Coursera Practical Machine Learning Project

Mark S.

Wednesday, July 22, 2015

Project Summary

The goal of this project is to build a model that can predict the manner in which an exercise is being performed. We are given two sets of data, a training set and a test set. The source and description of the data is found here: http://groupware.les.inf.puc-rio.br/har. The data can be downloaded via the following links:

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

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

Data Overview and Strategy

My interpretation of the problem is as follows: "Given a set of measurements *in this moment*, predict how the exercise is being performed by any person *in this moment*."

The "*in this moment*" phrase is emphasized to establish the fact that we want a model that can generalize to any moment in time. We don't want to make a prediction based off of measurements that occured in the past; rather we would like to make predictions based on single measurements occuring at a single moment in time. Also, we would like a model that can generalize to any user. With this reasoning, we will ignore any variables provided that are considered "window statistics", variables involving time, and the names of the participants. This includes the first 7 variables of the data set, as well as any variable name containing min, max, avg, var, std, total, kurtosis, skewness, or amplitude.

Since this is a classification task, we'll use a random forest and a gradient boosted machine. The entire process will be structured as follows:

1. Remove unwanted variables (explained above)
2. Split training data into 2 sets; call them "A" and "B" with a 70/30 split.
3. Perform data analysis and preprocessing using data set "A"
4. The models will be trained on set "A" using cross validation, and performance evaluated via set "B". This will give us an out of sample error estimate of the model.
5. The final chosen model will be used to make predictions for the submission part of the assignment.

Downloading and Cleaning the Data

First we download the data, and remove the unwanted columns, as explained previously:

```
#Set global seed for reproducibility
set.seed(8080)

#Load the data; use library RCurl to allow download of data in knitr
#note that some of the data contain "NA", "#DIV/0!" and empty values. These will be considered "NA"
temp<-getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",ssl.verifypeer=0L,f
data<- tbl_df(read.csv(text=temp, na.strings = c("NA","#DIV/0!","")))

temp<-getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",ssl.verifypeer=0L,fo
```

```
final.test<- tbl_df(read.csv(text=temp, na.strings = c("NA","#DIV/0!","")))

#remove temp variable
rm(temp)

#First 7 columns aren't measurements; they won't be included
data<- data[,-(1:7)]
final.test<- final.test[,-(1:7)]

#Create a vector of all feature names based on time or window statistics
feats<- grep("^min|max|avg|var|std|total|kurtosis|skewness|amplitude",names(data))

#Remove these features from the data
data<- data[,-feats]
final.test<-final.test[,-feats]
```

Now we split the data into train and test sets with a 70/30 split. The "final.test" set is the one we will make predictions on and submit.

```
#split data into train/test sets
n<- createDataPartition(y=data$classe,p=0.7,list=FALSE)
train<- data[n,]
test<- data[-n,]

#convert data to tbl_df to use with dplyr
train<- tbl_df(train)
test<- tbl_df(test)
```

Exploratory Data Analysis

First check the dimensionality of the data:

```
dim(train)
```

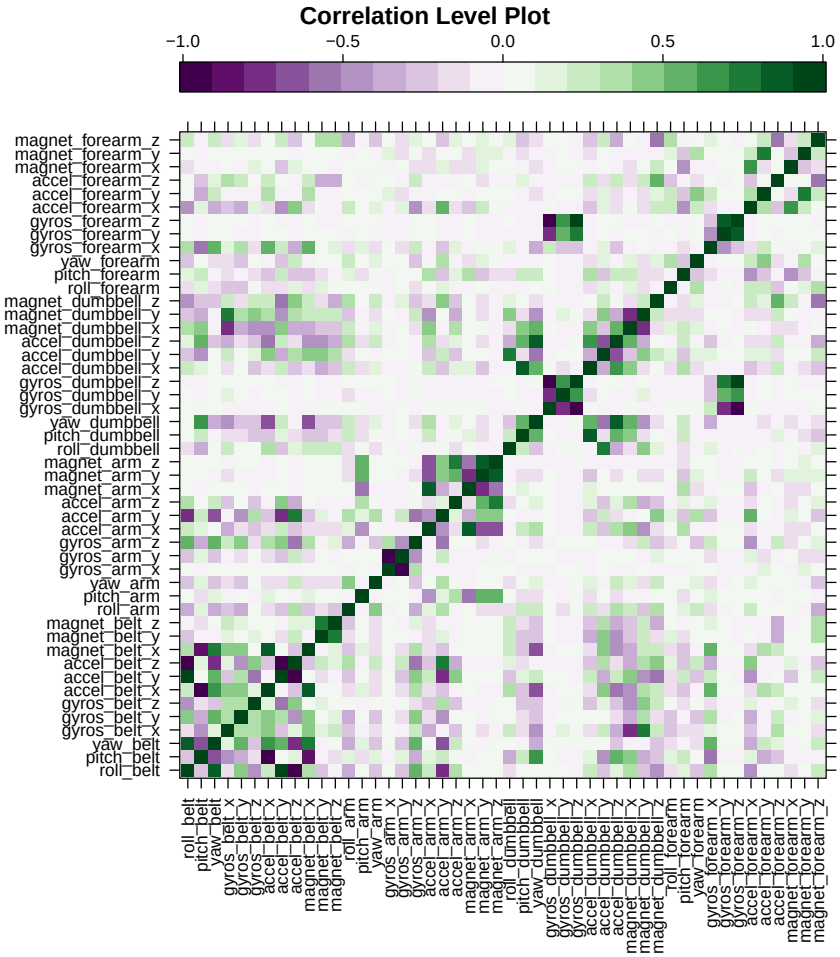
```
## [1] 13737    49
```

We've already eliminated many of the features (going from 159 to 48). We should also check to see if we have any features with near-zero variance:

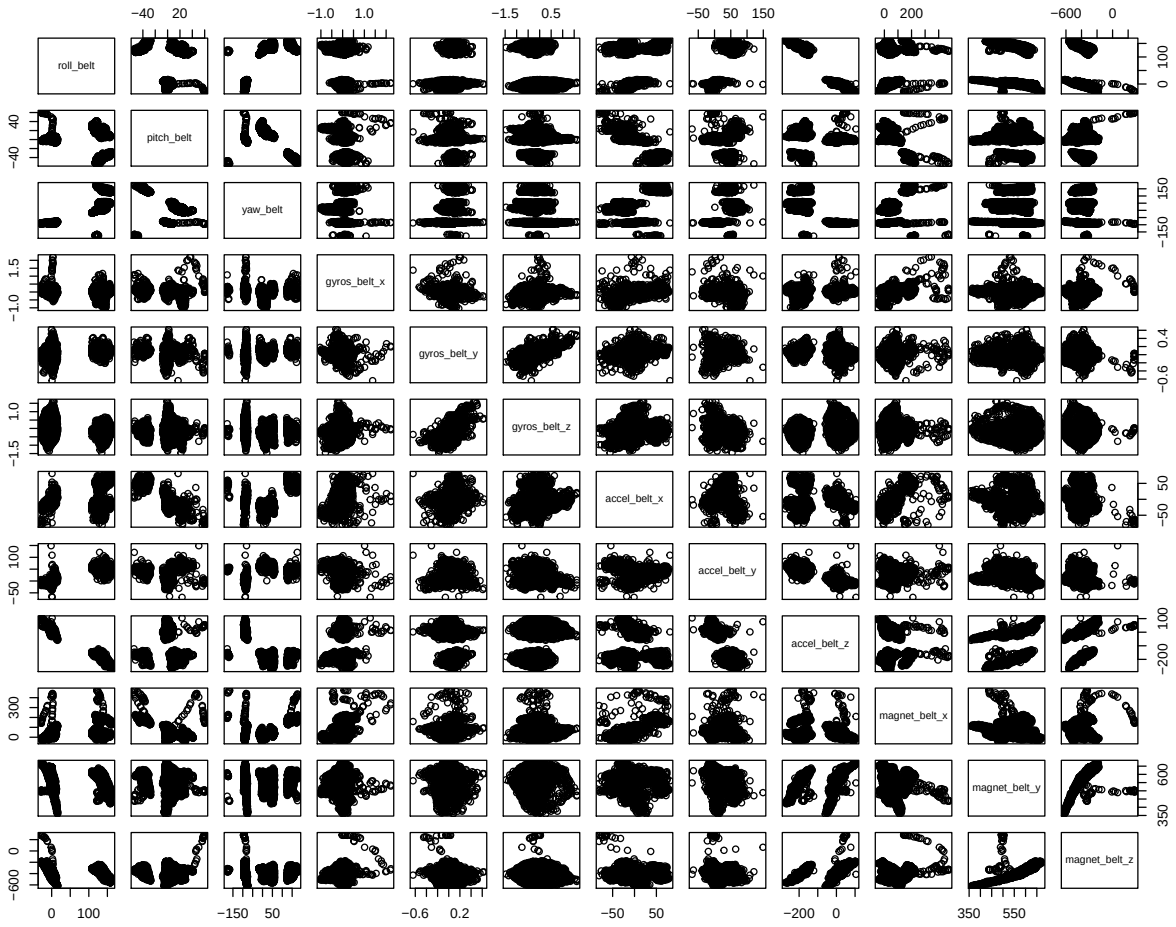
```
nearZeroVar(train)
```

```
## integer(0)
```

It would be difficult to plot feature pairs, but we can visualze the correlations to identify variables to inspect:



Based on the correlation plot it seems that some variables are highly correlated. We can inspect some of these further; let's take a look at the variables containing "belt" (variables in the bottom left corer of the correlation plot):



We observe some strange patterns between the variables. It's not obvious which variables should be removed based on inspection, so we will keep all of them for now. Since we are using a tree-based model, more preprocessing isn't necessary. We could try to reduce dimensionality via PCA, but we will continue using the data as is for now.

Building the Models

We'll build a random forest and gradient boosted model using 5-fold cross-validation:

```
fitControl <- trainControl(method = "cv",number = 5)
#Build the models; number of trees was chosen to reduce the time to build the models
mod1<- train(classe~., data = train, method = "rf",ntree= 20,trControl = fitControl, allowParallel=TRUE)
mod2<- train(classe~., method = "gbm", data = train, trControl = fitControl, tuneGrid = expand.grid(n.t
```

Next, test the models on the test set:

```
#Make predictions on the test set
pred1<- predict(mod1,test[,-49])
pred2<- predict(mod2,test[,-49])
```

```r
#Print the confusion matrix for each models predictions
confusionMatrix(pred1,test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    7    0    0    0
##          B    2 1127    5    0    1
##          C    2    5 1010    6    1
##          D    0    0   11  957    4
##          E    0    0    0    1 1076
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9898, 0.9944)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9903
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9895   0.9844   0.9927   0.9945
## Specificity            0.9983   0.9983   0.9971   0.9970   0.9998
## Pos Pred Value         0.9958   0.9930   0.9863   0.9846   0.9991
## Neg Pred Value         0.9990   0.9975   0.9967   0.9986   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1915   0.1716   0.1626   0.1828
## Detection Prevalence   0.2850   0.1929   0.1740   0.1652   0.1830
## Balanced Accuracy      0.9980   0.9939   0.9908   0.9948   0.9971
```

```r
confusionMatrix(pred2,test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1419  198   46   89   67
##          B   47  755   87   29  111
##          C   50  107  833   92   90
##          D  128   75   56  711   57
##          E   30    4    4   43  757
##
## Overall Statistics
##
##                Accuracy : 0.7604
##                  95% CI : (0.7493, 0.7713)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6962
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8477   0.6629   0.8119   0.7376   0.6996
## Specificity            0.9050   0.9423   0.9302   0.9358   0.9831
## Pos Pred Value         0.7801   0.7337   0.7108   0.6923   0.9033
## Neg Pred Value         0.9373   0.9209   0.9590   0.9479   0.9356
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2411   0.1283   0.1415   0.1208   0.1286
## Detection Prevalence   0.3091   0.1749   0.1992   0.1745   0.1424
## Balanced Accuracy      0.8763   0.8026   0.8711   0.8367   0.8414
```

The random forest performs better with an out-of-sample error estimate of approximately 0.8%. This model will be used to predict the final test set:

```
predict(mod1,final.test[,-49])
```

```
##  [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
```