Untitled

zuraini othman

Tuesday, December 15, 2015

# # Machine Learning Project

Initial setup. {r loading-stuff} library(caret) library(ggplot2) set.seed(88388) # setwd("/Users/01043/Google Drive/coursera/ProjectMACHLEARNING") ## Data analysis

Read training data and plot some statistics, e.g., frequency of classes (A through E). The task is to predict activity classes from accelerator measurements of participants performing barbell exercises.

{r freqplot,cache=TRUE} trainRaw=read.table("pml-training.csv",header=TRUE,sep=",") barplot(table(trainRaw$classe),xlab="classe",ylab="counts",main="Classes") dim(trainRaw)

The training file is rather balanced (more of class A, but not very skewed) and has 19622 instances and 159 variables (the 160th variable is the output to predict, called "classe" that is a factor with 5 levels). The str command shows that many columns contain lots of NA's and that a lot of real-valued data is surrounded by quotes and was thus interpreted incorrectly as factor.

## Data preparation

We now remove all data columns which consist of more than 90% NAs. The data set has now a much lower dimension (93 columns).

{r NAs} train=trainRaw[,colSums(is.na(trainRaw))<(nrow(trainRaw)\*0.9)] dim(train)

For the remaining columns, we inspect them with str to remove those that make less sense (i.e., counters, meta-data, time-related columns).

{r remove cols} #names(train) str(train) #remove columns train=subset(train, select=-c(X,user\_name,raw\_timestamp\_part\_1,raw\_timestamp\_part\_2,cvtd\_timestamp,new\_window,num\_window, kurtosis\_yaw\_belt,skewness\_yaw\_belt,max\_yaw\_belt,min\_yaw\_belt,amplitude\_yaw\_belt,kurtosis\_yaw\_dumbbell,skewness\_yaw\_dumbbell,amplitude\_yaw\_dumbbell,amplitude\_yaw\_dumbbell,skewness\_yaw\_forearm,amplitude\_yaw\_forearm,min\_yaw\_forearm,max\_yaw\_forearm,skewness\_pitch\_forearm,skewness\_roll\_forearm,kurtosis\_yaw\_forearm,kurtosis\_picth\_forearm,kurtosis\_roll\_forearm,max\_yaw\_dumbbell,min\_yaw\_dumbbell )) dim(train) #summary(train)

We are now down to 67 columns. Inspect again and make sure that all columns are in the right format, otherwise cast them.

{r cast} train$kurtosis\_roll\_belt=as.numeric(train$kurtosis\_roll\_belt) train$kurtosis\_picth\_belt=as.numeric(train$kurtosis\_picth\_belt) train$skewness\_roll\_belt=as.numeric(train$skewness\_roll\_belt) train$skewness\_roll\_belt.1=as.numeric(train$skewness\_roll\_belt.1) train$kurtosis\_roll\_arm=as.numeric(train$kurtosis\_roll\_arm) train$kurtosis\_picth\_arm=as.numeric(train$kurtosis\_picth\_arm) train$kurtosis\_yaw\_arm=as.numeric(train$kurtosis\_yaw\_arm) train$skewness\_roll\_arm=as.numeric(train$skewness\_roll\_arm) train$skewness\_pitch\_arm=as.numeric(train$skewness\_pitch\_arm) train$skewness\_yaw\_arm=as.numeric(train$skewness\_yaw\_arm) train$kurtosis\_roll\_dumbbell=as.numeric(train$kurtosis\_roll\_dumbbell) train$kurtosis\_picth\_dumbbell=as.numeric(train$kurtosis\_picth\_dumbbell) train$skewness\_roll\_dumbbell=as.numeric(train$skewness\_roll\_dumbbell) train$skewness\_pitch\_dumbbell=as.numeric(train$skewness\_pitch\_dumbbell)

Now the train data is in the right format, and lets train models.

## Train models on selected data and evaluate on held-out set

We now split the data into parts, and for initial inspection train models only on a random sample of 1000 train and 1000 dev instances to select which method to take. We use Naive Bayes (NB) as baseline method. Since the measurements are clearly not independent, we expect it to work badly. We expect random forests to work well, and compare them to support vector machines (SVM) with a linear and polynomial kernel.

{r initial-models,cache=TRUE} inTrain <- createDataPartition(train$classe, p=0.7,list=FALSE) #same as [[1]] instead of list=FALSE length(inTrain) inTrain=sample(inTrain,size=1000,replace=FALSE) #subset for testing training <- train[inTrain,] testing <- train[-inTrain,] testing=testing[sample(nrow(testing), 1000),] #random rows # we try several models for fun model=train(classe~.,data=training,method="rf") #model=train(classe~.,data=training,method="nb") #model=train(classe~.,data=training,method="svmLinear") #model=train(classe~.,data=training,method="svmPoly") model predictions=predict(model,newdata=testing) accuracy=sum(predictions == testing$classe)/length(testing$classe) #or: confusionMatrix(predictions,testing$classe)

In fact, the worst model is naive Bayes (NB). The best model is -- as expected -- random forest. More specifically, on the 1000 data sample, the accuracies we get are:

* NB accuracy: 60%,
* RF accuracy: 90.8%
* svmLinear: 69.7%
* svmPoly: 80.1%

## Feature selection

We now want to find the top-20 features of the RF model and train on the whole data set partition.

{r inspect} varImp(model)

Lets plot some of these features to get a feeling how well they distinguish the classes.

{r qplot} qplot(magnet\_dumbbell\_z,pitch\_forearm,data=training,color=classe)

The figure above shows that, for instance, class A and D are nicely separable using this two variables.

We now use only the top-20 features to train an RF model.

inTrain <- createDataPartition(train$classe, p=0.7,list=FALSE) #same as [[1]] instead of list=FALSE  
length(inTrain)  
training <- train[inTrain,]  
testing <- train[-inTrain,]  
training=subset(training,select=c(roll\_belt,pitch\_forearm,magnet\_dumbbell\_z,magnet\_dumbbell\_y,yaw\_belt,pitch\_belt,roll\_forearm,magnet\_dumbbell\_x,roll\_dumbbell,magnet\_belt\_z,accel\_forearm\_x,accel\_dumbbell\_y,gyros\_dumbbell\_y,roll\_arm,magnet\_belt\_x,accel\_belt\_z,magnet\_belt\_y,accel\_dumbbell\_z,magnet\_forearm\_x ,magnet\_forearm\_x,total\_accel\_dumbbell,classe))  
testing=subset(testing,select=c(roll\_belt,pitch\_forearm,magnet\_dumbbell\_z,magnet\_dumbbell\_y,yaw\_belt,pitch\_belt,roll\_forearm,magnet\_dumbbell\_x,roll\_dumbbell,magnet\_belt\_z,accel\_forearm\_x,accel\_dumbbell\_y,gyros\_dumbbell\_y,roll\_arm,magnet\_belt\_x,accel\_belt\_z,magnet\_belt\_y,accel\_dumbbell\_z,magnet\_forearm\_x ,magnet\_forearm\_x,total\_accel\_dumbbell,classe))  
dim(training)

We get the same accuracy (88.9%) but this is faster to train (takes 2/3 of the time). We will use this set of features with RF for our final model.

So far we have trained models on only a subset of the data. To get an estimate of the out-of-sample error we train a random forest model on the whole training data using cross-validation.

fitControl <- trainControl(## 10-fold CV  
 method = "repeatedcv",  
 number = 10,  
 repeats = 10)  
model=train(classe~.,data=training,method="rf",trControl=fitControl)

This takes quite some time to finish, nevertheless, from the cross validation accuracy (98.9%) we can get an estimated of the out-of-sample performance.

## Final model

Now we train the final model using the seleted features on the entire training data set and evaluate it on the test set of 20 instances.

model=train(classe~.,data=training,method="rf")  
predictions=predict(model,newdata=testing)  
accuracy=sum(predictions == testing$classe)/length(testing$classe) #or:  
accuracy  
confusionMatrix(predictions,testing$classe)

We reach an accuracy of 99.81% on the held-out set. More specifically, the sensitivity and specificity per class is, respectively: A 0.9988/0.9993, B 0.9939/0.9992, C 0.9981/0.9992, D 1.0/1.0, E 1.0/1.0.

Now we run it on the real test data and write the predictions to the files. Submitting the test run on the website shows that we actually got an accuracy of 100% (with just 20 features). Thus, our estimate carried nicely over. Yeah!

testRaw=read.table("data/pml-testing.csv",header=TRUE,sep=",")   
testing=subset(testRaw,select=c(roll\_belt,pitch\_forearm,magnet\_dumbbell\_z,magnet\_dumbbell\_y,yaw\_belt,pitch\_belt,roll\_forearm,magnet\_dumbbell\_x,roll\_dumbbell,magnet\_belt\_z,accel\_forearm\_x,accel\_dumbbell\_y,gyros\_dumbbell\_y,roll\_arm,magnet\_belt\_x,accel\_belt\_z,magnet\_belt\_y,accel\_dumbbell\_z,magnet\_forearm\_x ,magnet\_forearm\_x,total\_accel\_dumbbell))  
dim(testing)  
predictionsTest=predict(model,newdata=testing)  
length(predictionsTest)  
  
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)  
 }  
}  
pml\_write\_files(predictionsTest)

### References

[1] <http://groupware.les.inf.puc-rio.br/har>

[2] Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012.