Prediction of Barbell Lift Quality

ksplett1

Sunday, March 22, 2015

Practical Machine Learning Project

(Coursera predmachlearn-012)

Summary

This project predicts quality of barbell lifts for Unilateral Dumbbell Biceps Curl. The data set and related research can be found at http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (section on Weight Lifting Exercise). As described in the “Qualitative Activity Recognition of Weight Lifting Exercises” paper, users were recorded while they performed the same activity correctly and incorrectly based on a set of common mistakes. Wearable sensors (belt, glove, arm-band, and dumbbell) provided measurements of each mistake.

In this project, 5 random forests are used as a classification model, based on 500 trees.

Random Forest was selected as the classification model for the following reasons:

- Classification trees can predict non-linear models
- Predictor variables do not require normalization and the trees automatically rank variable significance
- Random Forest models are top-performing models
- Random Forest models can handle many predictor variables

Cross validation was done by partitioning the training data set into separate sub-training and validation sets. The random forest model was trained on the sub-training set, and then the model prediction was run on the validation set. The predicted classe output was compared to the actual classe output in the validation set to produce a confusion matrix. The accuracy percent was also calculated. Cross-validation checks for Type III or out-of-sample errors (errors suggested by the data - i.e., over-fitting) estimated a 97.6% out-of-sample accuracy.

note: Because the training set is large, the training set selected to train the model was 30% (4907 observation and 54 variables) of the original training set size. The decision to reduce the training set size was made because the model took about 15 minutes to run on a laptop. The resulting model is highly accurate, so further iterations of the model were deemed unnecessary.

A variable reduction method could also have been applied to decrease the model training time and remove lesser significant variables. A single classification tree was initially run to identify significant variables rapidly; however, this approach was not chosen as the final model because it would have required more development time to find the best reduced set of predictor variables.

Another possible model approach (not chosen) would be to re-summarize the raw device measurements (ex. mean, std deviation, total acceleration (sqrt(x^{2+y}2+z^2), max+std), which seems to be a common way to analyze accelerometer data. Again, this model would have also required more development time.

Data Processing

Data Set Description:

- 19,621 observations collected from six participants. 160 variables.
- .csv file format
- Measurements are taken from accelerometers on the belt, forearm, arm, and dumbbell
- classe output variable: Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes (throw elbows to the front (Class B), lift dumbbell only halfway (Class C), lower dumbbell only halfway (Class D) and throw hips to the front (Class E)).
- Training data is sorted by class output
- Training data is sequentially timestamped within a time window (indicated by new_window)
- Summarized measurements (kurtosis, skewness, variance, average, standard deviation, minimum, maximum) are calculated once per time window

---

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

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

```
## Warning: package 'ggplot2' was built under R version 3.1.1
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
# assume setwd done before running this program
setwd("C:/Users/ksplett1/machinelearning")
setwd("data")
df <- read.csv("pml-training.csv", header = TRUE, sep = ",", na.strings = "NA" )
df_test <- read.csv("pml-testing.csv", header = TRUE, sep = ",", na.strings = "NA" )
setwd("../")

# Remove window summary columns that start with new_window, amplitude, kurtosis, skewness, var, avg,
stddev, min, max
df_obs <- df[ , -grep("^(new_window|amplitude|kurtosis|skewness|var|avg|stddev|min|max)", names(df))
]
df_tst <- df_test[ , -grep("^(new_window|amplitude|kurtosis|skewness|var|avg|stddev|min|max)", name
s(df_test)) ]

# Not performing sequence or time series analysis - remove timestamp columns
# remove raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, X, user_name, num_window
df_obs <- df_obs[ , -grep("^(raw_timestamp_part_1|raw_timestamp_part_2|cvtd_timestamp|X|user_name)",
names(df_obs)) ]
df_tst <- df_tst[ , -grep("^(raw_timestamp_part_1|raw_timestamp_part_2|cvtd_timestamp|X|user_name)",
names(df_tst)) ]

# Scale data set down to a runnable size
InTrain <- createDataPartition(y=df_obs$classe, p=0.25, list=FALSE)
training <- df_obs[InTrain,]
validating <- df_obs[-InTrain,]

dim(training)
```

```
## [1] 4907   54
```

Model Build

```
# Classification model using Random Forest.
#  5 fold cross validation
set.seed(100)
rfModFit <-train(classe ~.-num_window, data=training, method="rf",
           trControl=trainControl(method="cv", number=5), prox=TRUE ,allowParallel=TRUE
           , importance = TRUE, verbose = TRUE)

print(rfModFit)
```

```
## Random Forest
##
## 4907 samples
##   53 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 3924, 3926, 3926, 3925, 3927
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa   Accuracy SD  Kappa SD
##    2    0.9694    0.9613  0.007548     0.009556
##   27    0.9729    0.9657  0.006797     0.008594
##   52    0.9676    0.9590  0.002650     0.003359
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
print(rfModFit$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE,      proximity = TRUE, allowPara
llel = TRUE, verbose = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 2.2%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1380   9   3   1   2     0.01075
## B   22 916  11   0   1     0.03579
## C    0  16 833   6   1     0.02687
## D    1   1  11 789   2     0.01866
## E    0   5   6  10 881     0.02328
```

Prediction Results

```
# Prediction output for Test set
# note: All test set outputs were predicted correctly
predModFit <- predict(rfModFit, df_tst)

# Prediction output for Validation set
predModFit <- predict(rfModFit, validating)

# Confusion Matrix
table( predModFit, validating$classe )
```

```
##
## predModFit    A    B    C    D    E
##          A 4146   74    0    0    0
##          B   30 2740   31    8    3
##          C    5   27 2509   69    3
##          D    3    3   26 2332   19
##          E    1    3    0    3 2680
```

Validation accuracy = 0.9791

```
# variable importance
varImp(rfModFit)
```

```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 52)
##
##                        A    B    C    D      E
## roll_belt         72.19 84.3 75.9 70.9 100.00
## pitch_belt        26.10 87.6 52.4 42.2  29.87
## magnet_dumbbell_y 66.38 60.5 78.5 60.0  52.46
## pitch_forearm     57.20 67.8 78.1 52.2  57.65
## magnet_dumbbell_z 69.87 48.2 68.2 47.4  46.11
## yaw_belt          63.19 49.3 58.2 58.3  46.33
## accel_forearm_x   21.76 35.6 36.3 50.6  39.05
## roll_forearm      49.99 37.9 43.9 30.0  32.16
## yaw_arm           40.95 17.0 23.6 25.2  14.60
## accel_dumbbell_y  31.99 28.4 38.9 26.2  29.20
## gyros_dumbbell_y  27.84 19.5 36.4 21.1  14.11
## roll_dumbbell     25.99 33.5 28.7 33.5  33.02
## gyros_belt_z      20.72 23.1 28.1 24.7  30.78
## magnet_belt_z     17.30 28.3 19.4 30.4  23.56
## accel_dumbbell_z  30.00 30.2 20.0 27.4  27.81
## yaw_dumbbell      12.98 28.2 22.2 17.5  19.95
## magnet_belt_x     12.03 26.7 27.8 16.5  21.11
## magnet_dumbbell_x 25.14 22.9 25.8 27.4  18.68
## magnet_belt_y     17.01 27.2 26.7 23.8  23.97
## gyros_forearm_y    7.96 26.4 13.9 12.4   5.89
```