The prediction of the way of exercises

AuthorNotShown

December 22, 2015

Abstract

In this study, whether people are doing exercise in the correct way is predicted based on the collected sensor data during the weight lifting exercises. After tuing the sporting vector machine model, an ovrall accuracy of >99% is achieved.

Read data and clean data

The data is first read into R and all the variables which have missing values in more than half of the observations are skipped in teh modelling.Also, the time stamps in the dataset is also skipped. The new data is saved in variable “hdata”.

```r
fdata<- read.csv("pml-training.csv",na.strings=c("NA","NaN", " ",""))
p<- sapply(fdata,function(x) sum(is.na(x)))
gdata<- fdata[,p<dim(fdata)[1]/2]
hdata<- gdata[,-c(1:5)]
```

Data partitioning

10% of the data is randomly selected for final testing of the model, saved in “testing” variable. For the remaining 90% of the data, 20% are randomly sampled for model validation process, saved in variable “validationData”. The remaining 80% of the data is used for training, saved in variable “training”.

```r
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
library(e1071)
inTrain<- createDataPartition(y=hdata$classe,p=0.9,list=FALSE)
testing<- hdata[-inTrain,]
totaltraining<- hdata[inTrain,]
inTrain2<- createDataPartition(totaltraining$classe,p=0.8,list=FALSE)
training<- totaltraining[inTrain2,]
validationData<- totaltraining[-inTrain2,]
```

Initial Modelling

A “linear discrimation analsyis” (lda) model is performed to predict the response “classe” variable in the training data set, using all other variables.

```
mlda<- train(classe~.,data=training,method="lda")
```

```
## Loading required package: MASS
```

```
confusionMatrix(predict(mlda,validationData),validationData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 808  88  55  34  22
##          B  50 460  62  20 103
##          C  61  79 405  71  52
##          D  77  28  79 435  66
##          E   8  28  15  19 406
##
## Overall Statistics
##
##                Accuracy : 0.712
##                  95% CI : (0.6967, 0.7269)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.636
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8048   0.6735   0.6575   0.7513   0.6256
## Specificity            0.9213   0.9175   0.9098   0.9153   0.9757
## Pos Pred Value         0.8024   0.6619   0.6063   0.6350   0.8529
## Neg Pred Value         0.9223   0.9214   0.9263   0.9494   0.9205
## Prevalence             0.2843   0.1934   0.1745   0.1640   0.1838
## Detection Rate         0.2288   0.1303   0.1147   0.1232   0.1150
## Detection Prevalence   0.2852   0.1968   0.1892   0.1940   0.1348
## Balanced Accuracy      0.8630   0.7955   0.7836   0.8333   0.8006
```

Unfortunately, the overall accuracy is not so good, around 70% for the validation dataset. This indicates the data may not be linear. Therefore, a “supporting vector machine” (svm) model is then applied to predict the response “classe” with all other variables int eh data set. The overall accuray has improved to around 95% for the validation data set.

```
msvm<- svm(classe~.,data=training)
confusionMatrix(predict(msvm,validationData),validationData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 998  42   0   1   0
##          B   0 631  15   0   0
##          C   5  10 594  54  14
##          D   0   0   7 523  13
##          E   1   0   0   1 622
##
## Overall Statistics
##
##                Accuracy : 0.9538
##                  95% CI : (0.9464, 0.9605)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9415
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9940   0.9239   0.9643   0.9033   0.9584
## Specificity            0.9830   0.9947   0.9715   0.9932   0.9993
## Pos Pred Value         0.9587   0.9768   0.8774   0.9632   0.9968
## Neg Pred Value         0.9976   0.9820   0.9923   0.9813   0.9907
## Prevalence             0.2843   0.1934   0.1745   0.1640   0.1838
## Detection Rate         0.2826   0.1787   0.1682   0.1481   0.1762
## Detection Prevalence   0.2948   0.1830   0.1917   0.1538   0.1767
## Balanced Accuracy      0.9885   0.9593   0.9679   0.9483   0.9789
```

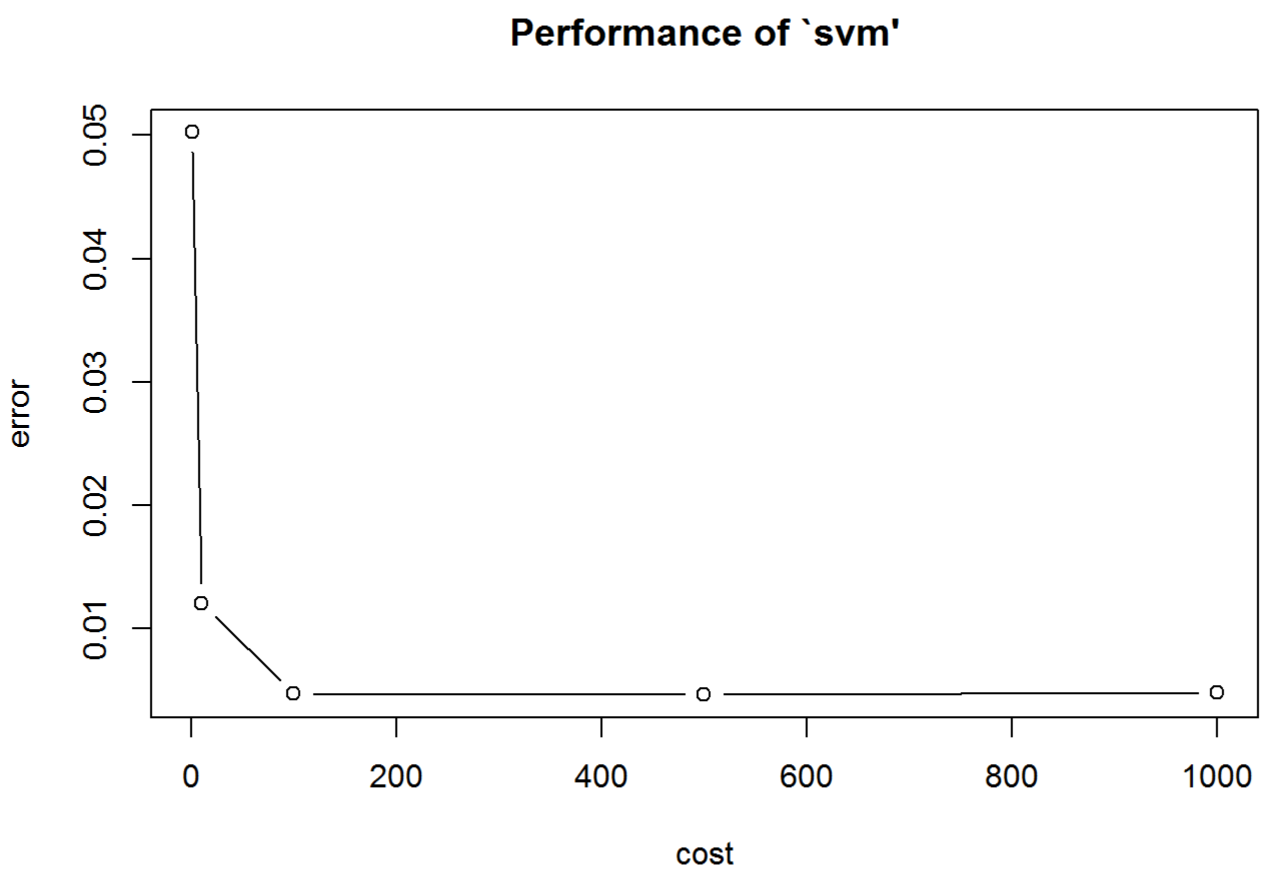
The selection of parameters

For SVM model, the most important parameteris the cost. Therefore, several SVM models with different cost values are modelled using cross-validation in the training data set.

```
obj<- tune.svm(classe~.,data=training,cost=c(1,10,100,500,1000))
print(obj)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   500
##
## - best performance: 0.004670763
```

```
plot(obj)
```



```
confusionMatrix(predict(obj$best.model,validationData),validationData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1001    1    0    0    0
##          B    2  679    3    0    0
##          C    0    1  610    3    0
##          D    0    1    3  571    1
##          E    1    1    0    5  648
##
## Overall Statistics
##
##                Accuracy : 0.9938
##                  95% CI : (0.9906, 0.9961)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9921
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9941   0.9903   0.9862   0.9985
## Specificity            0.9996   0.9982   0.9986   0.9983   0.9976
## Pos Pred Value         0.9990   0.9927   0.9935   0.9913   0.9893
## Neg Pred Value         0.9988   0.9986   0.9979   0.9973   0.9997
## Prevalence             0.2843   0.1934   0.1745   0.1640   0.1838
## Detection Rate         0.2835   0.1923   0.1728   0.1617   0.1835
## Detection Prevalence   0.2838   0.1937   0.1739   0.1631   0.1855
## Balanced Accuracy      0.9983   0.9962   0.9944   0.9922   0.9980
```

From the result, it can be seen the best result coming from with a cost value of 500, shown in the graph above. The test accuracy is >99.5% for the validation data set, which is satisfacotary.

The “out-of-sample” error in the testing dataset

```
confusionMatrix(predict(obj$best.model,testing),testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   A   B   C   D   E
##          A 557   0   0   0   0
##          B   1 378   1   0   0
##          C   0   0 341   1   0
##          D   0   1   0 318   0
##          E   0   0   0   2 360
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9969          
##                  95% CI : (0.9933, 0.9989)
##     No Information Rate : 0.2847          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9961          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9974   0.9971   0.9907   1.0000
## Specificity            1.0000   0.9987   0.9994   0.9994   0.9988
## Pos Pred Value         1.0000   0.9947   0.9971   0.9969   0.9945
## Neg Pred Value         0.9993   0.9994   0.9994   0.9982   1.0000
## Prevalence             0.2847   0.1934   0.1745   0.1638   0.1837
## Detection Rate         0.2842   0.1929   0.1740   0.1622   0.1837
## Detection Prevalence   0.2842   0.1939   0.1745   0.1628   0.1847
## Balanced Accuracy      0.9991   0.9980   0.9982   0.9950   0.9994
```

From the result, it can be seen that the test accuracy is still >99%.