Weight Lift Exercise Quality Prediction

Jozef M H Dassen

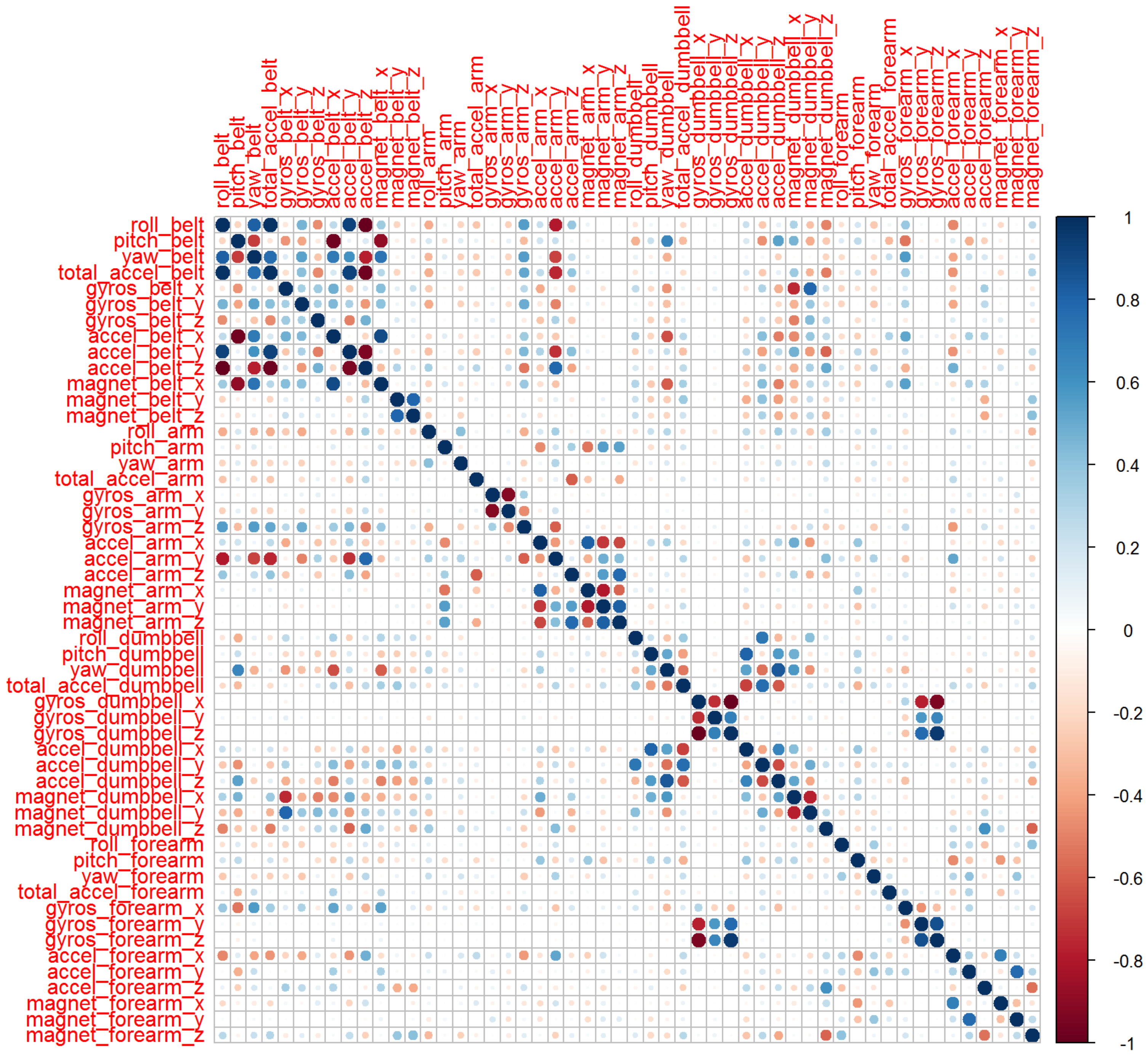
Sunday, May 17, 2015

This is a Machine Learning project attempting to decide the proper way of weight lifting exercises from a set of body movement sensors.

Data Set

The data set consists of processed sensor measurements. The processing was done using sliding windows on time series measurements of sensors. See http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (section on the Weight Lifting Exercise) for details. Each data row contains 152 measurements and is classified to a CLASS A,B,C,D or E indicating the quality of the exercise execution. Many of the columns however contain NA or DIV/0 values. These are removed. This will leave us with 52 usable measurements. No doubt some of these will be correlated since we use different sensor types for measuring the same movements.

We calculate the correlation matrix for the 52 measurements and visualize it with corrplot.



We see correlations between x,y,z measurements of the same sensor, but also between different sensors on the same measuring point. We will use these correlations to attempt to reduce the number of predictors in the ML implementation.

We also have a look at the nearZero Variance of the data to see if there are any measurements that would not contribute much to the solution.

```
nearZeroVar(train, saveMetrics=TRUE)
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.018868    7.94933392   FALSE FALSE
## pitch_belt            1.043796   12.29526097   FALSE FALSE
## yaw_belt              1.055710   13.01594235   FALSE FALSE
## total_accel_belt      1.057173    0.21110868   FALSE FALSE
## gyros_belt_x          1.085624    0.96090850   FALSE FALSE
## gyros_belt_y          1.155125    0.47317464   FALSE FALSE
## gyros_belt_z          1.093496    1.18657640   FALSE FALSE
## accel_belt_x          1.043165    1.15745796   FALSE FALSE
## accel_belt_y          1.138837    1.01186576   FALSE FALSE
## accel_belt_z          1.100000    2.10380724   FALSE FALSE
## magnet_belt_x         1.075099    2.20572177   FALSE FALSE
## magnet_belt_y         1.149660    2.09652763   FALSE FALSE
## magnet_belt_z         1.024465    3.13751183   FALSE FALSE
## roll_arm             45.673077   17.49290238   FALSE FALSE
## pitch_arm            72.000000   20.39018709   FALSE FALSE
## yaw_arm              32.534247   19.31280483   FALSE FALSE
## total_accel_arm       1.017350    0.46589503   FALSE FALSE
## gyros_arm_x           1.008242    4.58615418   FALSE FALSE
## gyros_arm_y           1.491573    2.70801485   FALSE FALSE
## gyros_arm_z           1.072000    1.67431026   FALSE FALSE
## accel_arm_x           1.095652    5.58346073   FALSE FALSE
## accel_arm_y           1.193333    3.84363398   FALSE FALSE
## accel_arm_z           1.076923    5.57618112   FALSE FALSE
## magnet_arm_x          1.000000    9.57268690   FALSE FALSE
## magnet_arm_y          1.076923    6.20222756   FALSE FALSE
## magnet_arm_z          1.092105    9.14318993   FALSE FALSE
## roll_dumbbell         1.085106   86.57639950   FALSE FALSE
## pitch_dumbbell        2.382979   84.29060202   FALSE FALSE
## yaw_dumbbell          1.068182   86.04498799   FALSE FALSE
## total_accel_dumbbell  1.127409    0.31302322   FALSE FALSE
## gyros_dumbbell_x      1.034965    1.68886948   FALSE FALSE
## gyros_dumbbell_y      1.184652    1.96549465   FALSE FALSE
## gyros_dumbbell_z      1.041763    1.42680352   FALSE FALSE
## accel_dumbbell_x      1.013393    3.02103807   FALSE FALSE
## accel_dumbbell_y      1.029070    3.29766325   FALSE FALSE
## accel_dumbbell_z      1.229814    2.85360705   FALSE FALSE
## magnet_dumbbell_x     1.126050    7.78918250   FALSE FALSE
## magnet_dumbbell_y     1.154472    5.99839849   FALSE FALSE
## magnet_dumbbell_z     1.146341    4.82638131   FALSE FALSE
## roll_forearm         11.639485   13.67838684   FALSE FALSE
## pitch_forearm        66.146341   19.27640678   FALSE FALSE
## yaw_forearm          15.761628   12.86307054   FALSE FALSE
## total_accel_forearm   1.161476    0.49501347   FALSE FALSE
## gyros_forearm_x       1.094183    2.03829075   FALSE FALSE
## gyros_forearm_y       1.014652    5.26315789   FALSE FALSE
## gyros_forearm_z       1.155425    2.14020528   FALSE FALSE
## accel_forearm_x       1.123077    5.66353643   FALSE FALSE
## accel_forearm_y       1.025974    7.06122152   FALSE FALSE
## accel_forearm_z       1.119266    4.10569993   FALSE FALSE
## magnet_forearm_x      1.017857   10.61367111   FALSE FALSE
## magnet_forearm_y      1.031250   13.27072869   FALSE FALSE
## magnet_forearm_z      1.022727   11.69105336   FALSE FALSE
## classe                1.469526    0.03639805   FALSE FALSE
```

It turns out that all the variables have sufficently large variation in measurement to be usable.

We now look at which columns can be removed from the predictors based on the correlation matrix. We use the findCorrelation function with 75% as the cut off point. This means we find predictors that are more than 75% correlated with other predictors already in use.

```
findCorrelation(corMatrix, cutoff=0.75)
```

```
##  [1] 10  1  9  4 36  8  2 37 35 38 21 34 23 25 12 48 19 46 45 31
```

```
names(train[findCorrelation(corMatrix, cutoff=0.75)])
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "accel_dumbbell_z"  "accel_belt_x"
##  [7] "pitch_belt"        "magnet_dumbbell_x" "accel_dumbbell_y"
## [10] "magnet_dumbbell_y" "accel_arm_x"       "accel_dumbbell_x"
## [13] "accel_arm_z"       "magnet_arm_y"      "magnet_belt_y"
## [16] "accel_forearm_y"   "gyros_arm_y"       "gyros_forearm_z"
## [19] "gyros_forearm_y"   "gyros_dumbbell_x"
```

Model selection

The problem is a multi level classification problem. Linear or logistic regression is not suitable. A decison tree algorithm seems more suitable and we choose the random forest algorithm. Random forest is reputed to be robust and recommended as a first cut (see https://www.kaggle.com/wiki/RandomForests (https://www.kaggle.com/wiki/RandomForests)).

The main issue we are facing that needs to be decided is which predictors to choose. We have total of 52 predictors but there are correlations between them as shown by the Correlation matrix. We will attempt to reduce the number of predictors. We will remove the highly correlated predictors found above from the predictors. But we will, as an alternative, also use PCA to reduce the predictors. To investigate the effect of the various methods we will run four models:

```
1 Remove High Correlation       we remove the highly correlated predictors
2 PCA with threshold 0.80       we use PCA to find predictors keeping 80% of the variance.
3 PCA with threshold 0.95       we use PCA to find predictors keeping 95% of the variance.
4 Use all predictors            we use all 52 predictors
```

We will evaluate accuracy and training time for each of the 4 cases.

Cross Validation and Accuracy

In order to determine the Out of Sample error rate of the trained model we need to have a dataset independent of the training data. Therefore we split the original data set into a training data set and a cross validation data set as follows:

```
inTrain <- createDataPartition(y=training$classe,p=0.7, list=FALSE)
train    <- training[inTrain,]
crossval <- training[-inTrain,]
```

We also notice that the range of values for the different predictors can be different by factor of up to 10,000. Therefore we preprocess each data row and scale to center the values (using mean and standard deviation). This preprocessing will be applied to both training and cross validation data sets (and later the test data set as well).

The *train* dataset will be used for training the model and the *crossval* will be used for predicting outcome using the trained model. We then compare the predicted outcome with the actual outcome in a confusion matrix as follows:

```
preProc <- preProcess(train[-lastCol], method = c("scale","center"))
trainPC <- predict(preProc, train1[-lastCol])
crossvalPC <- predict(preProc, crossval[-lastCol])
modelFit <- train(train$classe ~ .,method="rf",data=trainPC)
CM <- confusionMatrix(predict(modelFit,newdata=crossvalPC),crossval$classe)
```

The confusion matrix will give us the Accuracy of the prediction of the outcome on the cross validation data set by the model trained.

As explained above we run 4 different models. Two of the models will use PCA, principle components as predictors. In those cases the preprocessing changes to:

```
preProc <- preProcess(train[-lastCol], method = c("scale","center","pca"), thresh = 0.95)
```

Where *thresh* is the percentage of retained variance required.

We give the full code for the first model, which removes the highly correlated predictors, below. The model and Confusion matrix output are shown.

```
train1 <- train[,-highCorr]
cv1 <- crossval[,-highCorr]
lastCol <- dim(train1)[2]

preProc1 <- preProcess(train1[-lastCol], method = c("scale","center"))
trainPC1 <- predict(preProc1, train1[-lastCol])
cvPC1 <- predict(preProc1, cv1[-lastCol])
tic1=proc.time()[3]
modelFit1 <- train(train1$classe ~ .,method="rf",data=trainPC1)
toc1=proc.time()[3] - tic1
CM1 <- confusionMatrix(predict(modelFit1,newdata=cvPC1),cv1$classe)
modelFit1
CM1
```

```
## Random Forest
##
## 13737 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9861119  0.9824375  0.001544131  0.001946190
##   16    0.9848733  0.9808723  0.001123780  0.001414139
##   31    0.9755716  0.9691116  0.003222698  0.004064923
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1133    2    0    0
##          C    0    2 1022    9    0
##          D    0    0    2  955    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9961   0.9907   0.9991
## Specificity            0.9991   0.9996   0.9977   0.9994   1.0000
## Pos Pred Value         0.9976   0.9982   0.9894   0.9969   1.0000
## Neg Pred Value         1.0000   0.9987   0.9992   0.9982   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1737   0.1623   0.1837
## Detection Prevalence   0.2851   0.1929   0.1755   0.1628   0.1837
## Balanced Accuracy      0.9995   0.9972   0.9969   0.9950   0.9995
```

We see that the model uses 31 predictors and the resulting Accuracy is 0.99. We timed the execution using tic/toc variables and find that this model need 2205 seconds to train.

As explained we train 3 additional models, two using PCA and one with the full set of predictors. The result is shown in the table below.

Case	Predictors	Accuracy	Elapsed Time
Remove High Correlation	31	.99	2204.7
PCA with threshold 0.80	12	.96	1082.2
PCA with threshold 0.95	25	.98	1967.8
Use all predictors	52	.99	3446.8

It appears that removing the highly correlated predictors gives the same accuracy as using all predictors. Execution time improves by 36%, which roughly corresponds to the reduction in the number of predictors (40%). The Accuracy of the PCA runs depends on the threshold requested. A threshold value of 95% gives good accuracy but not as good as the models with high correlation predictors removed. The model with high correlation predictors removed seems to be the best compromise. We get a good reduction in computation time without losing accuracy.

Prediction of Test cases

We will now proceed to predict the outcome of the test data set using the trained models. The test data set needs to be prepared by applying the same preprocessing as was used in the training. We need to take care obviously to apply the correct preprocessing (since we have 4 models). We show the case for our preferred model, the High-Correlation-Removed model:

```
test    <- testing[,-highCorr]
lastCol <- dim(train)[2]
testPC  <- predict(preProc, test[-lastCol])
predict <- predict(modelFit, newdata=testPC)
```

Applying the same process for each of the other 3 models, we get the predictions as shown in the table below. Only the model with PCA at threshold 80% shows different predictions for test sample 3 and 11.

```
##    Remove High Corr. PCA threshold 0.80 PCA threshold 0.95 All predictors
## 1                 B                  B                  B              B
## 2                 A                  A                  A              A
## 3                 B                  A                  B              B
## 4                 A                  A                  A              A
## 5                 A                  A                  A              A
## 6                 E                  E                  E              E
## 7                 D                  D                  D              D
## 8                 B                  B                  B              B
## 9                 A                  A                  A              A
## 10                A                  A                  A              A
## 11                B                  A                  B              B
## 12                C                  C                  C              C
## 13                B                  B                  B              B
## 14                A                  A                  A              A
## 15                E                  E                  E              E
## 16                E                  E                  E              E
## 17                A                  A                  A              A
## 18                B                  B                  B              B
## 19                B                  B                  B              B
## 20                B                  B                  B              B
```

Conclusion

Random forest is an easy ML model to use in the given case. Training time is the only issue to be addressed by attempting to reduce the number of predictors used in the model. Simply removing highly correlated predictors is as effective, if not more so, than using PCA with high variance retention. The trained model works well on the cross validation set.