many columns with missing values. We drop those columns from both the training and testing data sets.

```
# Drop unnecessary columns, including constants
drops <- c("X","user_name","raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp")
training<-training[,!(names(training) %in% drops)]
testing<-testing[,!(names(testing) %in% drops)]

#Remove columns that have values of NA for all rows in testing set
dropIndex<-NULL

cols<-ncol(training)-1

for(i in 1:cols) {
  nas<-sum(is.na(testing[,i]))
  if(nas==20) {
    dropIndex<-append(dropIndex, i)
  }
}

testing<-testing[,-dropIndex]
```

```
training<-training[,-dropIndex]

# Convert columns to numeric
cols<-ncol(training)-1
for(i in 1:cols) {
  training[,i] <- as.numeric(training[,i])
}

cols<-ncol(testing)-1
for(i in 1:cols) {
 testing[,i] <- as.numeric(testing[,i])
}
```

Training and Testing Sets

Within the training data we separate data out into training and test sets. We'll use 75% of the training set to create our model and then perform cross validation to test the accuracy of our model using the remaining 25%.

```
#Define the outcome, y, which is classe in the training set
y<-which(names(training) == "classe")

inTrain = createDataPartition(training$classe, p = 0.75, list=FALSE)
data_train = training[inTrain,]
data_test = training[-inTrain,]
```

PCA

To reduce correlation between variables we apply Principal Component Analysis (PCA) to our training and testing data sets using a 90% threshold of variance to be retained by PCA.

```
preProc<- preProcess(data_train[,-y], outcome=classe, method="pca", threshold=0.9)
trainPC<-predict(preProc, data_train[,-y])
testPC<-predict(preProc, data_test[,-y])
```

Model Fitting

Using the PCA data we train a random forest classification model using the out-of-bag error estimate training control.

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
## Loading required namespace: e1071
```

Cross Validation

We then check the performance of the model using cross validation and see that we are getting **97%** accuracy with this model. The expected out of sample error in this case is approximately **3%**.

```
confusionMatrix(data_test$classe,predict(modelFit, testPC))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    1    0    1    1
##          B   13  916   17    0    3
##          C    2    8  842    2    1
##          D    0    0   31  771    2
##          E    1    1    3    5  891
##
## Overall Statistics
##
##                Accuracy : 0.981
##                  95% CI : (0.977, 0.985)
##     No Information Rate : 0.287
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.989    0.989    0.943    0.990    0.992
## Specificity             0.999    0.992    0.997    0.992    0.998
## Pos Pred Value          0.998    0.965    0.985    0.959    0.989
## Neg Pred Value          0.995    0.997    0.987    0.998    0.998
## Prevalence              0.287    0.189    0.182    0.159    0.183
## Detection Rate          0.284    0.187    0.172    0.157    0.182
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.994    0.990    0.970    0.991    0.995
```

Classification of Test Cases

Finally, we can use this model to predict the quality values of the downloaded testing data. To do this we need to first apply the same PCA used in our training set. We then run the predict function using our model against the testing set.

```
testProblemsPC<-predict(preProc, testing[,-y])
predictions<-predict(modelFit, testProblemsPC)
problem_id<-testing$problem_id
results<-data.frame(problem_id,predictions)
print(results)
```

```
##    problem_id predictions
## 1           1           B
## 2           2           A
## 3           3           B
## 4           4           A
## 5           5           A
## 6           6           E
## 7           7           D
## 8           8           B
## 9           9           A
## 10         10           A
## 11         11           B
## 12         12           C
## 13         13           B
## 14         14           A
## 15         15           E
## 16         16           E
## 17         17           A
## 18         18           B
## 19         19           B
## 20         20           B
```