Machine Learning Project

David H. Millis

Friday, July 25, 2014

Objective

The objective of this project was to determine whether the manner in which subjects performed an exercise could be predicted from activity data collected during exercise. We were provided with a training set of 19622 labeled instances and a test set of 20 instance. Our objective was to predict the correct labels for the 20 test instance.

Methodology

Several machine learning algorithms were tested to determine which model to use for this project. To evaluate multiple models, we separated the training set into training and test cases. We evaluated the accuracy of each of the models on the task of assigning labels to labeled test cases. We selected the model with the highest accuracy for labeling of the 20 unlabeled test cases provided.

Data Preparation

The raw data was read from the data files provided:

```
trn_dat_raw <- read.csv("./pml-training.csv", stringsAsFactors=FALSE)
tst_dat_raw <- read.csv("./pml-testing.csv", stringsAsFactors=FALSE)
```

The original data consists of 160 columns. This includes several columns with character data, two columns with timestamp data, many columns with predominantly "NA" entries, and many columns with predominantly blank entries. We began by creating a list of columns to be removed from consideration. We started this list with the first column, which is a row identifier, and the two timestamp columns:

```
dropcols <- c(1, 3, 4)
```

We then added to the list all columns consisting of character data. This also eliminated columns where most of the entries consist of a blank string (""). We preserved the instance labels in the "classe" column. This is the last column in the data set.

```
for (i in 1:colcount-1)
{
  if (is.character(trn_dat_raw[, i])) dropcols <- c(dropcols, i)
}
```

We added to the list all columns where more than 50% of the entries are NA:

```
drop_trigger <- round(0.5 * trn_rowcount)

for (i in 1:colcount-1)
{
  na_list <- which(is.na(trn_dat_raw[, i]))
  if (length(na_list) > drop_trigger) dropcols <- c(dropcols, i)
}
```

Finally, we removed the columns in the drop list from both the training data and testing data:

```
trn_dat_clean <- trn_dat_raw[, -dropcols]
tst_dat_clean <- tst_dat_raw[, -dropcols]
```

The data cleaning process left us with 53 features to be considered for further analysis.

Creation of Experimental Training Set and Test Set

We divided the training set into an experimental training set consisting of 60% of the training data and an experimental test set consisting of 40% of the training data, so that we could estimate the out of sample error for different models

```
library(caret)
inTrain <- createDataPartition(y=trn_dat_clean$classe, p=0.60, list=FALSE)

trn_dat_exp <- trn_dat_clean[inTrain, ]
tst_dat_exp <- trn_dat_clean[-inTrain, ]
```

Feature Reduction

Prior to model construction, we felt it would be helpful to reduce the feature set further to identify a set of features that would account for the majority of the variability in the data. We used principle components analysis (PCA) to accomplish this. First, we created unlabeled versions of both the experimental training and testing data:

```
trn_dat_unlabeled <- trn_dat_exp[, -54]
tst_dat_unlabeled <- tst_dat_exp[, -54]
```

We then use the preProcess method in caret to create a preprocessing object that would apply PCA to identify principle components accounting for 90% of the variability in the experimental training data. PCA identified 19 principle components. We applied the same preprocessing step to the experimental testing data.

```
ppr_obj <- preProcess(trn_dat_unlabeled, method="pca", thresh=.90)
trn_dat_pca <- predict(ppr_obj, trn_dat_unlabeled)
tst_dat_pca <- predict(ppr_obj, tst_dat_unlabeled)
```

Model Selection

Prior to model creation, we added the labels back to the experimental training and test data:

```
trn_dat_pca_labeled <- cbind.data.frame(trn_dat_pca, "classe" = trn_labels_exp)
tst_dat_pca_labeled <- cbind.data.frame(tst_dat_pca, "classe" = tst_labels_exp)
```

We evaluated the following models:

[1] K-Nearest Neighbor

```
library(class)
predictions <- knn(train = trn_dat_pca,
                       test = tst_dat_pca,
                       cl = trn_labels_exp,
                       k = 21)
```

[2] Support Vector Machine, kernel function = dot product

```
library(kernlab)
modelFit <- ksvm(classe ~., data=trn_dat_pca_labeled, kernel="vanilladot")
predictions <- predict(modelFit, newdata=tst_dat_pca_labeled)
```

[3] Support Vector Machine, kernel function = Gaussian

```
modelFit <- ksvm(classe ~., data=trn_dat_pca_labeled, kernel="rbfdot", kpar=list(sigma=0.05))
predictions <- predict(modelFit, newdata=tst_dat_pca_labeled)
```

[4] Random Forest

```
modelFit <- train(classe ~., data=trn_dat_pca_labeled, method="rf")
predictions <- predict(modelFit, newdata=tst_dat_pca_labeled)
```

Evaluation of Models

We generated a confusion matrix comparing the labels predicted for the experimental test set to the known labels. The accuracy of the four models is as follows:

k-nearest neighbor: 56%

SVM, kernel function = dot product: 56%

SVM, kernel function = Gaussian: 91%

Random forest: 97%

We selected the random forest model as the one we would use for labeling the unlabeled instances in the original data set. The random forest model the highest accuracy (97%). We estimated the out of sample error rate for this model at 3%.

Final Model Creation

To generate the best possible model for labeling the original unlabeled test data, we set up a training control in caret to perform five-fold cross validation, repeated five times:

```
ctrl <- trainControl(method = "repeatedcv",
                     number = 5, repeats = 5)
```

We repeated the data cleaning steps and PCA steps on the full original set of training data, and applied the same PCA processing to the unlabeled test data. We then generated a random forest model based on the full training data, with cross validation orchestrated by the training control:

```
modelFit <- train(classe ~., data = trn_dat_pca_labeled, method = "rf",
                  trControl = ctrl )
```

We applied the resulting model to generate a list of predicted labels for the unlabeled test data:

```
predictions <- predict(modelFit, newdata=tst_dat_pca)
```

Results

We submitted our final list of predictions to the project administrators. Our random forest model was able to apply correct labels to 100% of the 20 cases in the test data set.

Summary

Our objective was to test whether the manner in which subjects performed an exercise could be predicted from activity data collected during exercise. We compared several machine learning algorithms: k-nearest neighbor, support vector machines using dot product and Gaussian kernel functions, and random forest. We selected the random forest algorithm for building our final model. Our model was successful in correctly labeling 100% of the test cases in an unlabeled data set. This study confirms the potential usefulness of machine learning models in the prediction of behavioral characteristics from physiological data.