Practical Machine Learning - Assignment

Submitted by Anne Bricis

Executive Summary

A study was undertaken on a set of measurements recorded during an exercise routine. The objective of the study was to see if the measurements taken could be used to predict the manner in which the exercise had been performed.

Data preparation

The dataset contains measurements for 6 individuals who had fitted devices on the arm, forearm, belt and dumbbell. The measurements were captured while the individual performed a set of 10 repetitions of an arm curl. The arm curl was performed in 5 different ways: A=correct method, B=elbow to front, C=lift dumbbell halfway, D=lower dumbbell halfway and E=hips to front.

After downloading, the training dataset was analysed (Appendix 1). This contained 160 variables (including the outcome), a number of which had a majority of missing values A subset of the data set was created using only the columns that were considered relevant, ie excluding columns with missing values as well as date columns. The reduced data set contains only 54 variables (including the outcome) which was considered to be more manageable. This reduced set was further split into two, so that local training and testing sets could be applied.

Model selection

A number of models were developed and analysed.

Model 1 - Basic reduced variable model

First a model was fitted that used all the variables in the reduced data set as presented (Appendix 2). However this resulted in a poor model which did not even provide for the outcome "D".

Model 2 - Preprocessing with principal components analysis

Principal components analysis was then applied over all the variables in the reduced set except the user and outcome (Appendix 3) and this produced better results in that outcome "D" was considered, but the error rate was very high in that over half of the outcomes on the training data set were predicted incorrectly.

Plotting the results of the PCA however identified interesting patterns in the data. When plotted by user, there were four distinct groups with a single user in each, plus a fifth group which overlapped two users. When plotted by outcome, it was very hard to distinguish any difference between the correct method and any of the "incorrect" ones.

Model 3 - Model by user

Investigating the data further, attempts to model by individual user were unsuccessful as the resulting models were individually worse than that provided by Model 2.

Cross validation

The validation data set was created using 30% of the original training data, which was set aside to test the model. The results of the model using the training data were poor, so there was no expectation of much improvement with the validation set.

Expected "out of sample" error

With the "best" model found so far (Model 2), the value for in-sample-errors is $9482/19622 = 0.48$ and the value for out-of-sample-error is $2999/4904 = 0.61$.

Results and conclusion

A suitable model was not found as part of this analysis, however by chance the test results (x20) provided a higher success rate than then training and local testing. Further analysis is required to improve the model.

Reference

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Appendices

Appendix 1 - Data source

The data files were downloaded from the corresponding Groupware website (http://groupware.les.inf.puc-rio.br/har) as the "Weight Lifting Exercise Dataset". The training and test data were downloaded using the following r commands:

```
##train<-download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "train.c
##test<-download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "test.csv")
```

The training data file was reduced to remove superfluous variables (dates, mainly missing values) and further split into two for training and testing.

```
trainAll<-read.csv("train.csv")
testAll<-read.csv("test.csv")
trainSmall<-trainAll[,-c(1,3:7,12:36,50:59,69:83,87:101, 103:112,125:139,141:150)]
names(trainSmall)
```

```
##  [1] "user_name"            "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
```

```
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.2
```

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

```
## Warning: package 'ggplot2' was built under R version 3.2.1
```

```r
inTrain<-createDataPartition(y=trainSmall$classe, p=0.75, list=FALSE)
train<-trainSmall[inTrain,]
test<-trainSmall[-inTrain,]
```

Appendix 2 - Model 1 analysis - reduce data set

```r
modFitRed<-train(classe~.,method="rpart",data=train)
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 3.2.1
```

```r
print(modFitRed$finalModel)
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 129.5 13370  9240 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1190     7 A (0.99 0.0059 0 0 0) *
##      5) pitch_forearm>=-33.95 12180  9233 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 436.5 10266  7380 A (0.28 0.18 0.24 0.19 0.1)
##         20) roll_forearm< 123.5 6369  3786 A (0.41 0.18 0.18 0.17 0.056) *
##         21) roll_forearm>=123.5 3897  2573 C (0.078 0.18 0.34 0.22 0.18) *
##       11) magnet_dumbbell_y>=436.5 1914   939 B (0.032 0.51 0.043 0.23 0.18) *
##    3) roll_belt>=129.5 1348    55 E (0.041 0 0 0 0.96) *
```

```r
confusionMatrix(train$classe, predict(modFitRed,train))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3766   61  303    0   55
##          B 1178  975  695    0    0
##          C 1161   82 1324    0    0
##          D 1096  446  870    0    0
##          E  358  350  705    0 1293
##
## Overall Statistics
##
##                Accuracy : 0.4999
##                  95% CI : (0.4918, 0.508)
##     No Information Rate : 0.5136
##     P-Value [Acc > NIR] : 0.9996
##
##                   Kappa : 0.347
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4982  0.50940  0.33975       NA  0.95920
## Specificity            0.9415  0.85372  0.88513   0.8361  0.89432
## Pos Pred Value         0.8999  0.34235  0.51578       NA  0.47783
## Neg Pred Value         0.6399  0.92089  0.78825       NA  0.99542
## Prevalence             0.5136  0.13004  0.26478   0.0000  0.09159
## Detection Rate         0.2559  0.06625  0.08996   0.0000  0.08785
## Detection Prevalence   0.2843  0.19350  0.17441   0.1639  0.18386
## Balanced Accuracy      0.7198  0.68156  0.61244       NA  0.92676
```

Appendix 3 - Model 2 analysis - PCA

```r
preProc<-preProcess(train[-c(1,54)], method="pca",pcaComp=2)
trainPCA<-predict(preProc, train[-c(1,54)])
modFitPCA<-train(train$classe~.,method="rpart",data=trainPCA)
print(modFitPCA$finalModel)
```

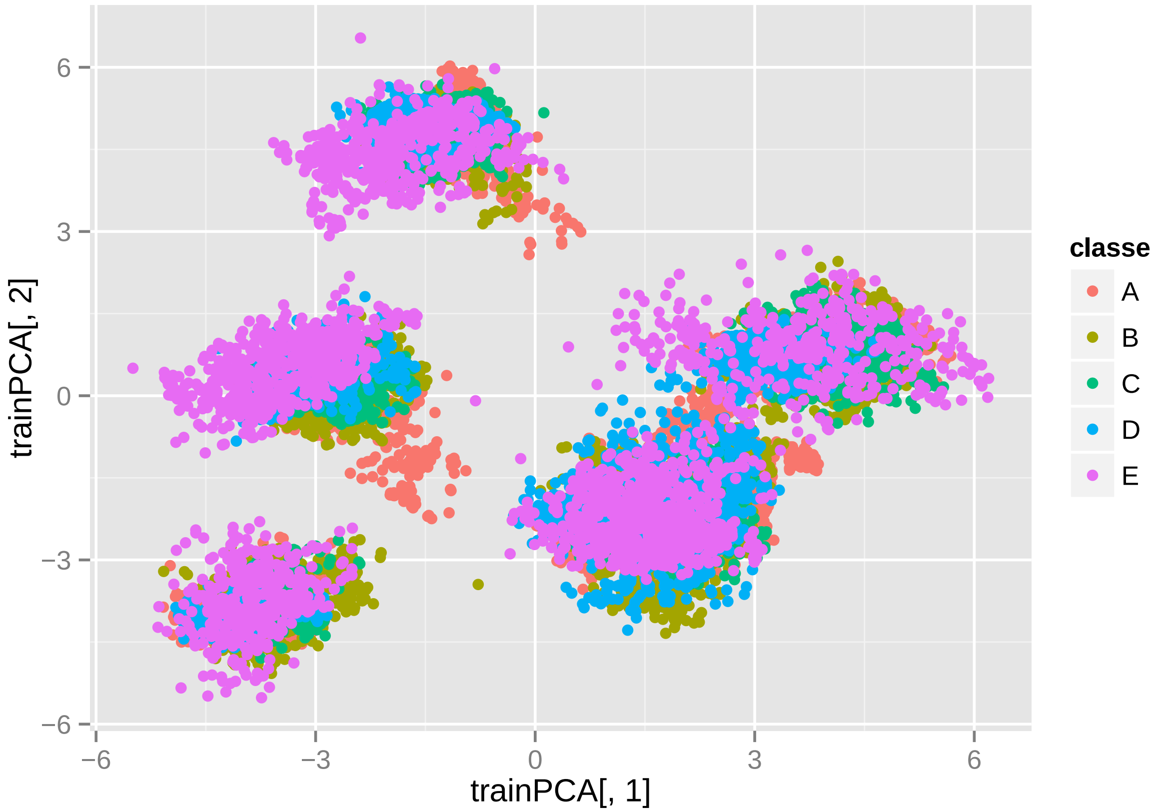
```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##     2) PC1>=5.03196 675   107 A (0.84 0.031 0.028 0 0.099) *
##     3) PC1< 5.03196 14043 10426 A (0.26 0.2 0.18 0.17 0.19)
##       6) PC1>=-3.759612 13013  9575 A (0.26 0.21 0.19 0.16 0.17)
##        12) PC1>=3.865412 1013   668 C (0.28 0.21 0.34 0.045 0.13)
```

```
##           24) PC2>=1.695108 165    36 A (0.78 0.073 0.067 0 0.079) *
##           25) PC2< 1.695108 848   514 C (0.18 0.23 0.39 0.054 0.14) *
##         13) PC1< 3.865412 12000  8845 A (0.26 0.21 0.18 0.17 0.18)
##           26) PC1>=2.184014 2372  1567 A (0.34 0.15 0.15 0.23 0.13)
##             52) PC2< 0.03382963 1486   829 A (0.44 0.16 0.16 0.14 0.096) *
##             53) PC2>=0.03382963 886   553 D (0.17 0.13 0.14 0.38 0.18) *
##           27) PC1< 2.184014 9628  7278 A (0.24 0.22 0.19 0.16 0.19)
##             54) PC1< -0.3269417 6459  4642 A (0.28 0.21 0.2 0.14 0.16)
##              108) PC1>=-1.184601 1098   557 A (0.49 0.17 0.22 0.045 0.075) *
##              109) PC1< -1.184601 5361  4085 A (0.24 0.22 0.2 0.16 0.18)
##                218) PC2< 0.346031 2534  1671 A (0.34 0.24 0.23 0.096 0.091) *
##                219) PC2>=0.346031 2827  2111 E (0.15 0.21 0.17 0.22 0.25)
##                  438) PC1>=-2.386565 1832  1341 B (0.17 0.27 0.19 0.21 0.16) *
##                  439) PC1< -2.386565 995   580 E (0.099 0.09 0.14 0.25 0.42) *
##             55) PC1>=-0.3269417 3169  2383 E (0.17 0.24 0.16 0.19 0.25)
##              110) PC2< -3.14511 256    99 B (0.078 0.61 0.039 0.25 0.023) *
##              111) PC2>=-3.14511 2913  2133 E (0.18 0.21 0.17 0.18 0.27) *
##        7) PC1< -3.759612 1030   639 E (0.17 0.13 0.024 0.3 0.38)
##         14) PC2< -3.008543 771   496 D (0.21 0.16 0.026 0.36 0.25) *
##         15) PC2>=-3.008543 259    58 E (0.073 0.012 0.019 0.12 0.78) *
```

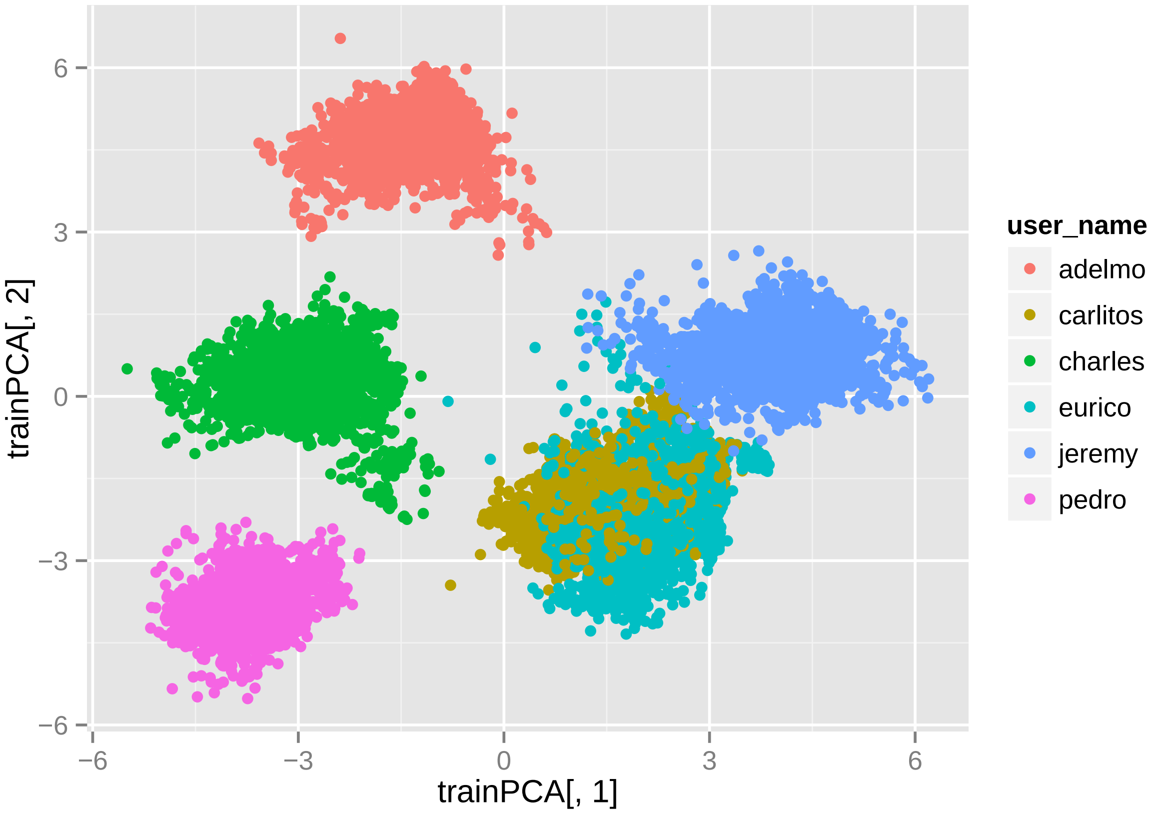
```
confusionMatrix(train$classe, predict(modFitPCA,trainPCA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2758  334  154  308  631
##          B 1062  648  199  245  694
##          C 1099  355  334  143  636
##          D  504  444   46  608  810
##          E  535  307  115  353 1396
##
## Overall Statistics
##
##                Accuracy : 0.3903
##                  95% CI : (0.3824, 0.3982)
##     No Information Rate : 0.4048
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.2152
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4629  0.31034  0.39387  0.36693  0.33501
## Specificity            0.8371  0.82581  0.83901  0.86188  0.87584
## Pos Pred Value         0.6590  0.22753  0.13011  0.25207  0.51589
## Neg Pred Value         0.6962  0.87869  0.95770  0.91476  0.76931
## Prevalence             0.4048  0.14187  0.05762  0.11258  0.28312
## Detection Rate         0.1874  0.04403  0.02269  0.04131  0.09485
## Detection Prevalence   0.2843  0.19350  0.17441  0.16388  0.18386
## Balanced Accuracy      0.6500  0.56808  0.61644  0.61440  0.60543
```

```
qplot(trainPCA[,1],trainPCA[,2],colour=classe,data=train)
```




```
qplot(trainPCA[,1],trainPCA[,2],colour=user_name,data=train)
```




```
predPCA<-predict(modFitPCA,trainPCA)
comparePCA<-data.frame("orig"=train$classe,"pred"=predPCA)
diffPCA<-comparePCA[comparePCA$orig!=comparePCA$pred,]
dim(diffPCA)
```

```
## [1] 8974    2
```

Appendix 4 - Cross validation

```
testPCA<-predict(preProc,test[-c(1,54)])
confusionMatrix(test$classe, predict(modFitPCA,testPCA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 887 134  52 113 209
##          B 367 195  61  85 241
##          C 369 116  95  61 214
##          D 154 138  22 224 266
##          E 191  84  40 138 448
##
## Overall Statistics
```

```
##
##                Accuracy : 0.377
##                  95% CI : (0.3634, 0.3908)
##     No Information Rate : 0.4013
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.1988
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4507  0.29235  0.35185  0.36071  0.32511
## Specificity            0.8270  0.82204  0.83599  0.86458  0.87153
## Pos Pred Value         0.6358  0.20548  0.11111  0.27861  0.49723
## Neg Pred Value         0.6919  0.88066  0.95678  0.90317  0.76767
## Prevalence             0.4013  0.13601  0.05506  0.12663  0.28100
## Detection Rate         0.1809  0.03976  0.01937  0.04568  0.09135
## Detection Prevalence   0.2845  0.19352  0.17435  0.16395  0.18373
## Balanced Accuracy      0.6388  0.55720  0.59392  0.61264  0.59832
```

```r
predPCATest<-predict(modFitPCA,testPCA)
comparePCATest<-data.frame("orig"=test$classe,"pred"=predPCATest)
diffPCATest<-comparePCATest[comparePCATest$orig!=comparePCATest$pred,]
dim(diffPCATest)
```

```
## [1] 3055    2
```

Appendix 5 - Test results

```r
testSmall<-testAll[,-c(1,3:7,12:36,50:59,69:83,87:101, 103:112,125:139,141:150)]
testPCASmall<-predict(preProc,testSmall[-c(1,54)])
answers=predict(modFitPCA,testPCASmall)

pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(answers)

answers
```

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