Machine Learning Course Project. Human Activity Recognition

Andrey Vlasenko

13.04.2017

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement вЂ“ a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here http://groupware.les.inf.puc-rio.br/har.

The goal of the project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Summary

In this work for classification task we take into account 5 different models: - random forest (rf), - decision tree (rpart), - stochastic gradient boosting (gbm), - linear discriminant analysis (lda), - support vector machine (svm).

There was 2 input sets: - training (pml-training.csv) which was split to train `trnSet_trn` (biulding the models) and test sets `trnSet_tst` (cross-validation). - testing (pml-testing.csv) which was used in final quiz.

To compare models we estimate out of sample error (accuracy). The best result demonstrate RF (0.99), GBM (0.96) and SVM (0.93) models.

```
library(caret); library(randomForest);
library(rpart); library(rpart.plot); library(plyr)
library(gbm); library(e1071); library(MASS)
```

Data preprocessing

Loading and clearing the data.

```
training<-read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                   na.strings=c("", "NA", "NULL", "#DIV/0!")) # there are NA values in the data
testing<-read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                   na.strings=c("", "NA", "NULL", "#DIV/0!")) # there are NA values in the data
c(dim(training),dim(testing)) # dimensions of training and testing sets without NA vars
```

```
## [1] 19622   160    20   160
```

```
table(training$classe) # there are 5 outcomes.
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
NAVals <- apply(is.na(training),2,sum)/dim(training)[1] # share of NA values in data
NAVals <- NAVals[NAVals>0.95]; NAnms<-names(NAVals); # NAnms - names of useless variables
trnSet <- training[,!(names(training) %in% NAnms)] # removing NA variables from training
tstSet <- testing[,!(names(training) %in% NAnms)] # removing NA variables from testing
trnSet <- trnSet[,-c(1:7)] # we have to remove first 7 variables linked with time
tstSet <- tstSet[,-c(1:7)] # and number of the observations to avoid overfitting of the model
c(dim(trnSet),dim(tstSet)) # dimensions of training and testing sets without NA vars
```

```
## [1] 19622    53    20    53
```

```r
str(trnSet)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
ValsOfVars<-apply(trnSet, 2, function(x)length(unique(x))) # Number of values for each variable
length(ValsOfVars[ValsOfVars<2]) # there are no meaningless variables in data (with only 1 value)
```

```
## [1] 0
```

There are 53 variables (52 predictors) in training (19622 rows) and testing (20 questions of the final test) data sets.

Modeling

Slicing training data set to training and testing sets to estimate the quality (accuracy) of the models by cross-validation.

```
set.seed(1435)
inTrain <- createDataPartition(y=trnSet$classe, p = 0.60, list=FALSE)
trnSet_trn <- trnSet[inTrain,] # training set (to biuld the models)
trnSet_tst <- trnSet[-inTrain,] # testing set for cross-validation (to estimate error)
```

Let's fit the models (using `trnSet_trn`). We'll take into account the following models: * random forest, * decision tree, * stochastic gradient boosting, * linear discriminant analysis, * support vector machine.

```
set.seed(125195)
model_RF <- randomForest(classe ~ ., data=trnSet_trn, importance = TRUE)
model_DT <- rpart(classe ~ ., data=trnSet_trn, method="class")
model_gbm <- train(classe ~ ., data=trnSet_trn, method="gbm")
model_lda <- train(classe ~ ., data=trnSet_trn, method="lda")
model_svm <- svm(classe ~ ., data=trnSet_trn)
```

Accuracy of the models

To estimate out of sample error (models were built on `trnSet_trn`) we'll use `trnSet_tst`.

Random forest model testing results

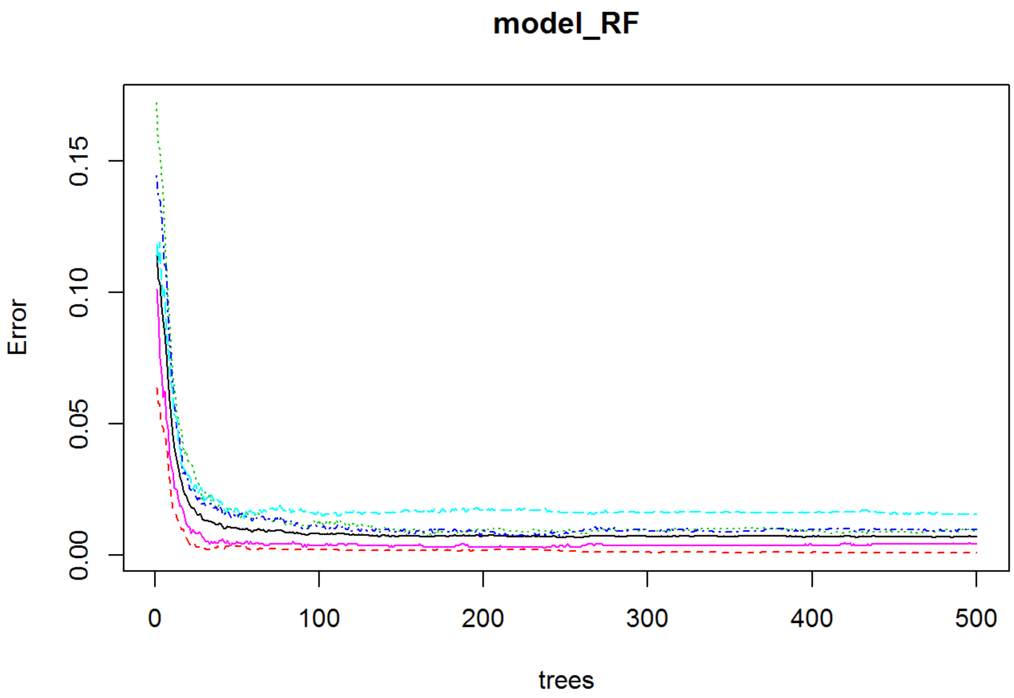
```
prediction_RF <- predict(model_RF, trnSet_tst)
ConMx <- confusionMatrix(prediction_RF, trnSet_tst$classe)
print(ConMx)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2232   12    0    0    0
##          B    0 1503    4    0    0
##          C    0    3 1362   24    1
##          D    0    0    2 1258    4
##          E    0    0    0    4 1437
## 
## Overall Statistics
##                                          
##                Accuracy : 0.9931         
##                  95% CI : (0.991, 0.9948)
##     No Information Rate : 0.2845         
##     P-Value [Acc > NIR] : < 2.2e-16      
##                                          
##                   Kappa : 0.9913         
##  Mcnemar's Test P-Value : NA             
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9901   0.9956   0.9782   0.9965
## Specificity            0.9979   0.9994   0.9957   0.9991   0.9994
## Pos Pred Value         0.9947   0.9973   0.9799   0.9953   0.9972
## Neg Pred Value         1.0000   0.9976   0.9991   0.9957   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1916   0.1736   0.1603   0.1832
## Detection Prevalence   0.2860   0.1921   0.1772   0.1611   0.1837
## Balanced Accuracy      0.9989   0.9947   0.9956   0.9887   0.9980
```

```
head(apply(-varImp(model_RF),2,order))
```

```
##       A  B  C  D  E
## [1,]  3  2 39  3  1
## [2,] 39  1  1  1  3
## [3,]  1  3 38 39 39
## [4,] 38 39  3 38 41
## [5,] 41 41 41 41 38
## [6,]  2 38  2  2  2
```

```
plot(model_RF)
```



Decision tree model testing results

```
prediction_DT <- predict(model_DT, trnSet_tst, type="class")
ConMx2 <- confusionMatrix(prediction_DT, trnSet_tst$classe)
print(ConMx2)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1946  234   74   67   81
##          B   96  972  101   95  125
##          C   39  146  933   85   71
##          D   97   57  235  951  159
##          E   54  109   25   88 1006
## 
## Overall Statistics
##                                           
##                Accuracy : 0.7402          
##                  95% CI : (0.7304, 0.7499)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.6708          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8719   0.6403   0.6820   0.7395   0.6976
## Specificity            0.9188   0.9341   0.9474   0.9165   0.9569
## Pos Pred Value         0.8102   0.6998   0.7323   0.6344   0.7847
## Neg Pred Value         0.9475   0.9154   0.9338   0.9472   0.9336
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2480   0.1239   0.1189   0.1212   0.1282
## Detection Prevalence   0.3061   0.1770   0.1624   0.1911   0.1634
## Balanced Accuracy      0.8953   0.7872   0.8147   0.8280   0.8273
```

```
order(-varImp(model_DT))
```

```
##  [1] 22 15 23 16 28  5 20 18 24 26 14  3 25 12 27 19 10  1 13  6  9 21  4
## [24]  2 11 17  8  7 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52
```

Stochastic Gradient Boosting model testing results

```
prediction_gbm <- predict(model_gbm, trnSet_tst)
ConMx3 <- confusionMatrix(prediction_gbm, trnSet_tst$classe)
print(ConMx3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2189   69    0    1    3
##          B   27 1392   21    0   18
##          C    9   54 1330   51   12
##          D    7    2   14 1226   25
##          E    0    1    3    8 1384
##
## Overall Statistics
##
##                Accuracy : 0.9586
##                  95% CI : (0.9539, 0.9629)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9476
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9807   0.9170   0.9722   0.9533   0.9598
## Specificity            0.9870   0.9896   0.9805   0.9927   0.9981
## Pos Pred Value         0.9677   0.9547   0.9135   0.9623   0.9914
## Neg Pred Value         0.9923   0.9803   0.9941   0.9909   0.9910
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2790   0.1774   0.1695   0.1563   0.1764
## Detection Prevalence   0.2883   0.1858   0.1856   0.1624   0.1779
## Balanced Accuracy      0.9839   0.9533   0.9764   0.9730   0.9790
```

Linear Discriminant Analysis model testing results

```
prediction_lda <- predict(model_lda, trnSet_tst)
ConMx4 <- confusionMatrix(prediction_lda, trnSet_tst$classe)
print(ConMx4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1841  244  137   85   66
##          B   57  972  123   46  269
##          C  159  171  879  153  121
##          D  170   56  187  952  139
##          E    5   75   42   50  847
##
## Overall Statistics
##
##                Accuracy : 0.6998
##                  95% CI : (0.6896, 0.71)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6198
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8248   0.6403   0.6425   0.7403   0.5874
## Specificity            0.9052   0.9218   0.9068   0.9159   0.9731
## Pos Pred Value         0.7758   0.6626   0.5927   0.6330   0.8312
## Neg Pred Value         0.9286   0.9144   0.9231   0.9473   0.9128
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2346   0.1239   0.1120   0.1213   0.1080
## Detection Prevalence   0.3024   0.1870   0.1890   0.1917   0.1299
## Balanced Accuracy      0.8650   0.7810   0.7747   0.8281   0.7803
```

Support vector machine model testing results

```
prediction_svm <- predict(model_svm, trnSet_tst)
ConMx5 <- confusionMatrix(prediction_svm, trnSet_tst$classe)
print(ConMx5)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2224  134    2    2    3
##          B    3 1317   44   12   15
##          C    5   64 1282  114   48
##          D    0    3   37 1154   29
##          E    0    0    3    4 1347
##
## Overall Statistics
##
##                Accuracy : 0.9335
##                  95% CI : (0.9277, 0.9389)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9157
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.8676   0.9371   0.8974   0.9341
## Specificity            0.9749   0.9883   0.9643   0.9895   0.9989
## Pos Pred Value         0.9404   0.9468   0.8473   0.9436   0.9948
## Neg Pred Value         0.9985   0.9689   0.9864   0.9801   0.9854
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2835   0.1679   0.1634   0.1471   0.1717
## Detection Prevalence   0.3014   0.1773   0.1928   0.1559   0.1726
## Balanced Accuracy      0.9856   0.9279   0.9507   0.9434   0.9665
```

Conclusions

In out of sample error the RF model demonstrates the highest accuracy (0.99). Also high accuracy was in gbm and svm models (0.96 and 0.93). Lowest accuracy was in lda (0.7) and rpart (0.74)

Final test predictions

```
prediction <- rbind(predict(model_RF, tstSet),predict(model_DT, tstSet,type="class"),
                    predict(model_gbm, tstSet),predict(model_lda, tstSet),
                    predict(model_svm, tstSet))
Pred_Fin <- as.data.frame(x=matrix(c("A","B","C","D","E")[prediction], ncol = 20, nrow = 5),
                          row.names = c("RF","DT","gbm","lda","svm"))
print(Pred_Fin)
```

```
##     V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20
## RF   B  A  B  A  A  E  D  B  A   A   B   C   B   A   E   E   A   B   B   B
## DT   A  A  A  D  A  C  D  E  A   A   B   C   B   A   D   E   A   B   A   B
## gbm  B  A  B  A  A  E  D  B  A   A   B   C   B   A   E   E   A   B   B   B
## lda  B  A  B  C  C  E  D  D  A   A   D   A   B   A   E   A   A   B   B   B
## svm  B  A  A  A  A  E  D  B  A   A   B   C   B   A   E   E   A   B   B   B
```

For final quiz we'll use the results of the random forest model which demonstrate the best accuracy (0.99).