Prediciting how well a fitness training has been done

by Andreas Eßbaumer 2017-01-07

This report is about the peer assignment of week 4 of the practical machine learning course. Goal is to build a model to predict the "classe" variable (which classifies how well the excercise has been done) by using the data provided.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Inital steps

First you need set the path to the csv.files and load required packages if necessary. Then the files are being imported. #

```
setwd("C:/Coursera/Practical//")
suppressPackageStartupMessages(library(nnet))

train_all <- read.table("pml-training.csv",sep=",", header=T)
test_all <- read.table("pml-testing.csv",sep=",", header=T)
```

The model

After some data exploration i have built the model. Because we want it to usee different training and test sets, a function is built taking a training-dataframe and a test-dataframe as an input. The function returns a vector of characters having the same number of rows as the test-dataframa. This vector gives the prediction of the classe-variable. # My approach is very basic and rather straight forward. I figured that the variable "user-name" may have some big impact. Therefore a different model has been trained for each of the participants making it 6 models in total. I removed all variables which had NA- or blank values and after that excluded the first seven variables because they seem to be additional information but not useful. Afterwards a multinomial log-linear model via neural networks has been performed. # Further information about the model creation can be found within in-code-comments. #

```
predictclass <- function(train,test){

    # The return vector
    predictions <- character(length=nrow(test))

    # The user_name is not used directly as a input variable
    # but more severly the data is partitioned by it,
    # resulting in a loop which creates 6 models
    train$user_name <- as.character(train$user_name)
    test$user_name <- as.character(test$user_name)

    for(nameit in seq(1,length(unique(train$user_name)))){
```

```
        # the partition for each user_name is done
        train1 <- train[train$user_name==unique(train$user_name)[nameit],]
        test1 <- test[test$user_name==unique(train$user_name)[nameit],]

        # All columns which have a NA or a blank value are identified
        a <- numeric(length = ncol(train1))

        for(i in seq(1,ncol(train1))){

            a[i] <- sum(is.na(train1[,i]) || trimws(train1[,i])=="")
        }

        b <- a==0

        # Those columns are discarded
        train1_cleaned <- train1[,b]
        test1_cleaned <- test1[,b[1:ncol(test1)]]
        # Furthermore the first 7 columns are discarded
        train1_final <- train1_cleaned[,-(1:7)]
        test1_final <- test1_cleaned[,-(1:7)]

        # The model is being trained
        model <- multinom(classe ~ ., data=train1_final, trace =F)

        # The predictions-vector is filled for this specific user
        predictions[test$user_name==unique(train$user_name)[nameit]] <- as.character(predict(model,newda



    }

    # After loooping through all users the vector is returned

    return(predictions)
}
```

Cross-Validation

To check how well this model performs and what out-of-sample-error can be expected a cross-validation has been performed using three disjunct samples from the training set. #

```
rownums <- 1:nrow(train_all)
set.seed(1111)
cv_set1 <- sort(sample(rownums,size = nrow(train_all)/3,replace = F))
set.seed(2222)
cv_set2 <- sort(sample(rownums[!(rownums %in% cv_set1)],size = nrow(train_all)/3,replace = F))
cv_set3 <- rownums[!(rownums %in% c(cv_set1,cv_set2))]
```

```
cv_errors <- numeric(length = 3)
```

Now the model needs to be done 3 times. Each time one of the cv-sets is the test data while the other two are the training data #

```
for(cv in 1:3){
    if(cv==1){
        predscv <- predictclass(train_all[c(cv_set1,cv_set2),],train_all[cv_set3,-ncol(train_all)])
        test_correct <- as.character(train_all[cv_set3,ncol(train_all)])
        cv_errors[cv] <- sum(predscv==test_correct)/length(test_correct)
    }
    if(cv==2){
        predscv <- predictclass(train_all[c(cv_set1,cv_set3),],train_all[cv_set2,-ncol(train_all)])
        test_correct <- as.character(train_all[cv_set2,ncol(train_all)])
        cv_errors[cv] <- sum(predscv==test_correct)/length(test_correct)
    }
    if(cv==3){
        predscv <- predictclass(train_all[c(cv_set3,cv_set2),],train_all[cv_set1,-ncol(train_all)])
        test_correct <- as.character(train_all[cv_set1,ncol(train_all)])
        cv_errors[cv] <- sum(predscv==test_correct)/length(test_correct)
    }
}
```

The vector cv_error measured the accuracy ("correct predictions"/"number of records") for the three runs. The mean of this vector gives an estimate how well the out-of-sample-error might be. #

```
mean(cv_errors)
```

```
## [1] 0.8682616
```

Predicting the test set

To predict the test set we only need to run the following code: #

```
preds <- predictclass(train_all,test_all)

print(preds)
```

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

With that easy model we already get 16 of 20 correct predictions. This is close to our estimate. # I hope you enjoyed reading through my model report. Thank you for reviewing and grading. Post your link to your own work in the comments and i´ll be please to review your work