Determining how well weight lifting exercises are performed via machine learning

Yang-Ming Zhu

Tuesday, April 21, 2015

This report is written to fulfill the project requirement of the Practical Machine Learning course offered by John Hopkins University on coursera.org.

Traditionally the machine learning or pattern recognition techniques are used to recognize the types of human activities, not how well or bad humans perform the activities. Velloso and collegues conducted the research to qualitatively assess the weight lifting execercies (see 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). They were generous to make the data avaliable at http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) and this project is based on their data.

The datasets can be loaded locally using the following R code

```
data <- read.csv("pml-training.csv")
testpa <- read.csv("pml-testing.csv")
```

While the detailed description of the data and its collection can be found from the authors' original paper, a short summary is provided here to given the context of this course project. The dataset contains 19622 observations and each observation consists of 160 variables. One of the variables (classe) is the classification label A, B, C, D, or E, which encodes how well the weight lifting is done: exactly according to the specification (A), throwing the elbows to the front (B), lifting the dumbbell only halfway (C), lowering the dumbbell only halfway (D) and throwing the hips to the front (E). A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. We were also given 20 observations with the classe field removed, and we were asked to predict the missing classes.

We did some simple exploratory analysis and preprocessing of the data using the following code

```
listNACols <- function(data) {
  nacols <- c()
  for (i in 1:ncol(data)) {
    if (sum(is.na(data[, i]))==nrow(data)) nacols <- c(nacols, i)
  }
  nacols
}
rmNACols <- function(data, col2rm) {
  for (i in length(col2rm):1) data[, col2rm[i]]<-NULL
  data
}
testNACols <- listNACols(testpa)
data2 = rmNACols(data, testNACols); testpa2 = rmNACols(testpa, testNACols)

data2 = subset(data2, data2$new_window=="no")
data2$new_window=NULL; testpa2$new_window=NULL

data2$X=NULL; testpa2$X=NULL
data2$raw_timestamp_part_1=NULL; testpa2$raw_timestamp_part_1=NULL
data2$raw_timestamp_part_2=NULL; testpa2$raw_timestamp_part_2=NULL
data2$cvtd_timestamp=NULL; testpa2$cvtd_timestamp=NULL
```

First note that in the test dataset there are 100 columns that contain NA in all observations. Since the entire columns are NA, it is not easy if not impossibe to impute the data. Hence those columns are removed from all datasets. After this simple step, both datasets don't contain NA. Further note that all test data with new_window=no. Thus the train data is subset using that field. The purpose is to match the dataset contents such that the prediction can be meaningful. We don't believe the index of the observation (X) and timestamps impact the classification. Those variables are removed as well. The side benefit is that the size of the dataset is reduced so that computation can be effective. After those preprocessing steps, there are 55 columns left.

The so-processed data2 is split into 70% and 30% portions for training and testing based on the random sampling of the classes

```
set.seed(1234)
library(lattice); library(ggplot2); library(caret)
inTrain = createDataPartition(data2$classe, p = 0.7)[[1]]
training = data2[ inTrain,]; testing = data2[-inTrain,]
```

We then use various machine learning algorithms taught in this class as well as some popular algorithms not taught in class and calculate their in-sample accuracy and out-of-sample accuracy with the following

```
reportAccuracy <- function(fit, data) {
  pred = predict(fit, data)
  sum(pred == data$classe)/nrow(data)
}
```

To report the in-sample accuracy, pass the model and training dataset; to report the out-of-sample accuracy, pass the model and the testing dataset. The code to train the models will be shown at the end of this report. Here we summarize the results in the following table:

Algorithms	In-sample accuracy	Out-of-sample accuracy
rpart	0.5733294	0.5655041
lda	0.7499443	0.7516918
nnet	0.6979112	0.6926948
knn	0.9865458	0.9652959
svm	0.9556233	0.9517612

The random forest and naive Bayes failed during the model updates for unknown reasons. The gbm runs very slow on a low end PC due to limited memory. The reason we didn't reduce the dimensionality of the data or compress the data using e.g. principal component analysis (PCA) is that other algorithms such as k-nearest neighbor (kNN) and support vector machine (SVM) give excellent results.

The algorithms taught in class such as rpart or linear discrimant analysis (LDA) didn't give good results. We thus tried neural network, kNN and SVM. The latter two give excellent results. We didn't attempt any regularization techniques, as the amount of data is pretty high and overfitting doesn't seem to be an issue.

In general, the in-sample accuracy is slightly higher than out-of-sample accuracy, as expected. For this project, we finally choose kNN or SVM algorithms as they both have the out-of-sample accuracy >95%. We didn't further combine the models as we are satisfied with the accuracy.

To predict the missing classes in the testpa2 dataset, simply do

```
predict(fitknn, testpa2)
```

We used the prediction results to complete the second half of the project and achieved 95% accuracy there.

In short, we have preprocessed the dataset to remove NA columns, tried various machine learning algorithms, and selected the kNN and SVM to predict how well weight lifting exercises are performed. The in-sample and out-of-sample accuracies for the chosen algorithms are >95% and on the second part of the project, we achieved 95% accuracy as well.

```
library(rpart)
fitrpart = train(classe ~ ., data=training, method="rpart")

library(MASS)
fitlda = train(classe ~ ., data=training, method="lda")

library(nnet)
fitnn = train(classe ~ ., data=training, method="nnet",
            preProcess = "range", tuneLength=2, trace=FALSE, maxit=100)

fitknn = train(classe ~ .,
            data=training, method="knn", preProcess = c("center", "scale"),
            tuneLength = 10, trControl = trainControl(method = "cv"))

library(e1071)
fitsvm = svm(classe ~ ., data=training)
```