Practical Machine Learning Course Project

D. Rice

Thursday, August 21, 2014

Synopsis

Human activity recognition (HAR) researchers studied whether it was possible to evaluate “how well” an activity was being performed. Specifically, they evaluated whether a weight-lifting exercise could be evaluated for correct performance, by using motion measurement instruments attached to test subjects. Data was collected from the instruments, along with an instructor’s “outcome evaluation”: whether the exercise was performed correctly (outcome “A”) or one of four incorrect performance categories (outcomes “B” through “E”). The collected data was used to derive a prediction algorithm using machine learning techniques. The final algorithm from this process had 98.2% accuracy. This study is described at http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). (The full citation for the research paper is at the end of this report.)

The objective of the present analysis is to use a subset of the experimental data and apply machine learning techniques from this class. The data can be downloaded from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv), and https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The techniques applied here will not match the sophistication of the above-mentioned study, and as a result yield a lower level of accuracy.

Methods

The steps in this analysis are:

```
1. Obtain and load data
2. Examine and clean data
3. Separate training data into training and cross-validation subsets
4. Apply model to training subset
5. Evaluate model predictions on training and cross-validation subsets
6. Use model to predict for test set
```

Obtain and load data

```
## R program for Data Sciences/Practical Machine Learning Class Project

## load required packages (assumes they have been installed)
require(caret)
```

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

```
require(rpart)
```

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

```
## set seed for random partitioning
set.seed(951)

## load training data (assume in working directory)
pml.training <- read.csv("pml-training.csv", stringsAsFactors=FALSE)

## load test data (assume in working directory)
pml.test <- read.csv("pml-testing.csv", stringsAsFactors=FALSE)
```

Examine and clean data

The training dataset includes 160 data columns for 19,622 observations. 67 of these columns contained a significant number of NAs. Another 33 columns contained a significant number of blanks (no data) or invalid data (divide by zero). Although there may be some small predictive value in these columns, for purpose of this analysis they were removed.

In addition, text columns were converted to factors. This included the subject's name and the A-E outcome ("classe"). A converted-to-text time variable was eliminated, and the numeric time columns were normalized.

Finally, the imported row number was deleted. (This is unfortunate, since it was a 100% accurate predictor of the outcome, since the observations were listed in outcome order; all of the "A"s, then all of the "B"s, etc.)

The same conversions were applied to the test dataset.

```
## identify columns in training set without any NAs
keep <- colSums(is.na(pml.training))==0

## remove NA columns
train <- pml.training[,keep]

## identify columns with mostly blank cells
## (there are about 400 rows with data in otherwise blank columns)
keep <- colSums(train=="")<19000

## remove mostly blank columns
train <- train[,keep]

## convert selected text columns to factors
train$user_name <- as.factor(train$user_name)
train$new_window <- as.factor(train$new_window)
train$classe <- as.factor(train$classe)

## remove text version of time to numeric, scale timestamp_part_1
train$cvtd_timestamp <- NULL
z <- min(train$raw_timestamp_part_1)
train$raw_timestamp_part_1 <- train$raw_timestamp_part_1 - z

## remove row number (perfect predictor, since the outcomes are ordered!)
train <- train[,-1]

## reduce test set to same columns as training set
test <- pml.test[,names(train)[1:ncol(train)-1]]  ## no classe in test set

## make variable changes to test to match train
test$user_name <- as.factor(test$user_name)
test$new_window <- as.factor(test$new_window)
test$raw_timestamp_part_1 <- test$raw_timestamp_part_1 - z
```

Separate training data into training and cross-validation subsets

“train1” is an 80% training subset of the original training set; “train2” is the remaining 20%, for cross-validation.

```
## split original training set into training (80%) and cross-valiadation sets (20%)
inTrain <- createDataPartition(y=train$classe,p=.8,list=FALSE)
train1 <- train[inTrain,]
train2 <- train[-inTrain,]
```

The remaining variables are examined for zero or near-zero variance. It is not significant; only one variable has near-zero variance.

```
## do analysis of variable variance
nsv <- nearZeroVar(train1, saveMetrics=TRUE)

## show variables with zero or near-zero variance
row.names(nsv)[nsv$zerovar==TRUE]
```

```
## character(0)
```

```
row.names(nsv)[nsv$nzv==TRUE]
```

```
## [1] "new_window"
```

Now the remaining variables are examined for high correlation (greater than 0.9). These 7 variables could be removed from the predictor set. However, the algorithm used to build the model will deal with these variables appropriately.

```
## show variables with high correlation
trainCorr <- cor(train1[,6:57])  ## numeric variables only
highCorr <- findCorrelation(trainCorr, 0.9) + 5
names(train1[highCorr])
```

```
## [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"
## [4] "accel_belt_x"     "gyros_arm_y"      "gyros_forearm_z"
## [7] "gyros_dumbbell_x"
```

Apply model to training subset

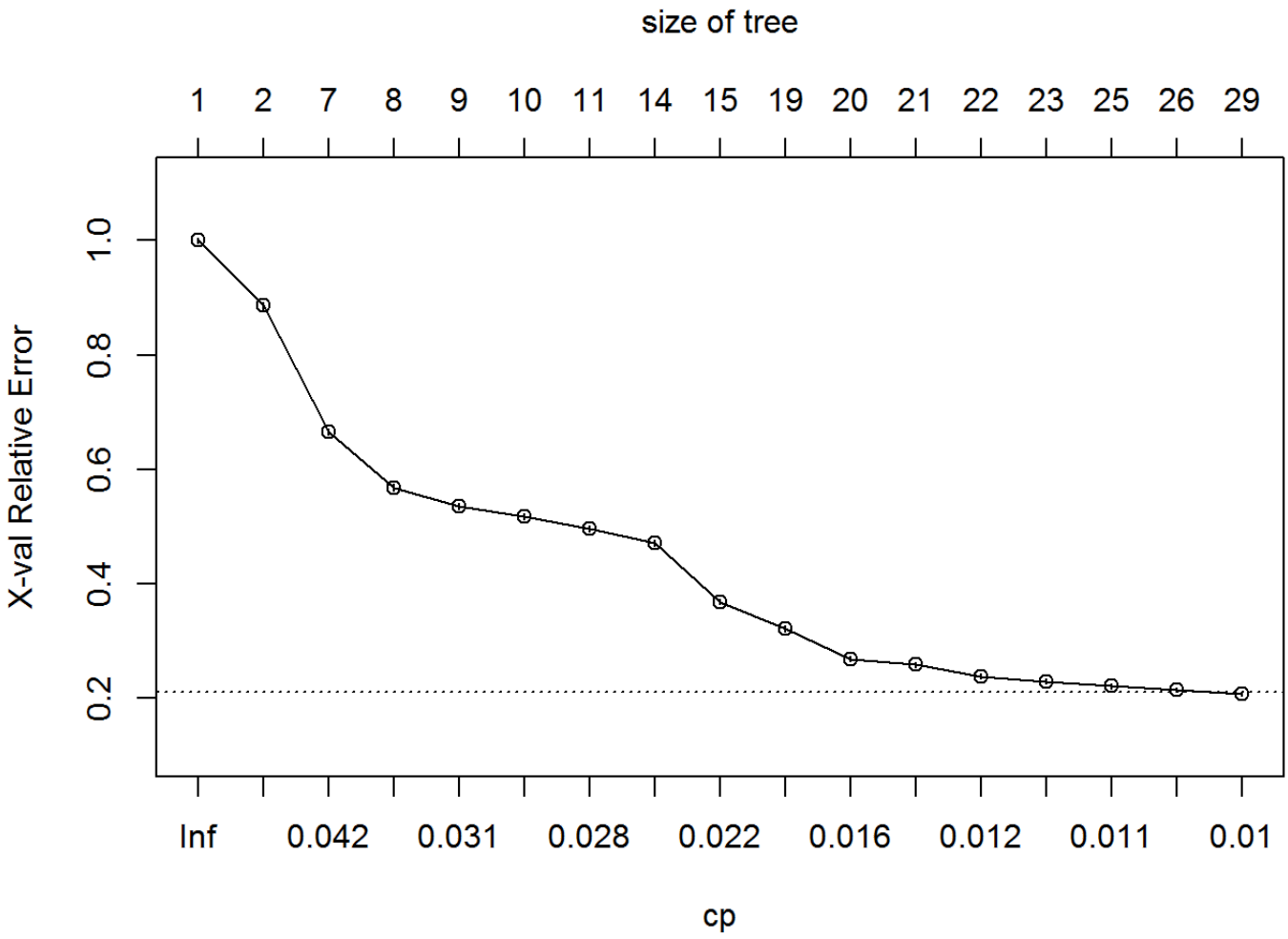
The “Rpart” (recursive partitioning) algorithm was selected. (See citation at the end for reference material.) This algorithm splits the observations on the basis of variable values, creating a tree structure. This tree is the basis of a rotation matrix for converting observation values into predictions.

A graph of the model’s convergence across tree building levels is satisfactory.

```
## remove no-longer-needed variables
rm(pml.training, pml.test, keep, z, inTrain, nsv, trainCorr, highCorr)

## fit a model to the training subset
model <- rpart(train1$classe ~ ., data=train1[,-(ncol(train1)-1)], method="class")

## display model results
plotcp(model)
```



Evaluate model predictions on training and cross-validation subsets

The derived prediction model is now applied to the training, cross-validation and test sets. These predictions are in the form of a 5-column matrix, with each column representing the estimated probability of the related A-E outcome. A function is defined to convert the matrix into an outcome prediction, based on the column with the highest probability.

```
## predict using model for training, cross-validation and test sets
p1 <- as.data.frame(predict(model,train1[,1:ncol(train1)-1]))
p2 <- as.data.frame(predict(model,train2[,1:ncol(train2)-1]))
ptest <- as.data.frame(predict(model,test[,1:ncol(test)]))

## function to convert prediction matrix to prediction value (using max)
createClasse <- function (pred){
  pp <- data.frame(classe=NA)
  for (i in 1:nrow(pred)) {
    pp[i,1] <- names(pred)[which(pred[i,]==max(pred[i,]))]
  }
  return(as.matrix(pp))
}
```

Now we compare the predicted outcome with the actual outcome for the training subset. The result has an accuracy of 89.0%, with “balanced accuracy” above 91% for all outcomes except “B”.

```
## compare predicted to actual values for training set
actual1_classe <- as.matrix(as.character(train1$classe))
pred1_classe<-createClasse(p1)
confusionMatrix(train1$classe,as.factor(pred1_classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4109  321    0   23   11
##          B   44 2824   63  107    0
##          C    3  316 2345   74    0
##          D    0  208  165 2067  133
##          E   33   81   15  128 2629
##
## Overall Statistics
##
##                Accuracy : 0.89
##                  95% CI : (0.885, 0.895)
##     No Information Rate : 0.267
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.861
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.981    0.753    0.906    0.862    0.948
## Specificity             0.969    0.982    0.970    0.962    0.980
## Pos Pred Value          0.920    0.930    0.856    0.803    0.911
## Neg Pred Value          0.993    0.927    0.981    0.975    0.989
## Prevalence              0.267    0.239    0.165    0.153    0.177
## Detection Rate          0.262    0.180    0.149    0.132    0.167
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.975    0.868    0.938    0.912    0.964
```

Likewise we compare the predicted outcome with the actual outcome for the cross-validation subset. We would expect the “out of sample” accuracy to be lower, and it is (slightly) lower, at 88.8%. Again, “balanced accuracy” is higher for all outcomes except “B”.

```
## compare predicted to actual values for cross-validation set
actual2_classe <- as.matrix(as.character(train2$classe))
pred2_classe<-createClasse(p2)
confusionMatrix(train2$classe,as.factor(pred2_classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1019   86    0    8    3
##          B    8  708   15   28    0
##          C    2   90  570   22    0
##          D    0   37   41  537   28
##          E   10   18    4   39  650
##
## Overall Statistics
##
##                Accuracy : 0.888
##                  95% CI : (0.878, 0.898)
##     No Information Rate : 0.265
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.859
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.981    0.754    0.905    0.847    0.954
## Specificity             0.966    0.983    0.965    0.968    0.978
## Pos Pred Value          0.913    0.933    0.833    0.835    0.902
## Neg Pred Value          0.993    0.927    0.981    0.970    0.990
## Prevalence              0.265    0.239    0.161    0.162    0.174
## Detection Rate          0.260    0.180    0.145    0.137    0.166
## Detection Prevalence    0.284    0.193    0.174    0.164    0.184
## Balanced Accuracy       0.974    0.868    0.935    0.907    0.966
```

Use model to predict for test set

Finally, we apply the model to the test set, and create the submission files for the second part of this assignment. Predictions are displayed below. There is no comparison for test outcome, until the results of the submission assignment.

```
## predict for test set
predTest_classe<-createClasse(ptest)

## display results
paste(predTest_classe)
```

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

```
## write files for prediction submission

## function to create one-charater files from test-set predictions
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE,eol="")
  }
}
## write files
pml_write_files(as.character(predTest_classe))
```

Citation for original study: 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. http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf (http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf)

Citation for RPART document: Therneau, T.; Atkinson, E.; Mayo Foundation. “An Introduction to Recursive Partitioning Using the RPART Routines” September 3, 1997. http://www.mayo.edu/hsr/techrpt/61.pdf (http://www.mayo.edu/hsr/techrpt/61.pdf)

(Total word count, excluding R code and output, is approximately 800 words.)