Predicting Correctness of Barbell Lift Exercise

Project for Practical Machine Learning Course

Simon Chan

November 18, 2015

Executive Summary

Wearable devices which can be used to track personal activities are very common nowadays. The most common usage is to track how much activity people do. Actually these devices can also be used to track how well people do the exercises. In this report I will take the training data of 6 participants doing the Barbell Lift exercise in 5 different manners with only one of the way being the correct way. I will perform feature selection, training with cross validation and estimation of errors. The trained model will then be used to evaluate a set of 20 readings of people doing the exercise and to predict which of the 5 ways they are doing it.

Data Source

The data for this report come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). For reproducibility the following are the source of the training and testing data:

Training data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

Testing data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The training data set consists of 160 variables including name of participant, time of exercise, different readings for the motions tracked by the gyrometer, acclerometer and magnetometer on sensor devices mounted on the arm, forearm, belt and Barbell and in which way (A, B, C, D, E) that the individual is doing the exercise. A is the correct way and the rest are different wrong styles of doing the exercise.

Exploratory Data Analysis

Let's first load the data and take a exploratory plot

```
library(caret)
```

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

```
set.seed(33833)
pmltrain <- read.csv("pml-training.csv"); dim(pmltrain)
```

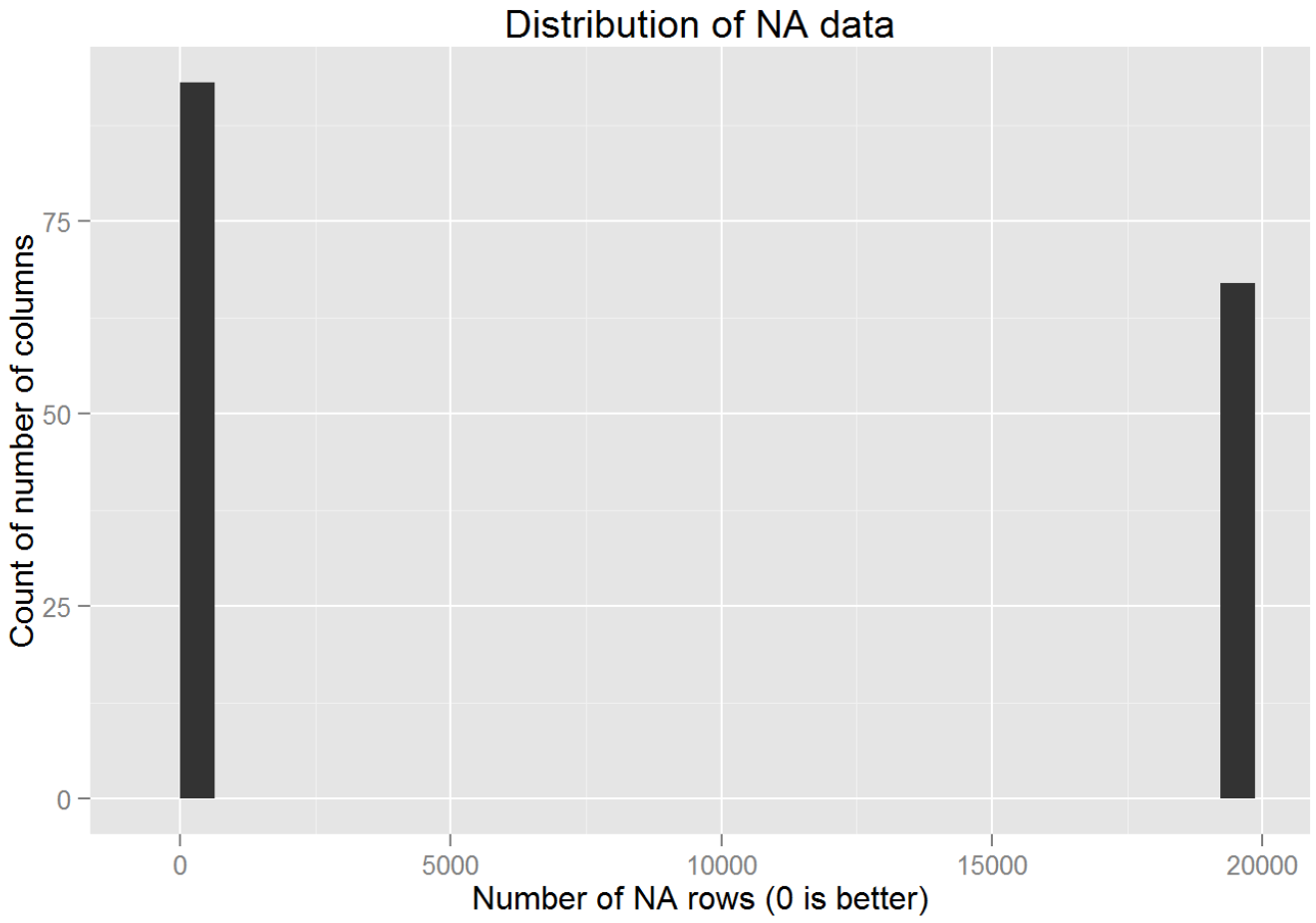
```
## [1] 19622   160
```

```
pmltest <- read.csv("pml-testing.csv"); dim(pmltest)
```

```
## [1]  20 160
```

```
ggplot(NULL, aes(x=colSums(is.na(pmltrain[,1:ncol(pmltrain)])))) + geom_histogram() +
    xlab("Number of NA rows (0 is better)") +
    ylab("Count of number of columns") +
    ggtitle("Distribution of NA data")
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



Feature Selection

From the above plot we can see there are around 60 columns with possibly all NA cells. They should provide very little predictive value. Furthermore we should also remove cells that are added on top of raw data such as time and name of individuals. We can also remove those columns with near zero variance. The following are the steps to remove the uncessary columns.

```
# Step 1. Remove columns with all NA's
data.step1 <- pmltrain[, apply(pmltrain, 2, function(x) all(!is.na(x)))];dim(data.step
1)
```

```
## [1] 19622    93
```

```
# step 2. Remove descriptive columns that is not raw data
data.step2 <- data.step1[,-(1:7)]; dim(data.step2)
```

```
## [1] 19622    86
```

```
# step 3. Remove columns with near zero variance
nzrcols <- nearZeroVar(data.step1)
if(length(nzrcols) > 0) {
    data.step3 <- data.step2[,-nzrcols]
} else {
    data.step3 <- data.step2
}
dim(data.step3)
```

```
## [1] 19622    52
```

```
# Step 4. Remove columns with over 95% blank cells
data.step4 <- data.step3[,apply(data.step3, 2, function(x)(sum(!grepl("^\\s+$|^$",x))/n
row(data.step3) > 0.05))];dim(data.step4)
```

```
## [1] 19622    30
```

Dividing Datasets for Cross Validation

I will divide the pmltrain data set into 75% training and 25% testing for cross valiation of models.

```
# Divide pmltrain dataset into 75% training and 25% testing
data.step4.sample = createDataPartition(data.step4$classe, p=0.75, list=FALSE)
data.step4.training <- data.step4[data.step4.sample,]; dim(data.step4.training)
```

```
## [1] 14718    30
```

```
data.step4.testing <- data.step4[-data.step4.sample,]; dim(data.step4.testing)
```

```
## [1] 4904   30
```

```
# Check if the testing dataset is 1/3 of training dataset and if they are evently distr
ibuted by "classe"
summary(data.step4.testing$classe)/summary(data.step4.training$classe)
```

```
##         A         B         C         D         E
## 0.3333333 0.3332163 0.3330736 0.3333333 0.3329638
```

The above result shows that the sampling is evently distributed for “classe” variable from A to E.

Train and Cross Validate Prediction Models

This is a classification problem and Random Forest (rf) method is usually the most accurate. I trained the model using cross validation with 3 folds. Then I further tested the out-of-sample accuracy using the testing set. Finallyl the accuracy is measured using a confusion matrix.

Random Forest model Training

```
# Train the Random Forest (rf) model
control.rf <- trainControl(method="cv", number=3, allowParallel = TRUE)
fit.rf <- train(classe ~.,data.step4.training, method="rf", trControl=control.rf, verbo
se=FALSE)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
fit.rf
```

```
## Random Forest
##
## 14718 samples
##    29 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9812, 9813, 9811
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9896725  0.9869353  0.001543719  0.001953155
##   15    0.9892648  0.9864197  0.002414761  0.003055136
##   29    0.9829459  0.9784249  0.003591861  0.004544724
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Using the sliced testing data set for cross validation

```
# Cross validate with testing dataset
data.step4.rf.prediction <- predict(fit.rf, newdata = data.step4.testing[,-30])
```

Estimation of out of sample errors

```
# Error for Random Forest prediction
cm <- confusionMatrix(data.step4.rf.prediction,data.step4.testing$classe); cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    1    0    0    0
##          B    3  945    7    0    1
##          C    0    3  848    7    1
##          D    0    0    0  797    1
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9958   0.9918   0.9913   0.9967
## Specificity            0.9997   0.9972   0.9973   0.9998   1.0000
## Pos Pred Value         0.9993   0.9885   0.9872   0.9987   1.0000
## Neg Pred Value         0.9991   0.9990   0.9983   0.9983   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1927   0.1729   0.1625   0.1831
## Detection Prevalence   0.2841   0.1949   0.1752   0.1627   0.1831
## Balanced Accuracy      0.9988   0.9965   0.9945   0.9955   0.9983
```

The accuracy statistic as seen from the out-of-sample testing dataset prediction is 99.5106036% which is very good. The error rate is 0.4893964%.

Predict with the Provided Testing Dataset

I now try to predict the 20 provided exercise readings and determine which style (A - E) they belong using the trained Random Forest model.

```
# select the features from columns similar to the training data set
data.test <- pmltest[,names(data.step4[,-30])]
# predict using random forest
data.test.prediction <- predict(fit.rf, newdata = data.test)
data.test.prediction
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

The above results are the predicted styles (A-E) for the provided testing dataset.

Output of the prediction results

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(data.test.prediction)
```