Practical Machine Learning

Perry Koorevaar

Friday, October 17, 2014

Introduction

In this assignment it is requested to predict the way in which people performed barbell lifts. These ways are classified into 5 categories A-E. The data to predict on consist of acceleration and movements measurements on both the body and the barbell.

Step 1: Exploring and cleaning the data

Both a training and a test set have been provided. By opening these files and visually inspecting them it becomes clear that several columns are irrelevant, either because they contain a lot of "NA's" or empty cells, or because they are obviously not relevant for predicting the outcome (e.g. the name of the test person is one of the variables). All these irrelevant columns are removed.

```
library(caret); library(kernlab);library(ggplot2); library(lattice)
```

```
## Warning: package 'caret' was built under R version 3.1.1
```

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

```
## Warning: package 'kernlab' was built under R version 3.1.1
```

```
traindata <- read.csv("pml-training.csv")
testdata <- read.csv("pml-testing.csv")
train1 <- traindata[colSums(is.na(traindata)) < 200]
train2 <- train1[colSums(train1=="") < 200]
c<-seq.int(8,60)
train3 <- train2[,c]
```

The column with the outcome, labelled "classe", is a character variable, but for my first modelling attempt I will try a glm, and therefore translate the character into a numeric value. In the second model, discussed below, I keep the classification nature of the classe variable and directly predict "characters / classes".

```
train3$classe <- as.numeric(train3$classe)
```

Identical clean up of columnss in the test set as for the training set

```
test1 <- testdata[colSums(is.na(testdata)) < 2]
test2 <- test1[colSums(test1=="") < 2]
c<-seq.int(8,60)
test3 <- test2[,c]
test3$problem_id <- as.numeric(test3$problem_id)
```

Step 2: Generalized linear model "glm"

First the training data is used with the "glm" model. Principal Component Analysis (PCA) pre-processing is performed, and for cross validation "repeated k-fold cross validation" is used with k=10 and 5 repeats. These parameters are chosen "arbitrarily" but seem to make sense given the size of the data sets.

```
modelFit <- train(train3$classe ~ ., method = "glm", preProcess = "pca",
                  data = train3,
                  trControl = trainControl(method = "repeatedcv", number=10, repeats=5,
                  preProcOptions = list(thresh = 0.8)))
print(modelFit)
```

```
## Generalized Linear Model
##
## 19622 samples
##    52 predictor
##
## Pre-processing: principal component signal extraction, scaled, centered
## Resampling: Cross-Validated (10 fold, repeated 5 times)
##
## Summary of sample sizes: 17660, 17659, 17659, 17661, 17661, 17659, ...
##
## Resampling results
##
##   RMSE  Rsquared  RMSE SD  Rsquared SD
##   1     0.2       0.06     0.04
##
##
```

```
summary(modelFit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.372  -0.974  -0.058   0.931   4.078
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.76929    0.00934  296.39  < 2e-16 ***
## PC1         -0.06881    0.00323  -21.29  < 2e-16 ***
## PC2         -0.02842    0.00328   -8.66  < 2e-16 ***
## PC3          0.14526    0.00432   33.62  < 2e-16 ***
## PC4         -0.04013    0.00460   -8.73  < 2e-16 ***
## PC5         -0.02681    0.00489   -5.48  4.2e-08 ***
## PC6         -0.10990    0.00539  -20.38  < 2e-16 ***
## PC7          0.06552    0.00624   10.50  < 2e-16 ***
## PC8          0.18648    0.00649   28.73  < 2e-16 ***
## PC9         -0.14640    0.00713  -20.53  < 2e-16 ***
## PC10         0.18924    0.00761   24.88  < 2e-16 ***
```

```
## PC11         0.06433    0.00794    8.10  5.7e-16 ***
## PC12        -0.29232    0.00879  -33.25  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.713)
##
##     Null deviance: 42717  on 19621  degrees of freedom
## Residual deviance: 33591  on 19609  degrees of freedom
## AIC: 66262
##
## Number of Fisher Scoring iterations: 2
```

Next we make a prediction of the (cleaned) test dataset which is in "test3", and translate back the numeric prediction into a character in the sequence A-E:

```
voorspel <- round(predict(modelFit, test3),0)
voorspelchar <- chartr("12345","ABCDE",voorspel)
voorspelchar
```

```
##  [1] "C" "B" "B" "B" "B" "C" "D" "C" "A" "B" "B" "B" "C" "B" "D" "B" "B"
## [18] "C" "C" "C"
```

For every row in the test dataset we now have a prediction for the " classe", i.e. the way in which the excercises wwere performed.

Step3: Classification model "rpart"

The second model I try is a true classification model with the "rpart" method. The same pre-processing and cross validation is used as with the model described in Step2.

```
train4 <- train2[,c]
modelFit2 <- train(train4$classe ~ ., method = "rpart", preProcess = "pca",
                  data = train4,
                  trControl = trainControl(method = "repeatedcv", number=10, repeats=5,
                                           preProcOptions = list(thresh = 0.8)))
```

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

```
#
print(modelFit2)
```

```
## CART
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: principal component signal extraction, scaled, centered
## Resampling: Cross-Validated (10 fold, repeated 5 times)
```

```
##
## Summary of sample sizes: 17659, 17659, 17660, 17661, 17659, 17661, ...
##
## Resampling results across tuning parameters:
##
##   cp    Accuracy  Kappa  Accuracy SD  Kappa SD
##   0.04  0.4       0.15   2e-02        0.03
##   0.06  0.3       0.07   2e-02        0.04
##   0.12  0.3       0.00   1e-04        0.00
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03568.
```

```
#
voorspel2 <- predict(modelFit2, test3)
voorspel3 <- as.character(voorspel2)
voorspel3
```

```
##  [1] "A" "A" "A" "A" "B" "A" "E" "A" "A" "A" "A" "A" "B" "A" "E" "B" "A"
## [18] "B" "A" "B"
```

We now have a second prediction for the test set, independent from the first one.

Discussion of results

To judge the accuracy of the results I first made a comparison between predicted and observed values for the classe variable in the training sets. For these sets the outcomes are known and one can get a feel of the accuracy by simply counting the number of correctly predicted observations over the total observations.

```
# Overview accuracy glm  model
predtrain <- round(predict(modelFit, train3),0)
predtrainchar <- chartr("12345","ABCDE",predtrain)
table(predtrainchar, train3$classe)
```

```
##
## predtrainchar    1    2    3    4    5
##             A  792   20    2    0   20
##             B 2483 1019 1161  324  507
##             C 2089 2437 2209 1968 2043
##             D  216  321   50  911  982
##             E    0    0    0   13   55
```

```
#
# Overview accuracy rpart  model
predtrain2 <- predict(modelFit2, train4)
table(predtrain2, train4$classe)
```

```
##
## predtrain2    A    B    C    D    E
##          A 4204 1645 2177 1062  998
```

```
##           B 1183 1817 1200 1451 1491
##           C    0    0    0    0    0
##           D    0    0    0    0    0
##           E  193  335   45  703 1118
```

From this it can be calculated that the percentage of correctly predicted observations for the glm model = 25% and for thr rpart model = 36%. These results are, to my opinion, both poor, as totally random guessing would yield a succes rate of 1 in 5. Since the rpart model perfroms better I will use this as the primary model in submitting the answers. Also, since the in sample error rate for this model is already high at 64% (1-36%), the out of sample rate is expected to be even more.