Mach.Learn.

FT

Wednesday, August 13, 2014

Machine learning

Executive summary: According to the requirements, a machine learning process is built in order to predict a target label (*class*) on a testing data set. After some considerations and insights gained from exploring the data sets and its characteristics (big dimensionality with >10000 rows), a support vector machine model is attempted and built, best suited for train speed, accuracy and agility in testing. The result are an in-sample accuracy of 96% and a K-Fold CV validated out-of-sample accuracy of 95%. *Due to the otherwise long time in regenerating the document and for security, only relevant snippets of codes are included, and boilerplate code (i.e. loading files) is not present*

Step 1 - Basic exploratory data analysis

Due to the high dimensionality of the training data a quick summarization is needed to see if a basic pruning of the features space is needed and how the features are indipendently distributed. Using a basic but useful "summary(training)" in R shows that are many features with almost only missing data; due to majority of missing data in such variables (>95% NAs), a meaningless interpolation attempt in not considered; instead they are candidate for total drop from the training data.

Step 2 - Preprocessing

First the columns with >95% values set to NA are dropped from the data using a small loop building a logical vector of the columns to include (*goodcols*):

```
ncols <- dim(train)[2]
goodcols <- logical(ncols)
for(ci in 1:ncols) {
        l <- length(train[, ci])
        nas <- sum(is.na(train[, ci]))
        goodcols[ci] <- nas/l < 0.95
}
```

Then looking at the remaining features in the data, the ones with too much specificity or unrelated to prediction (ie. the user name, row numbers...) are dropped from the data set with manual intevention:

```
train <- subset(train,
              select = -c(X, user_name, raw_timestamp_part_1,
                          raw_timestamp_part_2, cvtd_timestamp, new_window, num_window))
```

The post processed train data contains now 53 predictors (in comparison of the 160 variables of the original data set). Further feature space reduction is not attempted, considering that such number of predictors is a number that a proper SVM training algorithm can handle.

Step 3 - Model training and out of sample error estimation

First a basic decision tree was fitted on the tidied training data (not included in this report for coinciseness) for model exploration, but it was leading to poor performance and accuracy; for speed and accuracy then a SVM (support vector machine model) was chosen and trained on all the variables of the post processed data set, using the routines in the e1071 package. To estimate the out of sample accuracy, K-fold cross validation on K=10 splits was used, then the accuracies of the ten models were averaged.

```
pagree <- fuction(f,s) { mean(f == s) }
split <- rep(1:10, length.out = nrow(train))
split <- sample(split, replace=F)
accuracies <- numeric(10)
for(si in 1:10) {
        model <- svm(classe ~ ., data=train[(si != split), ])
        pred <- predict(model, train[si == split, ])
        accuracies[si] <- pagree(pred, train[si == split, ]$classe)
}
mean(accuracies)
```

```
[1] 0.9523492
```

The overall model is then expected to predict with an accuracy of 95%.

Conclusion

The final model was then simply trained with all the training data, producing an in-sample accuracy of 96%; it was then used to predict the submission based homework, with 100% accuracy, thus confirming the good estimation of out-of-sample error given by K-fold CV.