Predict the Amount of Active Ingredient of Pharmaceutical Tablets Data

Introduction

In this report a regularized logistic regression model with $L1$ penalty function is used to predict the amount of active ingredient given the input near infrared spectrometry (NIR) measurements of pharmaceutical tablets data.

The file consists of matrices such as:

- x = the input data for training (80%) and validation (20%)
- y = the target class labels (binary) for training (80%) and validation (20%)
- x_test = the input data for test
- y_test – the target class labels for test (binary)

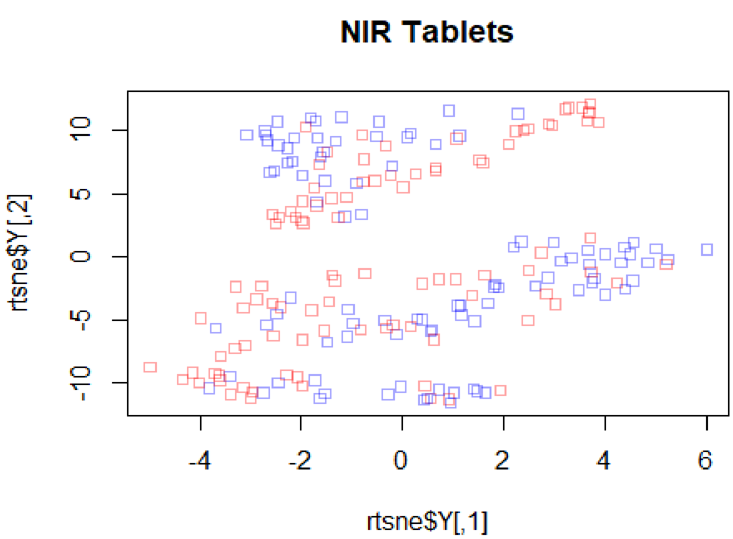
Here is the process detail of training, validation and performance assessment of the data.

Optimal Regularization Parameter

To get the best regularization parameter, here is step by step of tuning procedure:

1. **Standardizing the input data**
   When the file is loaded, the input data such x and x_test are large matrices. Meanwhile the target class data (y and y-test) are in binary format. So, the input data is standardized with 'scale' function.

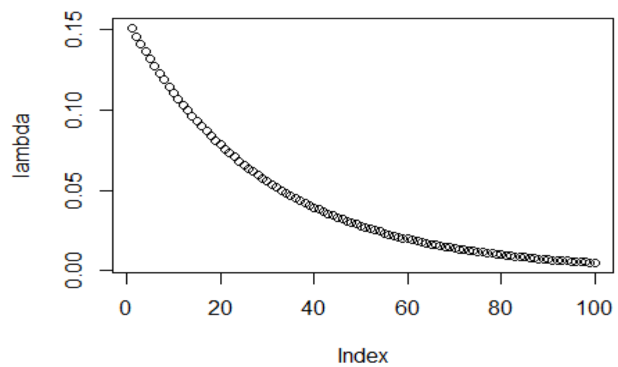
2. **Plot the data**



By plotting the data, we can see the sparse nature of the data.

3. Set the Hyperparameter
In this project, I used the range of values for the regularization hyperparameter around the interval of [0.005, 0.150]. The interval of hyperparameter (Lambda) is set so we can find the best value with the minimum error.

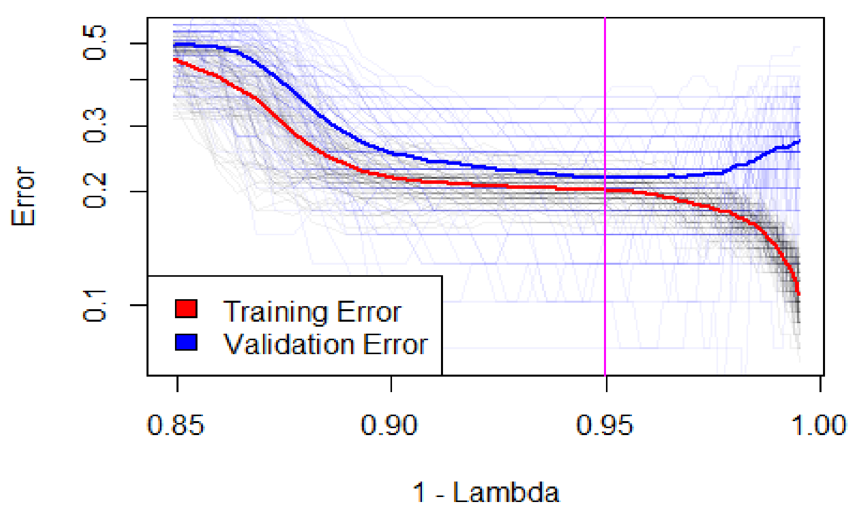
```
pt <- seq(-1.89,-5.3, length = 100)
lambda <- exp(pt)
```



The Assessment of the Predictive Performance

To assess the quality of prediction model, I put the data into 3 parts (training, validation and test) with 100 of repetitions. I compared the error result between the training and validation data.

1. **Compare the training and validation error**

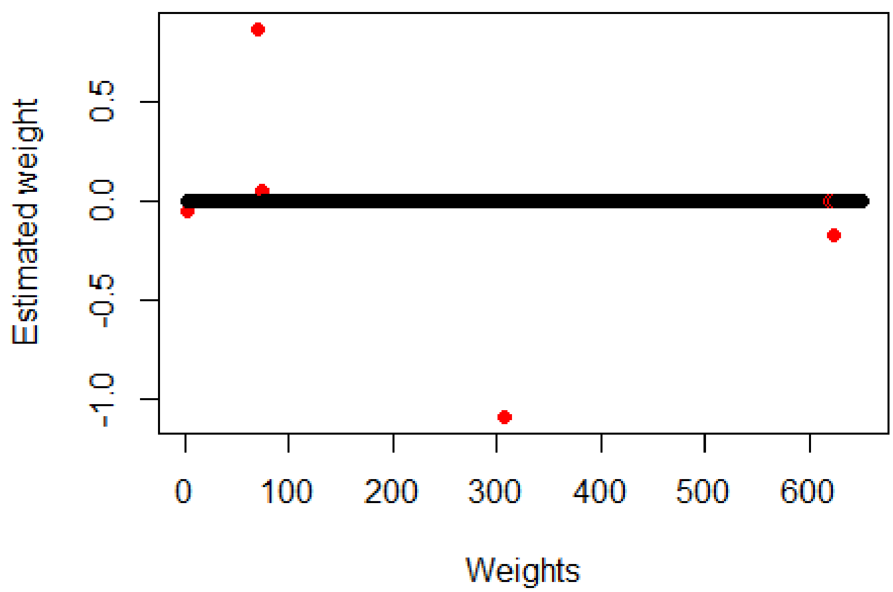



```
bestlambda

## [1] 0.05017581
```

From the graph, starting with lambda = 0.15, the data clearly need more regularization since the both errors are in their maximum value. This means that the bias at that point is very high, and it needs a better lambda. As decreasing the value of lambda a bit, the error is getting smaller and the validation error reach the minimum error with the optimal lambda = 0.05. After the validation error line is reaching the minimum error, the line of training error is getting closer to 0.1 as an indication of overfitting. With lambda closer from 0.05 to 0.005, the variance is increasing and the gap between two lines is growing. In my opinion based on the data, the model with best lambda = 0.05 (the optimum with pink line) is good to lower the variance at the cost of some bias.

```
misclassification_error(y_test, ytest)

## [1] 0.2869565
```

From the data, I generated the misclassification error as an estimated rate of how often the prediction will be wrong and it is equal to 28.7%. Means that the data is more than 70% accurate, which is still can be considered as a good.



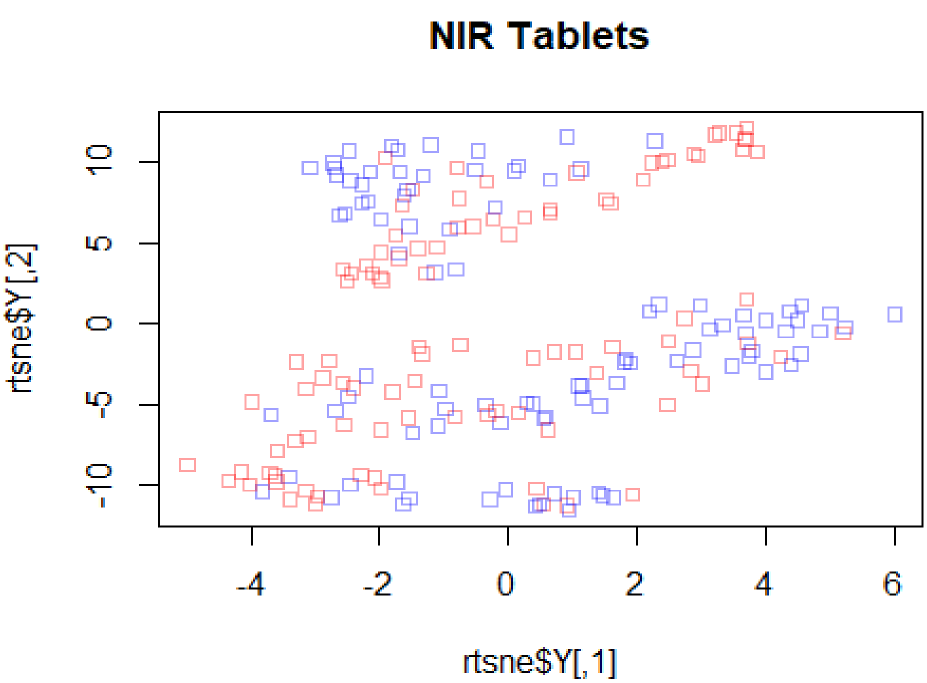
The nature of L1 is pushing the weight towards 0 or it can be called as a sparsity behavior. The purpose of pushing the weights (as a penalty) turn to reduce the model complexity, making our model simpler. The plot shows that there are 6 dots of weight while the rest are not active.

APPENDICES

The code is adapted from:

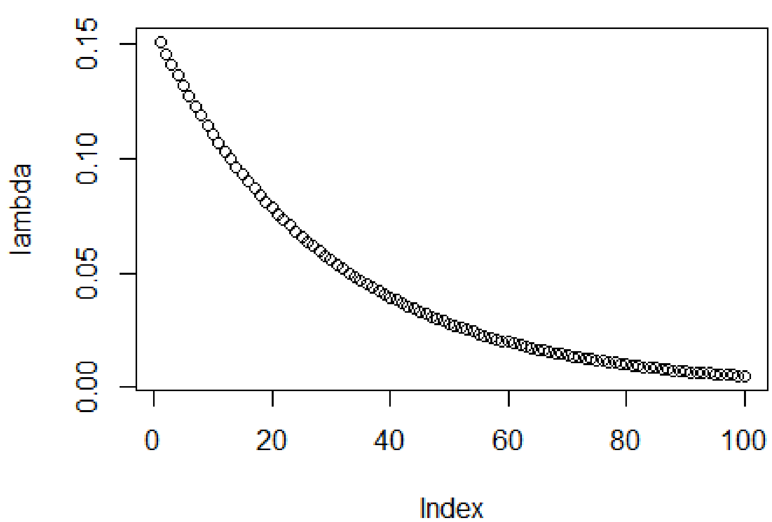
Machine Learning and AI – Practical Lab 2 (with additional modification)

```
load("data_nir_tablets.Rdata")
library(Rtsne)
range(x)

## [1] 2.350130 6.563778

#standardizing
x <- scale(x)
x_test <- scale(x_test)
rtsne <- Rtsne(x, perplexity = 35) #set the rtsne with range between 5-50
cols <- c("red", "blue")[y+1]

plot(rtsne$Y, pch = 22, col = adjustcolor(cols, 0.35), main = "NIR Tablets")
```



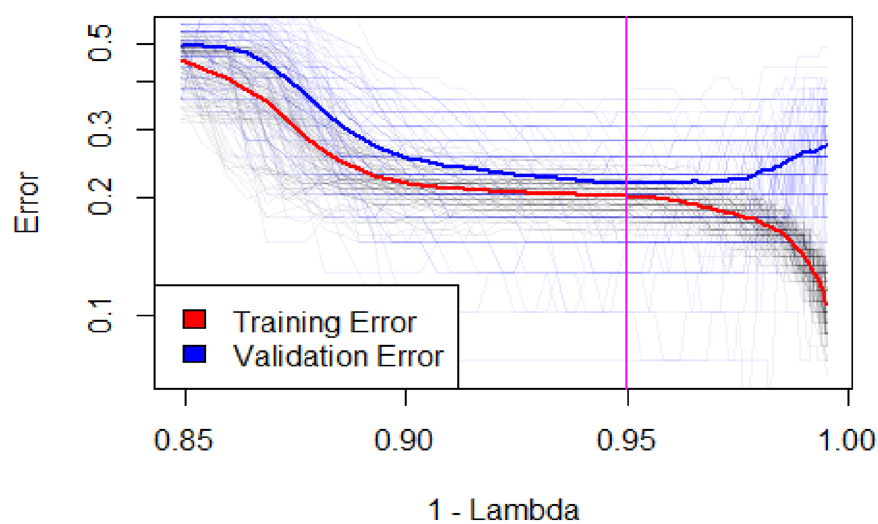

```
library(e1071)
#misclassification error function
misclassification_error <- function(y, yhat) {
 tab <- table(y, yhat)
 1 - classAgreement(tab)$diag }

library(glmnet)

## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2

# set training, validation and test data sizes
TOT <- nrow(x)
#80:20 rule
N <- floor(TOT*0.8) #training
L <- floor(TOT*0.2) #validation
test <- nrow(x_test) #test
# number of replication
B <- 100
#class checking
table(y)

## y
##  0  1
## 98 97

#set tau
tau <- 0.5
S <- 100

#set the lambda
pt <- seq(-1.89,-5.3, length = 100)
lambda <- exp(pt)

plot(lambda)
```




```
library(glmnet)
#define the errors
error_training <- error_validation <- matrix(NA, B, S)
error_test <- lambda_best <- rep(NA, B)

#training and validation for loop
```

```r
for ( b in 1:B ) {
 # sample train and validation data
 train <- sample(1:TOT, N)
 val <- setdiff(1:TOT, c(train))
 # train the model
 fit <- glmnet(x[train,], y[train], family = "binomial", alpha = 1, lambda = lambda)
 # obtain predicted classes for training data
 p_train <- predict(fit, newx = x[train,], type = "response")
 y_train <- apply(p_train, 2, function(v) ifelse(v > tau, 1, 0))
 # obtain predicted classes for validation data
 p_val <- predict(fit, newx = x[val,], type = "response")
 y_val <- apply(p_val, 2, function(v) ifelse(v > tau, 1, 0))
 # estimate misclassification error
 error_training[b,] <- sapply( 1:S, function(s) misclassification_error(y[train], y_train[,s]) )
 error_validation[b,] <- sapply( 1:S, function(s) misclassification_error(y[val], y_val[,s]) )
 # select lambda which minimizes misclassification error on validation data
 best <- which.min(error_validation[b,]) }

#plot the training vs validation error lines
matplot(x = 1-lambda, t(error_training), type = "l", lty = 1, ylab = "Error", xlab = "1 - Lambda", col = adjustcol
or("black", 0.05), log = "y")
matplot(x = 1-lambda, t(error_validation), type = "l", lty = 1, col = adjustcolor("blue", 0.05), add = TRUE, log
= "y")
lines(1-lambda, colMeans(error_training), col = "red", lwd = 2)
lines(1-lambda, colMeans(error_validation), col = "blue", lwd = 2)
legend("bottomleft", legend = c("Training Error", "Validation Error"),
   fill = c("red", "blue"))


# best lambda
bestlambda <- lambda[ which.min( colMeans(error_validation) ) ]
abline(v = 1 - bestlambda, col = "magenta")
```



```r
bestlambda

## [1] 0.05017581

# train model with optimal hyperparameter lambda to the data
fit <- glmnet(x, y, family = "binomial", lambda = bestlambda)

# compute misclassification error
p_test <- predict(fit, newx = x_test, type = "response")
ytest <- apply(p_test, 2, function(v) ifelse(v > tau, 1, 0))
table(y_test, ytest)

##       ytest
## y_test   0   1
##      0 142  88
##      1  44 186

misclassification_error(y_test, ytest)

## [1] 0.2869565

w_hat <- coef(fit, s = bestlambda)
cols <- ifelse(w_hat != 0, "red", "black")
plot(w_hat, col = cols, pch = 19,
    xlab = "Weights", ylab = "Estimated weight")
```