Executive summary

In this project an algorithm is made to predict if a barbell lift is performed correctly. Participants perform the lift in in one correct way and 5 different incorrect ways while being monitored by accelerometers on their belt, forearm, arm and dumbell. The data is used to train a random forest algorithm with PCA input components. The algorithm based on X input compontents and Y trees was shown to be most accurate with Z% of classes identified correctly.

Introduction

A random forest algorithm is used to predict the way a barbell lift is performed (correctly or either of the 5 incorrect ways). As random forests are prune to overfitting a test set is used to evaluate the performance of the random forest. To reduce the effect of overfitting an PCA is used on the input variables. The random forest algorithm is tested on 20, 23, 26 and 29 components (that cover 91% - 98% of the total variance) and with 64 and 128 trees (as suggested in https://www.researchgate.net/publication/230766603_How_Many_Trees_in_a_Random_Forest).

Data preparation and assumptions

Two datasets were loaded: pml-training.csv and pml-testing.csv.

```
df <- read.csv('pml-training.csv')
df_evaluate  <- read.csv('pml-testing.csv')
```

It was seen in pml-testing (renamed df_evaluate) that there were a lot of columns containing only NA data. These columns were filtered along with the context columns (column 1-7). Meanwhile the training data was split 70-30 in training data and test data.

```
col_names <- colnames(df_evaluate)
ind <- 1
indices <- c()
for (name in col_names){
    if (length(which(is.na(df_evaluate[name]))) == 20){
        indices <- c(indices, ind)

    }
    ind <- ind + 1
}

inTrain <- createDataPartition(y=df$classe, p=0.7, list=FALSE)
df_train <- df[inTrain, -indices]
df_test <- df[-inTrain, -indices]
df_evaluate <- df_evaluate[, -indices]

df_train <- df_train[, 8:dim(df_train)[2]]
df_test <- df_test[, 8:dim(df_test)[2]]
df_evaluate <- df_evaluate[, 8:dim(df_evaluate)[2]]
```

For the test data a PCA was done to determine the cumulative proportion of variance the PCA compontents contain. It can be seen that upwards of 23 components 95% of the variance in the data was explained.

```
df_train.pca <- prcomp(df_train[, 9:dim(df_train)[2]-1], center = TRUE, scale = TRUE)
summary(df_train.pca)$importance[2, ]
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
```

```
## 0.15890 0.11958 0.09540 0.08044 0.06378 0.04960 0.04474 0.04348 0.03744
##    PC10    PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18
## 0.03599 0.03093 0.02733 0.02405 0.02115 0.01688 0.01593 0.01511 0.01401
##    PC19    PC20    PC21    PC22    PC23    PC24    PC25    PC26    PC27
## 0.01096 0.01046 0.00923 0.00879 0.00727 0.00695 0.00666 0.00602 0.00565
##    PC28    PC29    PC30    PC31    PC32    PC33    PC34    PC35    PC36
## 0.00521 0.00435 0.00390 0.00322 0.00282 0.00264 0.00185 0.00155 0.00133
##    PC37    PC38    PC39    PC40    PC41    PC42    PC43    PC44    PC45
## 0.00124 0.00110 0.00082 0.00075 0.00069 0.00067 0.00053 0.00045 0.00015
```

It is known that random forest models are prune to overfitting. To control overfitting different numbers of PCA components and trees are tried in an iterative loop.

```
n_best <- 0
ntrees_best <- 0
ncomps_best <- 0
fit_best <- NULL
res <- NULL
for (ntrees in c(64, 128)){
    for (ncomps in seq(20, 30, by=3)) {
        pre_process <- preProcess(
            df_train[, -dim(df_train)[2]-1],
            method='pca',
            pcaComp=ncomps)
        train_pc <- predict(pre_process, df_train[, -dim(df_train)[2]])
        model_fit <- train(
            y=df_train[, dim(df_train)[2]],
            x= train_pc,
            method='rf',
            ntree=ntrees
            )

        test_pc <- predict(pre_process, df_test[, -dim(df_test)[2]])
        test_outcome <- predict(model_fit, newdata=test_pc)


        n_corr <- length(which(test_outcome == df_test[, dim(df_test)[2]]))

        res <- rbind(res, c(trees=ntrees, comps=ncomps, corr=n_corr))

        print(n_corr)
        if (n_corr > n_best){
            n_best <- n_corr
            print('best ncorr')
            ntrees_best <- ntrees
            ncomps_best <- ncomps
            fit_best <- model_fit
        }
    }
}

print(paste('The best fit is ', n_corr / dim(df_test)[1], '% correct with ', ntrees_best, ' trees and '
```

Appendix A evaluation results

The resulting algorithm is then used on the evaluation data to predict the quiz answers.

```
pre_process <- preProcess(
    df_train[, -dim(df_train)[2]-1],
    method='pca',
    pcaComp=ncomps_best)
evaluate_pc <- predict(pre_process, df_evaluate[, -dim(df_evaluate)[2]])
evaluate_outcome <- predict(fit_best, newdata=evaluate_pc)
print(evaluate_outcome)
```

```
v <- ggplot(df_res, aes(trees, comps, z = corr))
v <- v + geom_raster(aes(fill = corr))
# v <- v + geom_contour(colour='white')
print(v)
```