Machine learning project

Damien Chevalier

21 août 2015

Overview : 6 participants were asked to perform barbell lifts correctly and incorrectly in 5 different waysgoal. The aim is to use data from accelerometers on the belt, forearm, arm, and dumbell to predict (with machine learning) which exercice they are doing and if they are doing it well.

1) Load the data and first study and cleaning

```
library(caret)
library(randomForest)
library(gbm)
```

```
pml<- read.csv("pml-training.csv",header = TRUE, sep = ";", stringsAsFactors = TRUE, dec = ".", na.stri
```

We can see with the str() or summary() function that we get 19119 test and that several columns are without interest.

I decide to suppress - suppress column whith lots of NA or empty data - suppress the time elements (ie : 7 first column) because our data are not time series (each row is a test)

```
Nacol<-apply(pml,2,function(x) sum(is.na(x)))
pml1<-pml[,Nacol[]==0]
pml1<-pml1[,8:ncol(pml1)]
```

2) Preprocessing first on a small part

I choose to create fist a small training fold (10%) in order to test the best algorithm. In a second time, I will run again the algorithm on higer part (70%)

```
inTrain<-createDataPartition(y=pml1$classe, p=0.1, list = FALSE)
training<-pml1[inTrain,]
testing<-pml1[-inTrain,]
```

Then I check the covariance between predictors in order to supress redondant variables

```
nsv<-nearZeroVar(training,saveMetrics=TRUE)
```

After study no redondant variables appears.

3) Modelling

I don't use the linear modelling because we have to predict factor variable (A,B,C,D,E,F)

3.1) Attempt with trees

```
modfit1<-train(classe ~., method = "rpart", data = training)
confusionMatrix(testing$classe,predict(modfit1,testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4290  305  361    0   66
##          B 1369  926 1122    0    0
##          C 1357  126 1596    0    0
##          D 1061  730 1103    0    0
##          E  404  210 1096    0 1536
##
## Overall Statistics
##
##                Accuracy : 0.4728
##                  95% CI : (0.4654, 0.4802)
##     No Information Rate : 0.4803
##     P-Value [Acc > NIR] : 0.9778
##
##                   Kappa : 0.3148
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5058  0.40313  0.30239       NA  0.95880
## Specificity            0.9202  0.83784  0.88021   0.8361  0.89350
## Pos Pred Value         0.8542  0.27100  0.51835       NA  0.47320
## Neg Pred Value         0.6683  0.90373  0.74744       NA  0.99542
## Prevalence             0.4803  0.13008  0.29890   0.0000  0.09072
## Detection Rate         0.2429  0.05244  0.09038   0.0000  0.08699
## Detection Prevalence   0.2844  0.19351  0.17437   0.1639  0.18383
## Balanced Accuracy      0.7130  0.62049  0.59130       NA  0.92615
```

3.2) Attempt with bagging

```
fitControl2 <- trainControl(method = "repeatedcv",number = 10,repeats = 25)
modfit2<-train(classe ~., method = "treebag", data = training, trcontrol = fitControl2)
confusionMatrix(testing$classe,predict(modfit2,testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4906   39   35   33    9
##          B  173 3091  110   27   16
##          C   18  106 2912   38    5
##          D   32   23  106 2711   22
##          E   14   38   49   82 3063
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9448
##                  95% CI : (0.9413, 0.9481)
##     No Information Rate : 0.2913
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9301
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9539   0.9375   0.9066   0.9377   0.9833
## Specificity            0.9907   0.9773   0.9884   0.9876   0.9874
## Pos Pred Value         0.9769   0.9046   0.9458   0.9368   0.9436
## Neg Pred Value         0.9812   0.9855   0.9794   0.9878   0.9964
## Prevalence             0.2913   0.1867   0.1819   0.1637   0.1764
## Detection Rate         0.2778   0.1750   0.1649   0.1535   0.1735
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9723   0.9574   0.9475   0.9627   0.9854
```

3.3) Attempt with randomforest

```
set.seed(25)
modfit4<-randomForest(classe ~., data = training, ntree = 100, mtry = 35 )
confusionMatrix(testing$classe,predict(modfit4,testing))
```

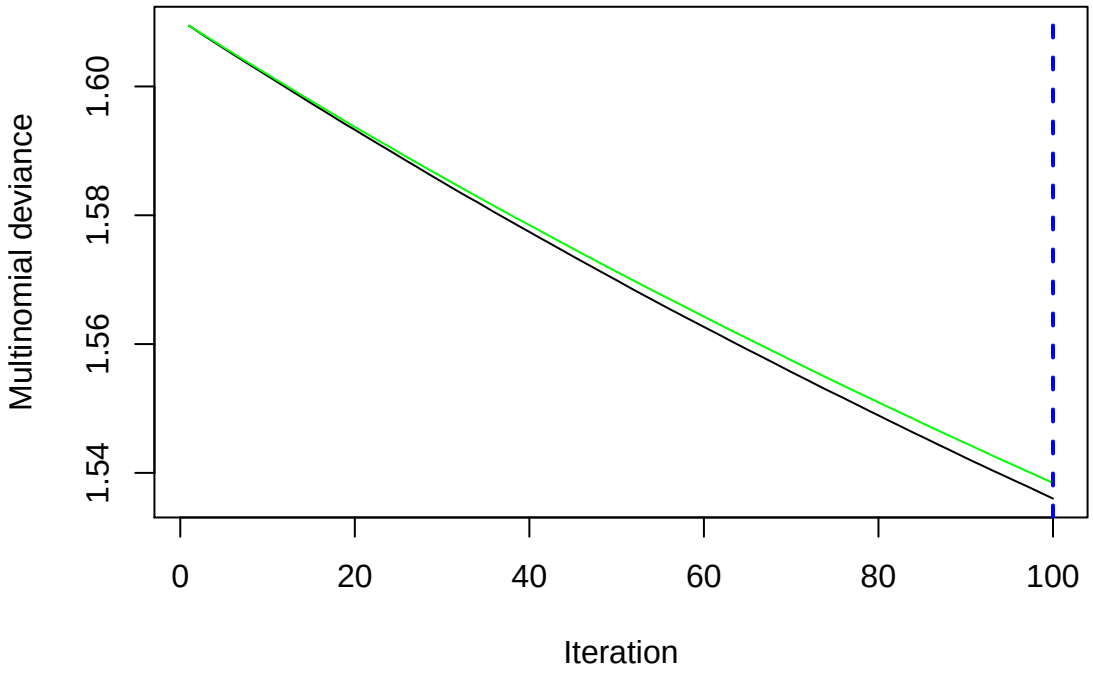
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4945   29   28   14    6
##          B  157 3112  124   16    8
##          C   22  108 2903   43    3
##          D   43   14  117 2693   27
##          E   10   44   19   61 3112
##
## Overall Statistics
##
##                Accuracy : 0.9494
##                  95% CI : (0.9461, 0.9526)
##     No Information Rate : 0.2932
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.936
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9552   0.9410   0.9097   0.9526   0.9861
## Specificity            0.9938   0.9787   0.9878   0.9864   0.9908
## Pos Pred Value         0.9847   0.9107   0.9428   0.9305   0.9587
```

```
## Neg Pred Value         0.9816   0.9863   0.9802   0.9909   0.9969
## Prevalence             0.2932   0.1873   0.1807   0.1601   0.1787
## Detection Rate         0.2800   0.1762   0.1644   0.1525   0.1762
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9745   0.9599   0.9488   0.9695   0.9884
```

The accuracy is excellent at this point and the calcul time is short (<1s)

3.4) Attempt with boosting

```
gbm1<-gbm(classe~., data = training, n.trees =  100, cv.folds = 30, distribution = "multinomial")
best.iter <- gbm.perf(gbm1,method="cv")
```




```
pred<-predict(gbm1, testing, best.iter ,type='response')
pred_class <- apply(pred, 1, which.max)
pred_class[pred_class==1]<-"A"
pred_class[pred_class==2]<-"B"
pred_class[pred_class==3]<-"C"
pred_class[pred_class==4]<-"D"
pred_class[pred_class==5]<-"E"

confusionMatrix(testing$classe,pred_class)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2530  121  995 1310   66
##          B  787 1309  602  719    0
##          C  543  124 1434  978    0
##          D  426  178  322 1968    0
##          E  616  441  326  327 1536
##
## Overall Statistics
##
##                Accuracy : 0.4971
##                  95% CI : (0.4897, 0.5045)
##     No Information Rate : 0.3003
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3674
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5161  0.60239  0.38978   0.3712  0.95880
## Specificity            0.8046  0.86387  0.88232   0.9251  0.89350
## Pos Pred Value         0.5038  0.38308  0.46574   0.6800  0.47320
## Neg Pred Value         0.8123  0.93933  0.84601   0.7742  0.99542
## Prevalence             0.2776  0.12306  0.20835   0.3003  0.09072
## Detection Rate         0.1433  0.07413  0.08121   0.1115  0.08699
## Detection Prevalence   0.2844  0.19351  0.17437   0.1639  0.18383
## Balanced Accuracy      0.6604  0.73313  0.63605   0.6481  0.92615
```

Conclusion of accuracy

- Trees : 0.48
- Bagging : 0.93
- RandomForest : 0.94
- Boosting : 0.5

To conclude the best model to developp is Randomforest

3) Random forest study

We extend the study by enlarging the training file and increasing the number of trees

```
inTrain2<-createDataPartition(y=pml1$classe, p=0.7, list = FALSE)
training2<-pml1[inTrain2,]
testing2<-pml1[-inTrain2,]

set.seed(25)
modfit5<-randomForest(classe ~., data = training2, ntree = 1000, mtry = 35 )
confusionMatrix(testing2$classe,predict(modfit5,testing2))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    1    1    1    0
##          B    8 1124    6    1    0
##          C    0    5 1017    4    0
##          D    0    1   10  950    3
##          E    0    0    2    3 1077
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.9896, 0.9943)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9901
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9938   0.9817   0.9906   0.9972
## Specificity            0.9993   0.9968   0.9981   0.9972   0.9990
## Pos Pred Value         0.9982   0.9868   0.9912   0.9855   0.9954
## Neg Pred Value         0.9981   0.9985   0.9961   0.9982   0.9994
## Prevalence             0.2853   0.1922   0.1760   0.1630   0.1835
## Detection Rate         0.2839   0.1910   0.1728   0.1614   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9973   0.9953   0.9899   0.9939   0.9981
```

The selection of random forest algorithm show no need to perform a cross validation (it's include in the algorithm)

Finaly we get an accuracy of 99.3% which is very good for this type of exercise

4) final test on the 20 samples

We use the same preprocessing as the training test set (suppression of columns) and use the predict function on the final model (modfit5)

```
pmltest<- read.csv("pml-testing.csv",header = TRUE, sep = ",", stringsAsFactors = TRUE, dec = ".", na.s
pmltest2<-pmltest[,Nacol[]==0]
pmltest2<-pmltest2[,8:ncol(pmltest2)]

answers<-predict(modfit5,pmltest2)

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(answers)
```

20/20 on assessment !