Machine Learning Course Project - Weight Lifting Exercise Prediction

kshiraishi

May 19, 2018

Executive Summary

Using fitness device data, we fit a machine learning model to predict a manner of exercises using multiple accelerometer data. Using a random forest model, we are able to classify with over 99% accuracy against our test set.

All data is generously provided from http://groupware.les.inf.puc-rio.br/har. More information about the data set and experiment can be found here (Velloso 2013).

Load Libraries

Relevant libraries and testing/training variables are loaded. The boruta library is used for feature selection. The parallel library is used to take advantage of multi-core processing in R.

```
library(dplyr)
library(ggplot2)
library(Boruta)
library(parallel)
library(doParallel)
library(caret)
```

Data Setup and Training/Test Sets: Hindsight is 20/20

We make things easy for ourselves by manually downloading the data and storing locally.
The first 8 columns are non-accelerometer data, so we remove those. The remaining columns are all accelerometer data and should be numeric (except for the last column predictor "classe"" variable) so we explicitly convert. (Not explicitly shown due to space contraints and for readability.)

We split our training data file (pml-training.csv) into a training data set and testing data set, to be used for cross-validation.

Inspection of the final validation data set (pml-testing.csv) reveals a number of columns that have all 'NA' values. We remove those columns from the training, testing, and the validation data set, leaving us with 20 columns to use in our model.

```
rawtrain <- read.csv('./pml-training.csv')
rawtrain <- rawtrain[,8:160]
rawtrain[,1:152] <- data.frame(lapply(rawtrain[,1:152], function(x) as.numeric(as.character(x))))

inTrain <- createDataPartition(y=rawtrain$classe, p=0.7,list=FALSE)
training <- rawtrain[inTrain,]
testing <- rawtrain[-inTrain,]

validation <- read.csv('./pml-testing.csv')
validation <- validation[,8:160]
```

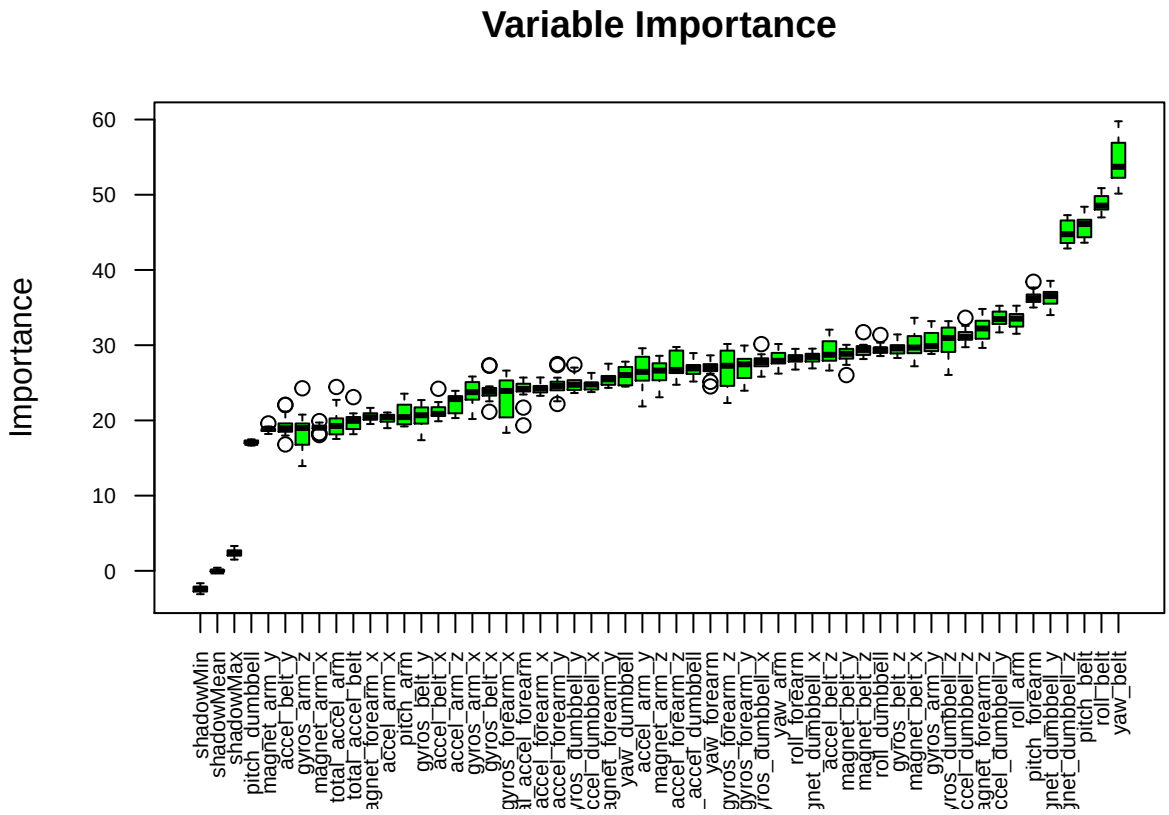
```
validation[,1:152] <- data.frame(lapply(validation[,1:152], function(x) as.numeric(as.character(x))))
validation <- validation[,colSums(is.na(validation))<nrow(validation)]

columnsToKeep <- colnames(validation)
columnsToKeep[length(columnsToKeep)] <- "classe"
training <- training %>% select(columnsToKeep)
testing <- testing %>% select(columnsToKeep)
```

Feature Selection using boruta

Given the 20 features, we should attempt to remove unneccessary columns to save on processing time and overfitting. We use the boruta package to indicates any such features and remove them from our training sets. (boruta was chosen since there are a lot of different instructions on the internet.) We only keep the "Confirmed" features, and add back in the "classe" variable. For good measure we plot the relative importance of each variable.

```
set.seed(1)
boruta_output <- Boruta(classe ~ ., data=training)
relVal <- getSelectedAttributes(boruta_output, withTentative=F)
relVal[length(relVal)+1] <- "classe"
training <- training %>% select(relVal)
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")
```



Applying a Random Forest Model

Now that we have our training and testing data set, we can fit a random forest model. To take advantage of multi-core processing, we following the detailed instructions here (lgreski, n.d.).

```
set.seed(1)

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()

predTesting <- predict(fit,testing)
confusionMatrix(predTesting,testing$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.9901444
```

The accuracy of our random forest model is over 0.99, so our current model is sufficient; we don't need to bother with other models or combining them.

Predict our Unknown Values

Using this random forest model we use this to predict against our validation set.

```
predValidation <- predict(fit, validation)
predValidation
```

```
##  [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
```

References

lgreski. n.d. "Improving Performance of Random Forest in Caret:train()." https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md.

Velloso, A.; Gellersen, E.; Bulling. 2013. *Qualitative Activity Recognition of Weight Lifting Exercises.* Stuttgart, Germany: ACM SIGCHI.