Machine Learning: Predicting how well an exercise was preformed using FitBit data

Christopher Ellis

06/06/2015

Executive summary

This document analyzes how well participants performed barbell lifts. The participants performed the barbell lifts correctly and incorrectly in 5 different ways and the results were recorded from accelerometers on the belt, forearm, arm, and dumbbell. First, the entire training set was used as an initial prediction (70 % success rate). Then the training set was split into 15 folds and a voting model was used to predict the correct answer. Surprisingly, this preformed worse than the original model (60 % success rate). The results from each model are presented in the following table.

method	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Test 1	A	A	C	A	A	E	D	B	A	A	B	E	B	A	A	E	A	A	C	B
Test 2	A	A	C	A	C	E	D	B/C	A	A	A	C	B	A	A	B	A	B	C	B
Correct	B	A	?	A	A	E	D	B	A	A	B	?	B	A	?	E	A	B	B	B

Exploratory analysis

Read in the training and test sets.

```r
library(caret)
```

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

```r
set.seed(1)
training <- read.csv('pml-training.csv', header=TRUE)
testing  <- read.csv('pml-testing.csv',  header=TRUE)
```

Isolate columns used for analysis.

```r
set.seed(1)
train_classe<-training[,grepl("classe",names(training))]
train_totals<-training[,grepl("^total",names(training))]
parse_training<-cbind(train_classe,train_totals)

test_id       <- testing[,grepl("problem",names(testing))]
test_totals   <- testing[,grepl("^total",names(testing))]
parse_testing <- cbind(test_id, test_totals)
```

This makes a random forest using all data in the test set (19,622) to predict on 20 in the test set. This using 99.9% of the data in the training set to prediect on test set which accounts for the remaining 0.1% of the data. Not the best way, however, it does predict 70% of the test set correctly. The last line of this chunk prints out the predictions.

```
set.seed(1)
rf_mod   <- train(train_classe~., method= 'rf', data = parse_training)
```

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

```
rf_pred  <- predict(rf_mod, parse_testing)
rf_pred
```

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

To build a better model the K-fold splitting method was applied to the training set.

```
set.seed(1)
library(caret)
folds<-createFolds(y=parse_training$train_classe, k=15, list=TRUE, returnTrain=FALSE)
sapply(folds, length)
```

```
## Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
##   1308   1309   1308   1308   1307   1308   1308   1309   1308   1307
## Fold11 Fold12 Fold13 Fold14 Fold15
##   1308   1309   1308   1309   1308
```

```
names(folds)[1] <- "train"
```

Isolate folds. I wasn't able to make a *for* loop work because of the data type, so I just listed them out manually.

```
set.seed(1)
#Perform random forest machine learning algorithm to each fold
rf_fold01   <- train(train_classe~., method= 'rf', data = parse_training[folds[[1]],])
rf_fold02   <- train(train_classe~., method= 'rf', data = parse_training[folds[[2]],])
rf_fold03   <- train(train_classe~., method= 'rf', data = parse_training[folds[[3]],])
rf_fold04   <- train(train_classe~., method= 'rf', data = parse_training[folds[[4]],])
rf_fold05   <- train(train_classe~., method= 'rf', data = parse_training[folds[[5]],])
rf_fold06   <- train(train_classe~., method= 'rf', data = parse_training[folds[[6]],])
rf_fold07   <- train(train_classe~., method= 'rf', data = parse_training[folds[[7]],])
rf_fold08   <- train(train_classe~., method= 'rf', data = parse_training[folds[[8]],])
rf_fold09   <- train(train_classe~., method= 'rf', data = parse_training[folds[[9]],])
rf_fold10   <- train(train_classe~., method= 'rf', data = parse_training[folds[[10]],])
rf_fold11   <- train(train_classe~., method= 'rf', data = parse_training[folds[[11]],])
rf_fold12   <- train(train_classe~., method= 'rf', data = parse_training[folds[[12]],])
rf_fold13   <- train(train_classe~., method= 'rf', data = parse_training[folds[[13]],])
rf_fold14   <- train(train_classe~., method= 'rf', data = parse_training[folds[[14]],])
rf_fold15   <- train(train_classe~., method= 'rf', data = parse_training[folds[[15]],])
```

```
set.seed(1)
#Perform predictions on the test set for each model
rf_pred01   <- predict(rf_fold01, parse_testing)
rf_pred02   <- predict(rf_fold02, parse_testing)
rf_pred03   <- predict(rf_fold03, parse_testing)
rf_pred04   <- predict(rf_fold04, parse_testing)
rf_pred05   <- predict(rf_fold05, parse_testing)
rf_pred06   <- predict(rf_fold06, parse_testing)
rf_pred07   <- predict(rf_fold07, parse_testing)
rf_pred08   <- predict(rf_fold08, parse_testing)
rf_pred09   <- predict(rf_fold09, parse_testing)
rf_pred10   <- predict(rf_fold10, parse_testing)
rf_pred11   <- predict(rf_fold11, parse_testing)
rf_pred12   <- predict(rf_fold12, parse_testing)
rf_pred13   <- predict(rf_fold13, parse_testing)
rf_pred14   <- predict(rf_fold14, parse_testing)
rf_pred15   <- predict(rf_fold15, parse_testing)
```

Sum the predictions of each set, the highest rated answer was selected as the answer.

```
set.seed(1)
rf_pred01<- as.character(rf_pred01)
rf_pred02<- as.character(rf_pred02)
rf_pred03<- as.character(rf_pred03)
rf_pred04<- as.character(rf_pred04)
rf_pred05<- as.character(rf_pred05)
rf_pred06<- as.character(rf_pred06)
rf_pred07<- as.character(rf_pred07)
rf_pred08<- as.character(rf_pred08)
rf_pred09<- as.character(rf_pred09)
rf_pred10<- as.character(rf_pred10)
rf_pred11<- as.character(rf_pred11)
rf_pred12<- as.character(rf_pred12)
rf_pred13<- as.character(rf_pred13)
rf_pred14<- as.character(rf_pred14)
rf_pred15<- as.character(rf_pred15)

for(i in 1:20){
ans<- c(rf_pred01[i],rf_pred02[i],rf_pred03[i],rf_pred04[i],rf_pred05[i],
        rf_pred06[i],rf_pred07[i],rf_pred08[i],rf_pred09[i],rf_pred10[i],
        rf_pred11[i],rf_pred12[i],rf_pred13[i],rf_pred14[i],rf_pred15[i])
#print out model number and determine most likey prediction
print(i)
result<-summary(as.factor(ans))
print(result)
}
```

```
## [1] 1
##  A  C
## 13  2
## [1] 2
##  A  B
## 14  1
## [1] 3
## A B C D E
## 1 1 9 1 3
## [1] 4
##  A  D
## 14  1
## [1] 5
## A B C D
## 3 3 7 2
## [1] 6
## C D E
## 5 2 8
## [1] 7
##  D  E
## 10  5
## [1] 8
## B C E
## 6 6 3
## [1] 9
##  A
## 15
## [1] 10
##  A
## 15
## [1] 11
## A B D
## 8 4 3
## [1] 12
## A B C D E
## 3 1 8 2 1
## [1] 13
##  B  E
## 10  5
## [1] 14
##  A
## 15
## [1] 15
##  A  B  D
## 13  1  1
## [1] 16
## B E
## 8 7
## [1] 17
##  A  B  E
## 12  1  2
## [1] 18
## A B D
## 5 8 2
## [1] 19
## B C E
## 5 9 1
## [1] 20
##  B
## 15
```