Machine Learning - Course Project

Jonathan Hall

Overview

A recent trend among tech and/or fitness enthusiasts is to quantify how *much* of a particular activity they do, however, they rarely quantify how *well* they do it. In this project we aim to use data from accelerometers attached to the belt, forearm, arm and dumbell of 6 participants while they complete repetitions of bicep curls with dumb bells to predict whether the participant was doing it correctly or not. This is identified in the data by the `classe` variable: A is performing the curls correctly, whereas the rest represent incorrect curls.

The data and accompanying study used can be found here: Qualitative Activity Recognition of Weight Lifting Exercises

Packages

```
# Libraries
library(caret)
library(ggplot2)
library(randomForest)
library(parallel)
library(doParallel)
```

Exploratory Data Analysis and Feature Selection

Importing Data

First we import our data. We set the testing set to be a *validation* set as we plan to split our training data into training and testing sets. We also convert the `classe` variable into a factor variable.

```
training_import <- read.csv('./pml-training.csv')
validating <- read.csv('./pml-testing.csv')
# classe as factor variable
training_import$classe <- as.factor(training_import$classe)
```

We split our training data into training and testing sets. The testing set will be used to test the model accuracy once we have built it before applying it to the validating set.

```
inTrain <- createDataPartition(y=training_import$classe,p=0.75, list=FALSE)
# subset spam data to training
training <- training_import[inTrain,]
# subset spam data (the rest) to test
testing <- training_import[-inTrain,]
```

Removing Zero Covariates

A first step we can take to reducing the amount of features we use in our model is to remove zero covariates. We use the `nearZeroVar` function identify features with near zero variance and then remove them from our training dataset. This is based on the `nzv` column having a value of `TRUE`.

```
ZV <- nearZeroVar(training, saveMetrics = TRUE)
# The first six Near Zero Variance predictors
head(ZV[ZV$nzv==TRUE,])
```

```
##                       freqRatio percentUnique zeroVar  nzv
## new_window             47.25574     0.0135888   FALSE TRUE
## kurtosis_roll_belt   2059.00000     2.0315260   FALSE TRUE
## kurtosis_picth_belt   655.13636     1.7053948   FALSE TRUE
## kurtosis_yaw_belt      47.25574     0.0135888   FALSE TRUE
## skewness_roll_belt   2402.16667     2.0247316   FALSE TRUE
## skewness_roll_belt.1  655.13636     1.7937220   FALSE TRUE
```

```
remove <- ZV[,4] # Logical vector for removing NZV predictors
# Remove these predictors from the training dataset
training <- training[,remove==FALSE]
```

Remove useless features

Taking a look at the first 5 variables in our dataset we can see the variables `X`, `user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp`.

```
head(training)[1:5]
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
## 5 5  carlitos           1323084232               196328 05/12/2011 11:23
## 7 7  carlitos           1323084232               368296 05/12/2011 11:23
## 8 8  carlitos           1323084232               440390 05/12/2011 11:23
## 9 9  carlitos           1323084232               484323 05/12/2011 11:23
```

All of the variables can be removed as they will not be predictive; `X` simply enumerates each observation; `user_name` is the name of the participant; and the remaining three are timestamps of the observations.

```
# Remove variables X, user_name and time stamp (character) - won't be needed
training <- training[,-c(1:5)]
```

There are also many variables which contain a lot of missing values.

```
subset(colMeans(is.na(training)),colMeans(is.na(training))>0)[1:4]
```

```
##  max_roll_belt max_picth_belt  min_roll_belt min_pitch_belt
##      0.9792771      0.9792771      0.9792771      0.9792771
```

These variables are statistics of others: max, min, amplitude etc. They add no new information to the data so we may also remove them.

```
training <- training[,colSums(is.na(training))==0]
```

Checking our features with Principal Component Analysis

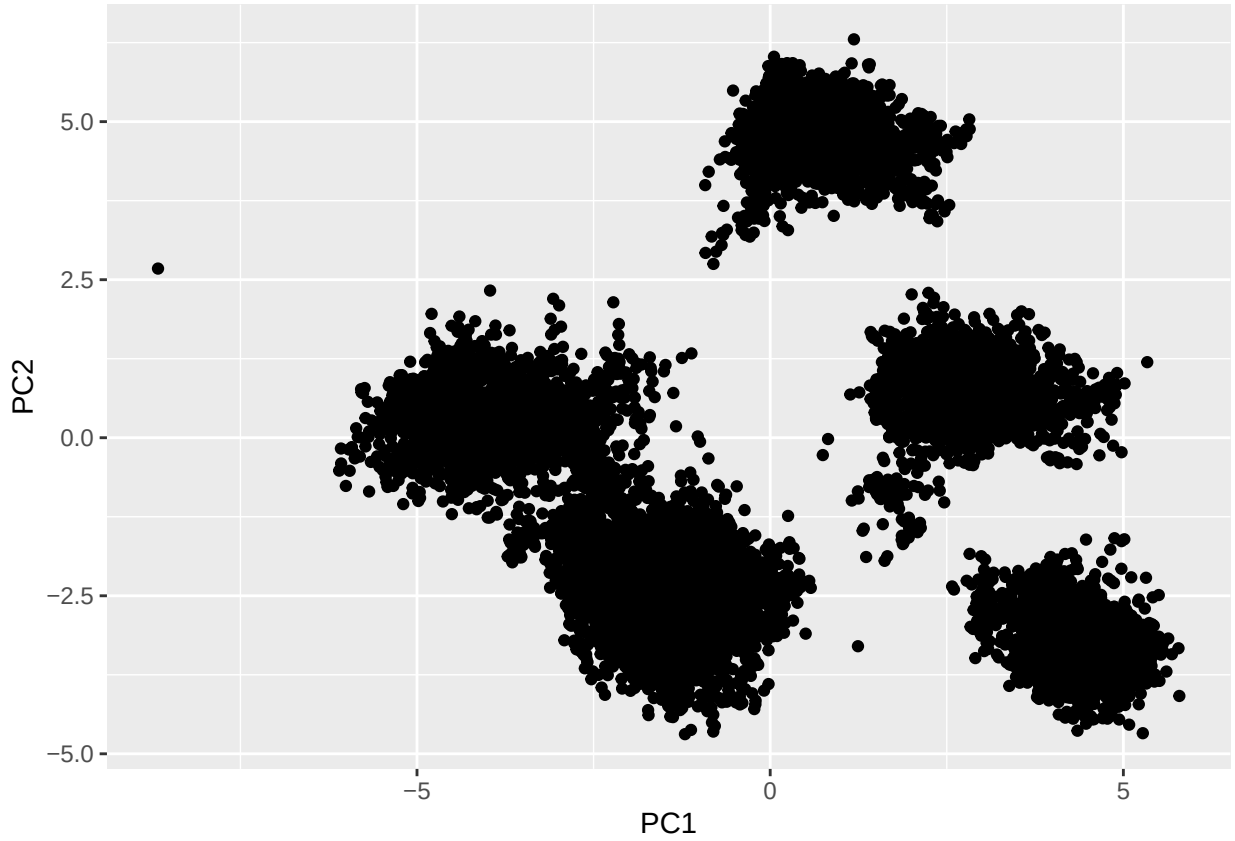
We can use the `prcomp` function to apply Principal Component Anaylsis to see how much variance is explained by the first few Principal Components.

```
pr <- prcomp(training[,-54], scale=TRUE)
data.frame(summary(pr)$importance)[1:5]
```

```
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     2.892626 2.857637 2.167771 2.071937 1.906041
## Proportion of Variance 0.157870 0.154080 0.088660 0.081000 0.068550
## Cumulative Proportion  0.157870 0.311950 0.400620 0.481610 0.550160
```

We can see by looking at the Cumulative Proportion that we do require a fair amount of features - the Cumulative Proportion does not increase very quickly. Let's plot the first two Principal Components and see if we think our model will be able to do a good job of separating the data into the `classe` variable.

```
qplot(x=PC1, y=PC2, data=data.frame(pr$x))
```



We can clearly see 5 distinct groups. This gives us confidence that model fitting should work well.

Model Fitting

We will train a Random Forest model on our training dataset and then check for its accuracy on our testing dataset. Note I am using parallel processing here to better utilise my computers hardware and allow for quicker speeds in training the model.

```
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)

fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)

fit <- train(classe ~ ., method="rf",data=training,trControl = fitControl)

stopCluster(cluster)
registerDoSEQ()
```

Checking the Confusion Matrix, we can see that our model seems to work very well with a close to 100% accuracy.

```
confusionMatrix(fit)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.4  0.0  0.0  0.0  0.0
##          B  0.0 19.3  0.0  0.0  0.0
##          C  0.0  0.0 17.4  0.1  0.0
##          D  0.0  0.0  0.0 16.3  0.0
##          E  0.0  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9978
```

We use our testing dataset, which was created as a partition of the training dataset, to gain more insight into how accurate our model is.

```
# Use testing dataset to verify
pred <- predict(fit,newdata=testing)
testing$predRight <- pred==testing$classe
# What percentage are correct predictions?
mean(testing$predRight==TRUE)
```

```
## [1] 0.9995922
```

```
# Tabulate the predictions
table(pred, testing$classe)
```

```
##
## pred    A    B    C    D    E
##    A 1395    1    0    0    0
##    B    0  948    0    0    0
##    C    0    0  855    1    0
##    D    0    0    0  803    0
##    E    0    0    0    0  901
```

Again, we see that our model is extremely accurate. The final step is to apply this model to the validating dataset to attain our answers to the quiz. I have witheld the output of this code to follow the Honour Code we all comply to in taking this course.

```
# Apply model to validating set to answer quiz questions
predict(fit, validating)
```