Coursera Machine Learning Project

In this project, we work with a large data set containing records of participants doing various physical exercises, and our task is to predict the type of exercise. There are a great many predictors available to us, usually from things like accelerometers, and gyro-meters.

The empirical work the data set is derived from can be found at <http://groupware.les.inf.puc-rio.br/har>.

We start by reading the dataset, which we read from a local file, since it is quite sizeable. We also load package caret, our preferred package in R for machine learning.

if(!require(caret))install.packages("caret")

## Loading required package: caret

## Warning: package 'caret' was built under R version 3.0.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.0.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.0.3

fittest<-read.csv("pml-training.csv",stringsAsFactors=F)  
fittest<-fittest[,-1] # delete first col with duplicate row nums

We notice that the data file is fairly messy. Many columns have a glut of missing values, and many represent sample statistics on the measures themselves (e.g. kurtosis, skewness). Also, there are a number of records which seem to have a single entry in a variable called "new window", not helpful to us.

We therefore delete the the columns that have too many missing values to be useful, and this will also reduce dimensionality, which is a problem in this dataset. We also remove columns that have too many blank entries - that is, too many (>5000) zeros. We also delete records which have "new window" = y, since these are usually sparse.

rmnw<-!fittest$new\_window=="yes"  
fittest<-fittest[c(rmnw),] # delete records with new\_window=yes  
wntg2<-which(colSums(is.na(fittest))>5000)  
fittest<-fittest[,-c(wntg2)]  
df<-fittest[,1:91]  
df[, sapply(df, is.character)]<-list(NULL)  
df<-cbind(df,fittest[92])  
fittest<-df  
rm(df)

Now we take our data set and create training and test sets. We note further that we have an additional 20 records to check our model on. This is ample cross-validation, and we will not create folds or other slices in the training set. Furthermore, one of the methods we report in this analysis is Random Forest, which has cross-validation and re-sampling built into it, so this can already be considered redundant for this purpose.

intrainfit<-createDataPartition(y=fittest$classe,p=.75,list=F)   
trainfit<-fittest[intrainfit,]  
testfit<-fittest[-intrainfit,]  
trainfit$classe<-as.factor(trainfit$classe)

We now consider our options for building a predictive model: we need to predict the type of physical activity, encoded in the 'classe' variable, which can take any of the categorical values A,B,C,D or E.

We therefore reject any of the linear modeling techniques (e.g. linear regression, or logistic regression), and opt instead for Support Vector Machines, and Random Forests.

In this dynamic data report, the svm modeling is re-created on the fly, but the Random Forest analysis proved to be too slow to repeat here. We report the analysis in table format below, but note that it is not re-created dynamically within this document.

\*\* Support Vector Machines

We load the package 'e1071', in order to use the svm method there. We conduct the svm procedure with the default arguments (although advised against, the results are good).

if(!require(e1071))install.packages("e1071")

## Loading required package: e1071

## Warning: package 'e1071' was built under R version 3.0.3

fitalllin<-svm(classe~.,data=trainfit[,8:56])

We now interrogate the model to ascertain the accuracy of our predictions.

# Check accuracy:  
pred <- predict(fitalllin, trainfit, decision.values = TRUE)  
predtable<-table(pred, trainfit$classe)  
prop.table(predtable,2)

##   
## pred A B C D E  
## A 0.9961014 0.0674077 0.0015911 0.0012706 0.0000000  
## B 0.0012183 0.9128720 0.0314240 0.0004235 0.0034014  
## C 0.0024366 0.0179276 0.9594272 0.0940280 0.0173847  
## D 0.0000000 0.0007171 0.0067621 0.9017366 0.0260771  
## E 0.0002437 0.0010757 0.0007955 0.0025413 0.9531368

xtable<-prop.table(predtable,2)  
sum(xtable[1,1],xtable[2,2],xtable[3,3],xtable[4,4],xtable[5,5])/5

## [1] 0.9447

We see that accuracy is above 90% for each of the types of physical activity, and if we average across the predictions we get average accuracy as 0.9447

This accuracy data is on the training set, and we need to check the accuracy on the test set. We report this below.

# Check accuracy:  
pred <- predict(fitalllin, testfit, decision.values = TRUE)  
predtable<-table(pred, testfit$classe)  
prop.table(predtable,2)

##   
## pred A B C D E  
## A 0.992685 0.083961 0.003580 0.003817 0.001134  
## B 0.002926 0.892357 0.048926 0.001272 0.002268  
## C 0.002195 0.021529 0.939141 0.082697 0.019274  
## D 0.000000 0.000000 0.005967 0.909669 0.022676  
## E 0.002195 0.002153 0.002387 0.002545 0.954649

xtable<-prop.table(predtable,2)  
sum(xtable[1,1],xtable[2,2],xtable[3,3],xtable[4,4],xtable[5,5])/5

## [1] 0.9377

We see that accuracy is above 89% for each of the types of physical activity, and if we average across the predictions we get average accuracy as 0.9377. We therefore consider the model cross-validated. Further testing with the additional 20 cases produced 100% accuracy.

\*\* Random Forests

As declared earlier, we do not report this in dynamic document form i.e. re-created by knitr from an .Rmd file, as the analysis takes several hours on a very fast computer with multiple processors. However we did run the Random Forest analysis, and we report the results below in non-dynamic form.

Record of analysis from console window in RStudio:

SCRIPT

fitalllin<-train(classe~.,data=trainfit[,8:56],method="rf") pred<-predict(fitalllin,trainfit) table(pred,trainfitclasse)/length(testfit$classe)

OUTPUT

pred

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 4104 | 0 | 0 | 0 | 0 |
| B | 0 | 2789 | 0 | 0 | 0 |
| C | 0 | 0 | 2514 | 0 | 0 |
| D | 0 | 0 | 0 | 2361 | 0 |
| E | 0 | 0 | 0 | 0 | 2646 |

[1] 1

We can see from this that the average accuracy for the Random Forest is perfect. We test it now on the cross-validation testing set.

SCRIPT

pred2 <- predict(fitalllin, newdata=testfit) sum(pred2==testfitclasse)

OUTPUT

[1] 0.9902124

Accuracy with the Random Forest model is therefore clearly extremely good. Predictions are almost perfect.

We leave the analysis as it is at this point.