Predicting Quality of Movement: Unilateral Dumbbell Biceps Curl

9/08/2020

Executive summary

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this report, I quantify the manner in which people did an exercise using data from accelerometers on the belt, forearm, arm, and dumbell of six participants (see the appendix for more information).

I built an majority vote multi-class classifier based on three different supervised learning models: random forest, generalized boosted model, and XGBoost:

$$mvm\,(X) = mode(rfm(X), gbm(X), xgb(X))$$

I split the pml-training.csv data into 50% training and 50% validation. I used 5-fold cross validation to build the random forest model and the generalized boosted model by training on the training data. I used the 50% validation data for estimating the out-of-sample error of the three models.

Table 1: Accuracy and 95% confidence intervals for the on the validation data

model	model_ID	accuracy	accuracy_conf_interval
random forest	rfm	0.9990	[0.9981, 0.9995]
generalized boosted	gbm	0.9958	[0.9943, 0.997]
XGBoost	xgb	0.9993	[0.9985, 0.9997]
majority vote	mvm	0.9992	[0.9984, 0.9996]

The XGBoost model has the best accuracy on validation data. However, the combination of the three models is expected to have a lower error rate (Gareth James, Majority Vote Classifiers: Theory and Applications, PhD thesis). For this reason, I used the majority vote model for predictions on the test data.

Training and validation data

Import and clean the training and validation data.

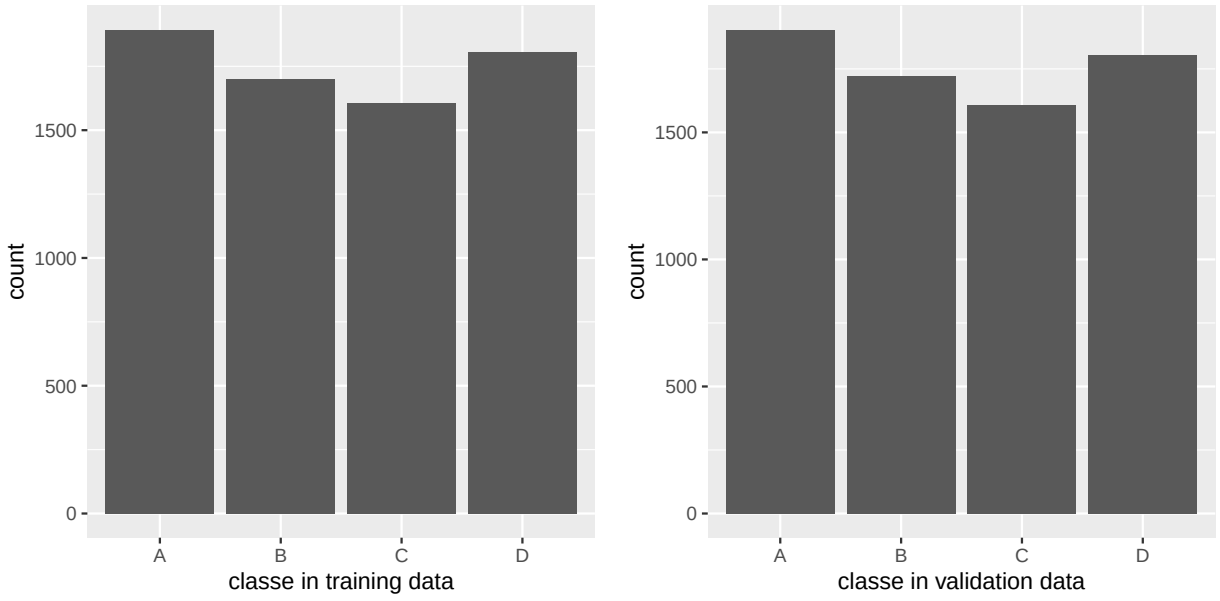
```
set.seed(123)
pmlTrain <- read.csv('./pml-training.csv',strip.white=TRUE)

# replace spaces and #DIV/0! with NA, and remove the column X
pmlTrain[pmlTrain==""]<-NA
pmlTrain <- data.frame(lapply(pmlTrain, function(x) {gsub("#DIV/0!", NA, x)}))
pmlTrain$X <- NULL
# remove columns with many NA fields and only keep complete observations
pmlTrain <- pmlTrain[, which(colMeans(!is.na(pmlTrain)) > 0.97)]
pmlTrain <- pmlTrain[complete.cases(pmlTrain),]
# transform the time to a numeric value
pmlTrain[c("cvtd_timestamp")] <- lapply(pmlTrain[c("cvtd_timestamp")], function(x) as.numeric(as.POSIXc
```

```
# write and import to automatically assign better data types
write.csv(pmlTrain,"./train_temp.csv",row.names = FALSE)
pmlTrain <- read.csv("./train_temp.csv",strip.white=TRUE)

# one hot encoding
user_name_ohe <- model.matrix(~user_name-1,pmlTrain)
new_window_ohe <- model.matrix(~new_window-1,pmlTrain)
m_ohe_1 <- data.frame(data.matrix(cbind(user_name_ohe,new_window_ohe)))
pmlTrain <- pmlTrain[,-which(names(pmlTrain) %in% c("user_name","new_window"))]
pmlTrain <- cbind(pmlTrain,m_ohe_1)

# convert the labels to numeric for XGBoost
classes <- pmlTrain$classe
pmlTrain$classe <- as.integer(pmlTrain$classe) - 1

# split the data into training and validation
inTrain  <- createDataPartition(pmlTrain$classe, p=0.5, list=FALSE)
pmlValidate  <- pmlTrain[-inTrain, ]
pmlTrain <- pmlTrain[inTrain, ]
```

The training and validation datasets have a similar number of observations, the same number of columns, and a similar number of each classe feature.

Table 2: training and validation data

data	num_rows	num_columns
training	9812	65
validation	9810	65



Random Forest

The code below is used for training the random forest model and printing the confusion matrix with relevant error metrics on the validation data. I used 5-fold cross validation because there are a large number of

observations and doing LOOCV would be very computationally expensive. Cross validation is useful to identify problems like overfitting or selection bias. The confusion matrix is built on validation data in order to get a good estimate of the out-of-sample error.

```
set.seed(123)

#The model requires the classe feature to be in the form of a factor.
pmlTrain_rf1 <- pmlTrain
pmlTrain_rf1$classe <- as.factor(pmlTrain_rf1$classe)
pmlValidate_rf1 <- pmlValidate
pmlValidate_rf1$classe <- as.factor(pmlValidate_rf1$classe)

rf1 <- train(classe~., method="rf",data=pmlTrain_rf1, trControl=trainControl(method="repeatedcv", numbe

rf1_pred <- predict(rf1, newdata=pmlValidate_rf1)
rf1_cm <- confusionMatrix(rf1_pred, pmlValidate_rf1$classe)
rf1_cm
```

Generalized Boosted Model

The explanations for the choice of 5-fold cross validation and the confusion matrix are the same as the one explained in the section for Random Forest

```
set.seed(123)

#The model requires the classe feature to be in the form of a factor.
pmlTrain_gbm1 <- pmlTrain
pmlTrain_gbm1$classe <- as.factor(pmlTrain_gbm1$classe)
pmlValidate_gbm1 <- pmlValidate
pmlValidate_gbm1$classe <- as.factor(pmlValidate_gbm1$classe)

gbm1 <- train(classe~., method="gbm",data=pmlTrain_gbm1,verbose=FALSE,trControl=trainControl(method="re

gbm1_pred <- predict(gbm1, newdata=pmlValidate_gbm1)
gbm1_cm <- confusionMatrix(gbm1_pred, pmlValidate_gbm1$classe)
gbm1_cm
```

XGBoost

The code for training the XGBoost model is shown below. I did not use 5-fold cross validation (like for the other 2 models) due to time limitations; there is no direct way of getting a model using k-fold cross validation in R. I trained the model using the training set and then estimate the out-of-sample error on the validation data. XGBoost outputs a probability for each classe, and I chose the classe with the greatest probability as the single prediction of the XGBoost model.

```
# process the data using xgb.DMatrix for training and validation
xgb1_label_train <- pmlTrain$classe
xgbtrain <- as.matrix(pmlTrain[,-which(names(pmlTrain) %in% c("classe"))])
xgb1_train <- xgb.DMatrix(data=xgbtrain,label=xgb1_label_train)
xgb1_label_validate <- pmlValidate$classe
xgbvalidate <- as.matrix(pmlValidate[,-which(names(pmlValidate) %in% c("classe"))])
xgb1_validate <- xgb.DMatrix(data=xgbvalidate,label=xgb1_label_validate)

# specify the parameters and train the model
num_classes <- length(unique(xgb1_label_train))
xgb1_params <- list(
```

```
  booster="gbtree", # default is gbtree: tree based models
  objective="multi:softprob", # multi-class classification : output the probabilities
  #objective = "multi:softmax" # multi-class classification : output the class with max probability
  eval_metric="mlogloss", #multi-class logloss (aka logistic loss or cross-entropy loss, negative log-l
  eta=0.3, # control overfitting
  max_depth=5, # depth of trees
  num_class=num_classes
)
xgb1 <- xgb.train(
params = xgb1_params,
data = xgb1_train,
nrounds = 100,
verbose = FALSE,
)

# predict with XGBoost
xgb1_pred <- predict(xgb1, newdata=xgb1_validate,reshape = T)
xgb1_pred <- as.data.frame(xgb1_pred)
colnames(xgb1_pred) <- levels(classes)
# Use the predicted label with the highest probability
xgb1_pred$prediction <- apply(xgb1_pred,1,function(x) colnames(xgb1_pred)[which.max(x)])
xgb1_pred$label <- levels(classes)[xgb1_label_validate+1]

xgb1_cm <- confusionMatrix(as.factor(xgb1_pred$prediction), as.factor(xgb1_pred$label))
xgb1_cm
```

Majority vote classifier

Prediction of majority vote classifier on the validation data

```
set.seed(123)

# function to get the mode (for majority vote)
Mode <- function(x) {
ux <- unique(x)
ux[which.max(tabulate(match(x, ux)))]
}

# merge all models in a single dataframe
df_pred <- data.frame(cbind(rf1_pred,gbm1_pred,xgb1_pred$prediction))
# convert to classes A, B, C, D, E
df_pred[c('rf1_pred','gbm1_pred')] <- lapply(df_pred[c('rf1_pred','gbm1_pred')], function(x) unique(cla
#create column of majority vote
df_pred$majority_vote <- apply(df_pred, 1, FUN = function(x) as.factor(Mode(c(x[1],x[2],x[3]))))
# translate the majority vote column to numeric
df_pred$majority_vote_num <- as.integer(df_pred$majority_vote) - 1

cm_majority_vote <- confusionMatrix(as.factor(df_pred$majority_vote_num), as.factor(pmlValidate$classe)
cm_majority_vote
```

Import and clean the test data using a process that yields consistent features and types of data.

```
pmlTest <- read.csv('./pml-testing.csv',strip.white=TRUE)

# replace spaces and #DIV/0! with NA, and remove the column X
```

```
pmlTest[pmlTest==""]<-NA
pmlTest <- data.frame(lapply(pmlTest, function(x) {gsub("#DIV/0!", NA, x)}))
pmlTest$X <- NULL
# remove columns with many NA fields and only keep complete observations
pmlTest <- pmlTest[, which(colMeans(!is.na(pmlTest)) > 0.97)]
pmlTest <- pmlTest[complete.cases(pmlTest),]
# transform the time to a numeric value
pmlTest[c("cvtd_timestamp")] <- lapply(pmlTest[c("cvtd_timestamp")], function(x) as.numeric(as.POSIXct(

# write and import to automatically assign better data types
write.csv(pmlTest,"./test_temp.csv",row.names = FALSE)
pmlTest <- read.csv("./test_temp.csv",strip.white=TRUE)

#one hot encoding
user_name_ohe_test <- model.matrix(~user_name-1,pmlTest)
pmlTest$new_windowno <- 1
pmlTest$new_windowyes <- 0
m_ohe_1_test <- data.frame(data.matrix(cbind(user_name_ohe_test)))
pmlTest <- pmlTest[,-which(names(pmlTest) %in% c("user_name"))]
pmlTest <- cbind(pmlTest,m_ohe_1_test)

pmlTest <- pmlTest[names(pmlTrain)[names(pmlTrain) != "classe"]]
```

The test data has 20 observations and one feature less (classe) than the training and validation datasets.

Table 3: test data

data	num_rows	num_columns
training	9812	65
validation	9810	65
test	20	64

Majority vote predictions on the test data

```
set.seed(123)
# random forest on test
rf1_pred_test <- predict(rf1, newdata=pmlTest)
# Generalized Boosted Model on test
gbm1_pred_test <- predict(gbm1, newdata=pmlTest)

# XGBoost on test
xgb1_label_test <- factor("NA")
xgbtest <- as.matrix(pmlTest)
xgb1_test <- xgb.DMatrix(data=xgbtest)
xgb1_pred_test <- predict(xgb1, newdata=xgb1_test,reshape = T)
xgb1_pred_test <- as.data.frame(xgb1_pred_test)
colnames(xgb1_pred_test) <- levels(classes)
# Use the predicted label with the highest probability
xgb1_pred_test$prediction <- apply(xgb1_pred_test,1,function(x) colnames(xgb1_pred_test)[which.max(x)])

# merge all models in a single dataframe
df_pred_test <- data.frame(cbind(rf1_pred_test,gbm1_pred_test,xgb1_pred_test$prediction))
# convert to classes A, B, C, D, E
```

```
df_pred_test[c('rf1_pred_test','gbm1_pred_test')] <- lapply(df_pred_test[c('rf1_pred_test','gbm1_pred_t
#create column of majority vote
df_pred_test$majority_vote <- apply(df_pred_test, 1, FUN = function(x) as.factor(Mode(c(x[1],x[2],x[3])
# translate the majority vote column to numeric
df_pred_test$majority_vote_num <- as.integer(df_pred_test$majority_vote) - 1
df_pred_test
df_pred_test$majority_vote
```

Appendix A: Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

The participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Appendix B: Risks and next steps

If more time is invested in this project, it would be good to tune the hyperparameters for each model, and to explore droping features based on an importance ranking.

The split into training and validation could have been done 80% training and 20% validation. However, 50% was used because of the limited availability of computing resources.

The columns in the dataset were not standardized because this is not required for the types of models used.

The softmax objective function could be used in XGBoost to directly output the class with the greatest probability.

Appendix C: References

1. http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har
2. https://web.stanford.edu/~hastie/THESES/gareth_james.pdf
3. https://en.wikipedia.org/wiki/Cross-validation_(statistics)