Practical ML Course Project

Thao Tran

21 May 2015

Overview

I trained a machine learning model on the "Weight Lifting Exercise" data set to predict whether a participant performs a "Unilateral Dumbbell Biceps Curl" correctly or not. (See http://groupware.les.inf.puc-rio.br/har for more details.)

Exploratory Data Analysis & Feature Selections

In the study, four groups of sensors were used to monitor participant motion: sensors on 1) the dumbbell, 2) the forearm, 3) the arm, and 4) the waist/belt. The output of these sensor groups became the input data for the machine learning process. In all, there were 159 available parameters.

A quick `summary` of the data showed that many parameters (100 of them) contained very little data, or none at all. In addition, only the parameters related to the four sensor groups above seemed to me to be worth keeping.

After filtering these two populations out, there remained the following types of parameters:

- 3 types related to rotation (roll/pitch/yaw).
- 3 types related to the acceleration sensor (x, y, z) + a total accel value.
- 3 types related to the gyro (x, y, z).
- 3 types related to the magnetic sensor (x, y, z).

This tallied up to: 13 parameter types $\times$ 4 sensor groups = 52 parameters, all of which I kept as features.

Model Building

I used the "caret" framework in R to conduct the machine learning process. Part of this process is the selection of a machine learning algorithm. "Caret" supports many strategies, but because of limited computer resources, I could only use methods that were quick and undemanding. In the end, I studied four models:

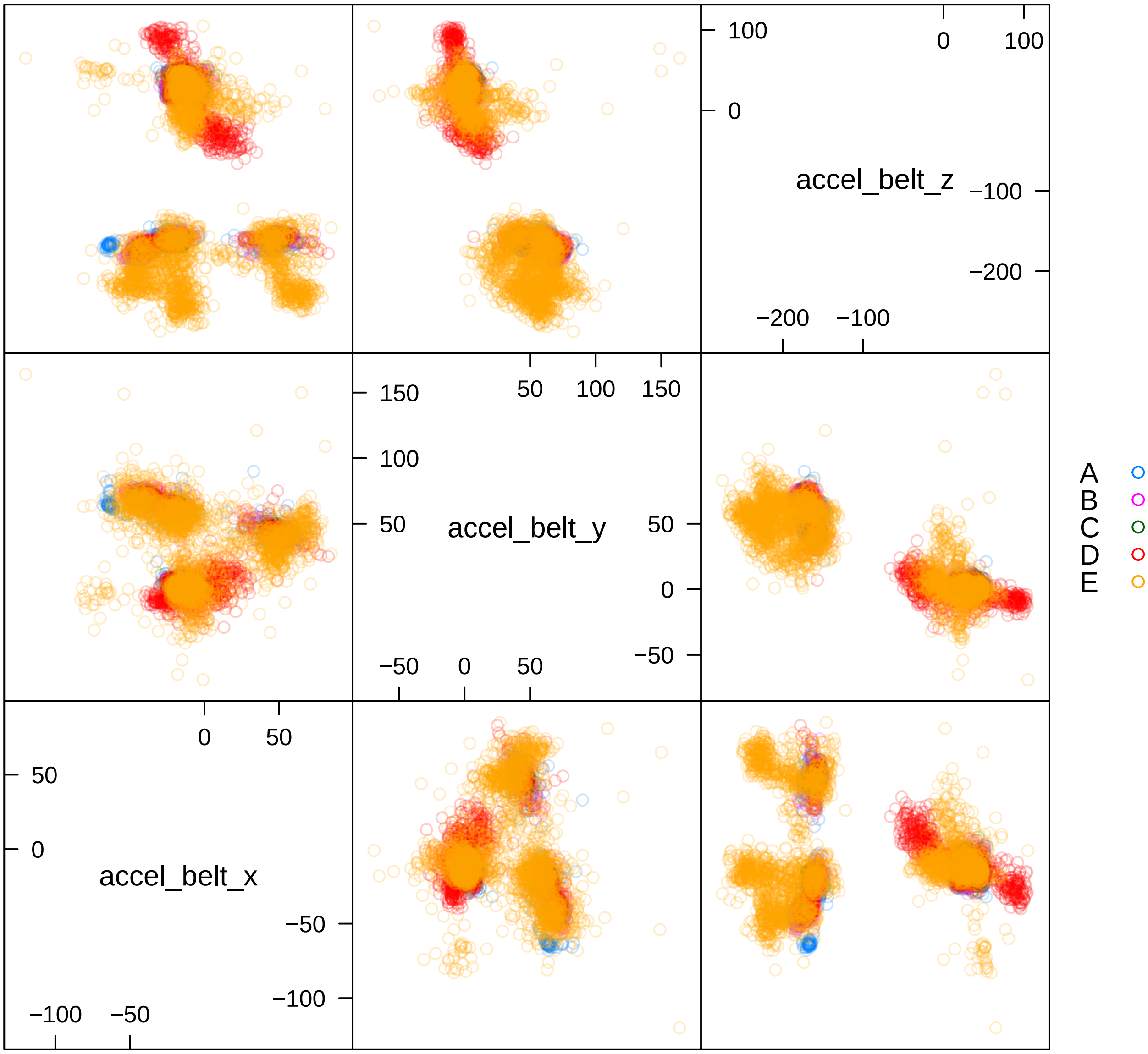
1. The Classification and Regression Tree (CART) model using recursive partitioning method ("rpart" on caret).
2. The linear discriminant analysis (parametric model) method ("lda" on caret).
3. The quadratic discriminant analysis method ("qda" on caret).
4. And the robust quadratic discriminant analysis method ("QdaCov" on caret).

Other methods, such as Naive Bayes ("nb"), bagging ("bagEarth"), Boost ("gbm"), were impossible to run on my computer because it lacked enough memory.

The discriminant analysis methods were especially appealing to me because being parametric model based, they do not take up so much memory, only cpu. Even then, the computations were not intensive.

Also, these methods assume the predictors follow a Gaussian distribution. Luckily, the assumption works fairly well with the predictors in this Weight Lifting Exercise data set. (See Figure 1.)

Fig 1: Distribution of Sample Predictors



Scatter Plot Matrix

Training & Cross Validation

The caret framework made machine training simple, especially for the above methods. Please see the appendix for the code I used.

For the purpose of model comparison and out of sample error estimate, I used the k-fold cross validation (CV) technique to subdivide the training set "on-the-fly". I chose a k-fold setting of: `number` = 10, and `repeat` = 10. This meant that the CV step was repeated 10 times. Each time, 10% of the data was used for testing, and 90% was used for training. The location of the 10% testing block was shifted each time, so that after 10 steps, it traversed the whole training set.

As per classification problems, the training process produced an average "Accuracy" number. This number came from the cross validation phase where it compared the predicted classifications to the known classifications. Of course the predicted and known values came from the 10% testing blocks.

One can use these "Accuracy" numbers to compare the performance of the models to each other. In additon, because they were produced from a "test" set, these Accuracy numbers should be representative of an actual test run. Thus, they also provided the expected out of sample error estimate.

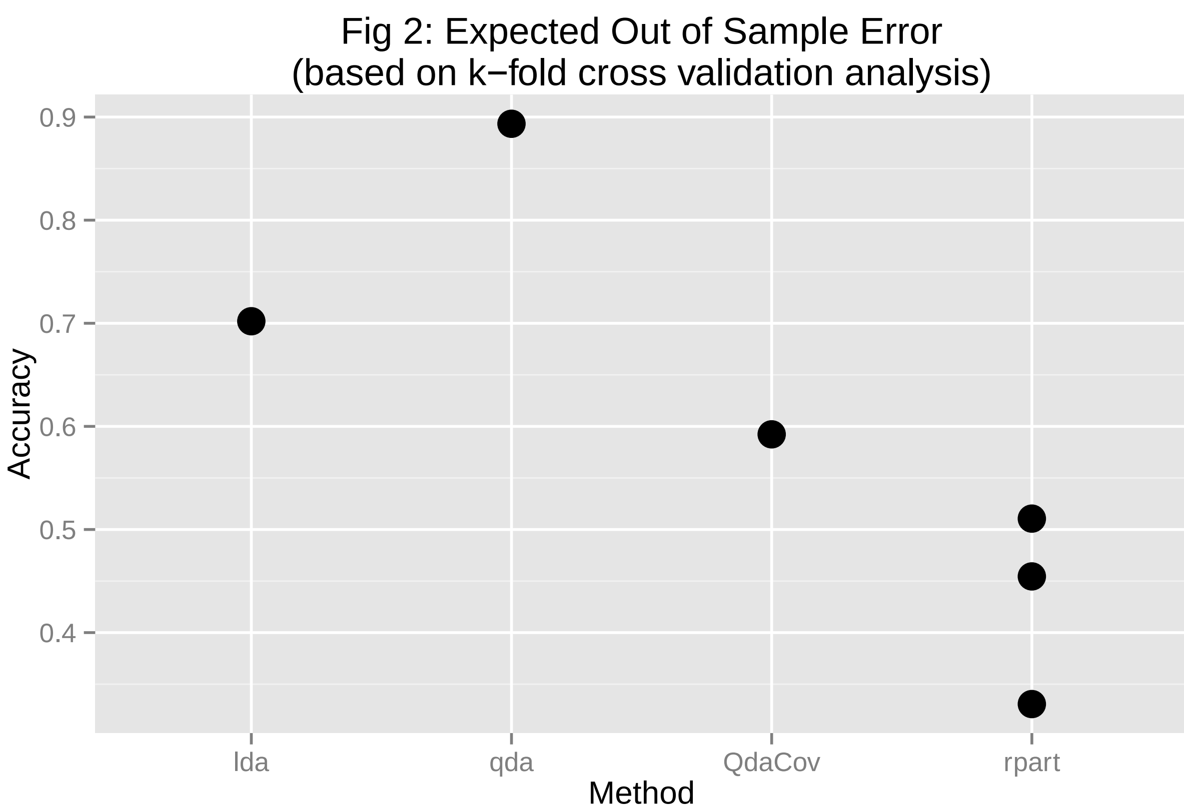
Result

From the four methods in the study, the quadratic discriminant analysis ("qda") did much better than the other three methods. (See Figure 2.) Surprisingly, in this study the robust qda method "QdaCov" performed worse than "qda", and even "lda".

The "qda" method gave an accuracy of 89.3%. A prediction using this method should be ≈ 90% accurate. In other words, the expected out of sample error is roughly 10% with this method.

In fact, with the "pml-testing.csv" set, the accuracy was 95%.

—




```
##      X Method nFeature  Accuracy
## 1   1    lda       52 0.7019676
## 2   2    qda       52 0.8934864
## 3   3  rpart       52 0.5105470
## 4   4  rpart       52 0.4544924
## 5   5  rpart       52 0.3307055
## 11 11 QdaCov       52 0.5922953
```

(NB: the "rpart" method reported three Accuracy numbers from its recursive run. The best number should be used as the final result.)

Appendix: R Code

```
library(lubridate)
library(plyr)
library(dplyr)
library(ggplot2)
library(caret)
library(doMC)

registerDoMC(cores=3)
```

```
training <- read.csv("./data/pml-training.csv")
testing <- read.csv("./data/pml-testing.csv")

# features selection...
fullColNames <- names(training)

# filter out columns with too many NAs or blanks
naCounts <- data.frame(fullColNames,
                       count=sapply(
                           fullColNames,
                           function(x) sum(is.na(training[,x])
                                           |grepl("^$", training[,x]))))
colNames <- as.character(naCounts[naCounts$count<1000, ]$fullColNames)

# grab columns with interesting names
varDumbbell <- grep("dumbbell", colNames, value=T)
varForearm <- grep("forearm", colNames, value=T)
varBelt <- grep("belt", colNames, value=T)
varArm <- grep("_arm", colNames, value=T)
goodVars <- c(varDumbbell, varForearm, varBelt, varArm)
fullFormula <- formula( paste("classe ~", paste(goodVars, collapse = " + ")) )

doFit <- function(tDf, tFormula, method) {
    kFoldControl <- trainControl(method="cv", number=10, repeats=10)
    set.seed(343243)
    fit <- NULL
    if (method %in% c("lda", "qda", "rpart", "QdaCov")) {
        fit <- train(tFormula, data=tDf, method=method, trControl=kFoldControl)
        #confusionMatrix info
        ans <- predict(fit, newdata=tDf)
        confMat <- confusionMatrix(ans, tDf$classe)
        print(paste("fit:", fit$result["Accuracy"],
                    "confMat:", confMat$overall["Accuracy"]))
    }
    return (fit)
}

keepErrStats <- function(errors, fit) {
    if (!is.numeric(errors)) {
        errors <- unique(
            bind_rows( errors, data.frame(
```

```r
                Method=fit$method,
                # need to hack for QdaCov
                nFeature=ifelse(fit$method=="QdaCov", 52, length(fit$finalModel$xNames)),
                Accuracy=fit$result["Accuracy"]) ) )
    } else {
        errors <- data.frame(
                Method=fit$method,
                nFeature=length(fit$finalModel$xNames),
                Accuracy=fit$result["Accuracy"])
    }
    return (errors)
}

if (!exists("errors")) errors <- -1
rpartFit <- doFit(training, fullFormula, "rpart")
errors <- keepErrStats(errors, rpartFit)
ldaFit <- doFit(training, fullFormula, "lda")
errors <- keepErrStats(errors, ldaFit)
qdaFit <- doFit(training, fullFormula, "qda")
errors <- keepErrStats(errors, qdaFit)
qdaCovFit <- doFit(training, fullFormula, "QdaCov")
errors <- keepErrStats(errors, qdaCovFit)

ans <- as.character(predict(qdaFit, newdata=testing))
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(ans)
```