Machine Learning Course Project

Scott Ziemke

April 19, 2016

Introduction

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, my goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

Libraries and Data

I will use caret, rpart, and randomForest Libraries.

```
library(caret)
library(rpart)
library(randomForest)
library(knitr)
library(markdown)
library(rmarkdown)

training <- read.csv("pml-training.csv", header = TRUE)
```

Removing Irrelevant Variables and Cleansing

There are many variables that are irrelevant to predicting classe. Given there are 160 variables in the dataset including classe, I am going to spend most of my effort reducing this dataset to speed the calculation while still maintaining accuracy.

First, I found that there were many with many NAs and many variables that were constant or nearly constant accross all observations. The following scripts were used to remove variables with NAs and with less than 3 unique instances per variable. THis takes the number of variables from 160 to 84.

```
#remove irrelevant variables
na_count_train <-as.data.frame(sapply(training, function(y) sum(length(which(is.na(y))))))
var_list <- row.names(subset(na_count_train,na_count_train == 0))
training <- training[,var_list]
low_unique <- as.data.frame(sapply(training, function(y) length(unique(y))))
var_list2 <- row.names(subset(low_unique, low_unique[,1] >3))
training <- training[,var_list2]
```

Next, I will convert all variables to numeric.

```r
#clean data
training <- (sapply(training, function(y) as.character(y)))
training <- suppressWarnings(cbind(training[,1:6],apply(training[,7:83], 2, function(x){as.numeric(x)})
training <- cbind(training[,1:6],apply(training[,7:83], 2, function(x){replace(x, is.na(x), 0)}), train
training <- as.data.frame(training, stringsAsFactors= FALSE)
for (i in 7:83){
    training[,i] <- as.numeric(training[,i])
}
names(training)[names(training)=="V84"] <- "classe"
training$amplitude_yaw_belt <- NULL
```

Finally, I will remove variables that are highly correlated (>75%) and remove other variables that do not predict classe. This takes the variable list from 84 to 52 including classe.

```r
#remove variables w high correlation
correlationMatrix <- cor(training[,7:82])
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)
training <- training[-c(1 +highlyCorrelated)]
training$X <- NULL
training$num_window <- NULL
training$raw_timestamp_part_2 <- NULL
```

Cross Validation Set Up

I will partion the data so that 3/4 of the data is in the training data set while 1/4 is in the testing dataset.

```r
#create train set and test set
inTrain = createDataPartition(training$classe, p = 3/4)[[1]]
training <- training[ inTrain,]
testing <- training[-inTrain,]
```

Model Fitting and Conclusions

```r
#initial test of accuracy using all variables and random forest
set.seed(33833)
fit1 <- train(factor(classe)~., method = "rf", data = training)
fit1
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9833875  0.9789751  0.002289539  0.002898130
##   27    0.9900279  0.9873830  0.001861399  0.002352523
##   52    0.9840088  0.9797665  0.003362554  0.004257413
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

The accuracty of the random forrest model on the training data set is 99.05%.

```
predicted <- predict(fit1,testing[,1:52])
confusionMatrix(predicted, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1058    0    0    0    0
##          B    0  677    0    0    0
##          C    0    0  666    0    0
##          D    0    0    0  617    0
##          E    0    0    0    0  660
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2877
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2877   0.1841   0.1811   0.1678   0.1794
## Detection Rate         0.2877   0.1841   0.1811   0.1678   0.1794
## Detection Prevalence   0.2877   0.1841   0.1811   0.1678   0.1794
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

We are seeing 100% accuracy when applying the model to the test set.