Qualitative Activity Recognition of Weight Lifting

Paul Ringsted, 5th February 2019 - Course 8 (Practical Machine Learning)

Synopsis

The purpose of this study was to analyze data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, in order to predict a qualitative assessment of the exercise being performed, classified 5 different ways on a scale of A-E (A being specified execution of the exercise, and B-E corresponding to common mistakes).

The dataset was first cleaned to remove columns with incomplete data. We concluded that using a Random Forest algorithm with 5-fold cross validation, trained on 75% of the provided data and tested using the remaining 25% of the training data, provided an accuracy rate of 99.51% (with a 95% confidence interval of 99.27-99.69%). The model was applied to the 20 test cases provided, which were all predicted correctly.

References

Note: R code is reflected in the Appendix. PDF was used as per prior projects instead of HTML.

More information on the original study is available from the website here:

http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Guidance on configuration for parallel processing:

https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md

Data Loading and Initial Analysis

The training and testing data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Table 1: Rowcount by Classe for Training Data

Dataset	Rowcount	Classe A	Classe B	Classe C	Classe D	Classe E
Training total rows	19622	5580	3797	3422	3216	3607
Training rows with NAs	19216	5471	3718	3352	3147	3528
Training rows with no NAs	406	109	79	70	69	79

Only 406 rows have values for all columns, these correspond to records where the 'new_window' flag equals to 'yes'. The training data provided has a time window structure to it based on participant and exercise. For the purposes of this analysis we removed all columns which have NA values (which will exclude some data on the very small subset of complete rows, but this data is not useful if unavailable for all observations). We also removed the first 7 columns, which provide test case and time window information, that should not be used as predictors.

The resulting dataset we will use to build the prediction model consists of 19622 observations of the 'classe' outcome, on 52 predictors.

Model Training

As a starting point given the large number of predictors and need for high level of accuracy, we will execute a Random Forest model using 5-fold cross-validation as a baseline for the prediction model. To help validate the accuracy prior to applying it to the provided "testing" data (20 cases), we will split the data into 75% observations for "training" and 25% for "validation". This model was implemented using parallel processing.

Table 2: Breakdown of Training and Validation Datasets

Dataset	Rowcount	Classe A	Classe B	Classe C	Classe D	Classe E
Training data - training subset	14718	4185	2848	2567	2412	2706
Training data - validation subset	4904	1395	949	855	804	901

The results of the model are as follows, showing an average accuracy over 5 folds of 99.13%

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.69%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4176    5    2    0    2 0.002150538
## B   20 2822    6    0    0 0.009129213
## C    0    9 2548   10    0 0.007401636
## D    0    3   29 2377    3 0.014510779
## E    0    1    5    6 2694 0.004434590
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.4  0.2  0.0  0.0  0.0
##          B  0.0 19.1  0.1  0.0  0.0
##          C  0.0  0.1 17.3  0.2  0.0
##          D  0.0  0.0  0.1 16.1  0.1
##          E  0.0  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9913
```

```
##    Accuracy     Kappa Resample
## 1 0.9915024 0.9892489    Fold1
## 2 0.9901461 0.9875356    Fold3
## 3 0.9925272 0.9905464    Fold2
## 4 0.9942255 0.9926957    Fold5
## 5 0.9881154 0.9849662    Fold4
```

Model Validation

Next, we used the model to predict the classe for the validation population we set aside, and reviewed the resulting confusion matrix. **This confirmed an acceptable accuracy rate of 99.51% (with a 95% confidence interval of 99.27-99.69%).**

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    3  944    2    0    0
##          C    0    3  845    7    0
##          D    1    0    4  797    2
##          E    0    0    1    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9958   0.9918   0.9913   0.9978
## Specificity            0.9997   0.9987   0.9975   0.9983   0.9998
## Pos Pred Value         0.9993   0.9947   0.9883   0.9913   0.9989
## Neg Pred Value         0.9989   0.9990   0.9983   0.9983   0.9995
## Prevalence             0.2851   0.1933   0.1737   0.1639   0.1839
## Detection Rate         0.2843   0.1925   0.1723   0.1625   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9984   0.9973   0.9947   0.9948   0.9988
```

Model Prediction of Testing Data

Finally, we can now predict the classe category A-E for the 20 "testing" observations provided, with the following results:

Table 3: Results of Model Prediction on Testing Data

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B

Appendix - Model Alternative with PCA

Given the large number of predictors, we also tried to apply PCA pre-processing during the original model fitting, to see if that yields any improvement. Dimensions of the PCA matrix:

```
## [1] 14718    13
```

The results of the model are as follows. This showed a significant decrease in accuracy to 95% when using pre-processing, so **PCA was not used in the final model.**

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.81%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4098   21   41   18    7  0.02078853
## B   76 2699   54   14    5  0.05231742
## C   30   39 2460   29    9  0.04168290
## D   18   13  112 2264    5  0.06135987
## E    7   20   20   23 2636  0.02586844
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 27.6  0.6  0.3  0.1  0.0
##          B  0.2 18.1  0.3  0.1  0.2
##          C  0.3  0.5 16.5  0.9  0.2
##          D  0.3  0.1  0.3 15.2  0.2
##          E  0.1  0.1  0.1  0.1 17.8
##
##  Accuracy (average) : 0.9526
```

```
##    Accuracy     Kappa Resample
## 1 0.9470109 0.9329864    Fold1
## 2 0.9514431 0.9385655    Fold2
## 3 0.9537887 0.9415587    Fold5
## 4 0.9581917 0.9470804    Fold4
## 5 0.9527853 0.9402635    Fold3
```

Code Appendix - R Code

```r
library(knitr)
opts_chunk$set(fig.width=6, fig.height=3.5, fig.pos = "H", echo=FALSE, eval=TRUE)
#-------------------------------------------------------------------------------------
library(caret)
library(parallel)
library(doParallel)
library(kableExtra)
library(data.table)
set.seed(23675)         # Set seed to make results reproducible

#-------------------------------------------------------------------------------------
# Load training data from working directory and perform initial review
data<-read.csv('pml-training.csv',na.strings=c("","NA"))
test<-read.csv('pml-testing.csv',na.strings=c("","NA"))

# Split the data between data without NAs and with NAs
data_nona<-data[rowSums(is.na(data)) == 0,]
data_na<-data[rowSums(is.na(data)) > 0,]

# Only keep columns for which the non-NA count equals to the rowcount (no NA data)
# Strip first 7 columns which have information on the test case
data_subcols<-data[,colSums(!is.na(data))==nrow(data)]
data_subcols<-data_subcols[,-(1:7)]

# Generate basic statistics table
stats1<-transpose(as.data.frame(aggregate(data$classe,by=list(data$classe),FUN=length)$x))
stats1<-cbind(dataset="Training total rows",tot=nrow(data),stats1)
stats2<-transpose(as.data.frame(aggregate(data_na$classe,
                                          by=list(data_na$classe),FUN=length)$x))
stats2<-cbind(dataset="Training rows with NAs",tot=nrow(data_na),stats2)
stats3<-transpose(as.data.frame(aggregate(data_nona$classe,
                                          by=list(data_nona$classe),FUN=length)$x))
stats3<-cbind(dataset="Training rows with no NAs",tot=nrow(data_nona),stats3)

statstab <- rbind(stats1,stats2,stats3)
statstab %>% kable(
                col.names=c("Dataset","Rowcount","Classe A","Classe B","Classe C",
                "Classe D","Classe E"),
                booktabs=T,align=rep("r",6),digits=c(0,0,0,0,0,0),
                caption="Rowcount by Classe for Training Data") %>%
                kable_styling(latex_options = "hold_position")

#-------------------------------------------------------------------------------------
# Split 'training' dataset 75-25%
intrain<-createDataPartition(data_subcols$classe,p=0.75)[[1]]
data_train<-data_subcols[intrain,]
data_valid<-data_subcols[-intrain,]

# Build a basic stats table to check the partitions
stats4<-transpose(as.data.frame(aggregate(data_train$classe,
                                          by=list(data_train$classe),FUN=length)$x))
```

```
stats4<-cbind(dataset="Training data - training subset",tot=nrow(data_train),stats4)
stats5<-transpose(as.data.frame(aggregate(data_valid$classe,
                                          by=list(data_valid$classe),FUN=length)$x))
stats5<-cbind(dataset="Training data - validation subset",tot=nrow(data_valid),stats5)

statstab2 <- rbind(stats4,stats5)
statstab2 %>% kable(
                col.names=c("Dataset","Rowcount","Classe A","Classe B","Classe C",
                "Classe D","Classe E"),
                booktabs=T,align=rep("r",6),digits=c(0,0,0,0,0,0),
                caption="Breakdown of Training and Validation Datasets") %>%
                kable_styling(latex_options = "hold_position")

#-------------------------------------------------------------------------------------
# Set up training control structure & initiate clustering
fitControl<-trainControl(method="cv",number=5,allowParallel=TRUE)
cluster<-makeCluster(detectCores()-1)
registerDoParallel(cluster)

# Take a walk through the random forest
fit<-train(classe~.,method="rf",data=data_train,trControl=fitControl)

# Turn off clustering, return to single-threaded
stopCluster(cluster)
registerDoSEQ()

#-------------------------------------------------------------------------------------
# Display results from the fit and confusion matrix based on folds
fit$finalModel
confusionMatrix.train(fit)
fit$resample

#-------------------------------------------------------------------------------------
# Run prediction against validation dataset and display confusion matrix
pred<-predict(fit,data_valid)
confusionMatrix(data_valid$classe,pred)

#-------------------------------------------------------------------------------------
# Run prediction against test data and display the results
predtest<-predict(fit,test)
pred_df<-transpose(as.data.frame(predtest))
colnames(pred_df)<- c(1:20)
pred_df %>% kable(booktabs=T,
                  caption="Results of Model Prediction on Testing Data") %>%
                kable_styling(latex_options = "hold_position")

#-------------------------------------------------------------------------------------
# Pre-process with PCA
pca<-preProcess(data_train,method="pca",thresh=0.8)
data_pca<-predict(pca,data_train)
dim(data_pca)

#Turn on clustering
```

```r
cluster<-makeCluster(detectCores()-1)
registerDoParallel(cluster)

# Take a walk through the random forest
fitpca<-train(classe~.,method="rf",data=data_pca,trControl=fitControl)

# Turn off clustering, return to single-threaded
stopCluster(cluster)
registerDoSEQ()

#---------------------------------------------------------------------------------
#Display results from the PCA model fit
fitpca$finalModel
confusionMatrix.train(fitpca)
fitpca$resample
```