Comparison for Effective of Five Classification Methods by Monte Carlo Simulation under Two Different Scenarios

Haoyan Zhang

UNI: hz2400

Teachers College, Columbia University

Comparison for Effective of Five Classification Methods by Monte Carlo Simulation under Two Different Scenarios

In many statistic procedures, we need to consider carefully about the assumptions and scenarios under which these procedures can perform effectively. This research focuses on cluster analysis and tries to answer the question that in cluster analysis, what would happen to the effectiveness of method if the method that is assumed to analyze data with a linear decision boundary is actually nonlinear; or vice versa, what is the effect on error rate of classification if other methods that are more flexible apply to data with apparent linear decision boundaries. These are the types of questions that are difficult to work out analytically.

Thus we introduce Monte Carlo Simulation, which can generate data for different situations to search the questions mentioned above specifically. By using simulation, it becomes possible to calculate the accurate error rate for each classification in different situations and show how they performed differently directly and virtually by statistics and plots.

Specifically, this research is aimed at comparing five different classification methods, namely LR1 (Logistic Regression without interaction term), LDA (Linear Discriminant Analysis), QDA (Quadratic Discriminant Analysis), KNN1 (1-Nearest Neighbor) and KNN10 (10-Nearest Neighbor), by their corresponding classification error rate (i.e. the error rate is defined as the sum of False Negative Rate and False Positive Rate) under two different scenarios: data containing two groups with linear Bayes' decision boundary or with apparent non-linear Bayes' decision boundary. Those two categories data are generated from two different distributions with 100 duplicates using R and package: mvtnorm, MASS and class. Then we get 100 independent simulation procedures and corresponding results. By plotting error rate by a

boxplot containing five different methods, the performance under two different scenarios for five methods are easily shown and compared.

Methods of Classification

Different classification methods are all based on modeling Pr (Y = k|X = x), the probability that Y is belonging to the kth category given X values; and make classification by comparing the probabilities for each category.

Logistic Regression (LR1)

The Logistic Regression studied in the research includes only main effects and has the following function form.

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

Here $p(x) = \Pr(Y = 1|X)$, which gives us the probability of Y belonging to category 1 given specific X values. Since the probability can not be out of the range 0-1 (included), we use logistic function,

$$p(x) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2}}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2}}$$

The quantity $p(x)/[1 - p(x)]$ is called the odds having range of 0 to infinite. Traditionally odds can be used to show the chance of winning (identify Y to category 1). Then calculate the log of odds, it is linear in X.

Linear Discriminant Analysis (LDA) & Quadratic Discriminant Analysis (QDA)

Instead using logistic function, LDA uses Bayes' theorem.

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Here $f_k(X) \equiv Pr(X = x|Y = k)$ is the density function of X for an observation that comes out from the kth category of Y. π_k represents the overall probability that a randomly chosen observation comes from the kth category of response variable Y.

LDA: Assuming that $\mathbf{X} = (\boldsymbol{X_1}, \boldsymbol{X_2})$ is dawn from a multivariate normal distribution $N(\boldsymbol{\mu_k}, \boldsymbol{\Sigma})$ with a class-specific mean vector and a common covariance matrix. The distribution function of X can be written as

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

Plugging the distribution function into the Bayes' theorem, it shows that the Bayes classifier assigns an observation to the class for which the probability equation below is largest.

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}{\mu_k}^T\Sigma^{-1}\mu_k + log\pi_k$$

QDA: Assuming that $\mathbf{X} = (\boldsymbol{X_1}, \boldsymbol{X_2})$ is dawn from a multivariate normal distribution $N(\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$ with a class-specific mean vector and covariance matrix. Likewise, plug the distribution function of X into Bayes' classifier, the classification decision function is as below.

$$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + log\pi_k$$

$$= -\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_k - \frac{1}{2}{\mu_k}^T\Sigma^{-1}\mu_k + log\pi_k$$

K-Nearest Neighbor (KNN)

K-Nearest Neighbor uses the nearest n points around the test data to estimate the response variable of test data. Unlike the other methods, KNN do not need to train a model. It needs to calculate the distance of predictors X between training data and test data to find the k nearest points.

1-NN: Find the nearest neighbor to estimate the class of response variable. The class of the nearest neighbor of test data point will be assigned to that test point.

10-NN: Find 10 nearest neighbors to estimate the class of response variable. The class which has larger proportion in the 10 nearest neighbor points will be assigned to the test data.

Design of Simulation Procedure

To compare the performance of five clustering methods (LR1, LDA, QDA, KNN-1 and KNN-10) described above, we created two data generation processes (call them scenario1 and scenario2). In the first scenario the Bayes' boundary is linear, and in the second scenario it is non-linear. In this part, we use R to run the simulation processes and package mvtnorm, MASS and class are used in the research.

Here, we set 100 replicates for training set (40 sample 20 for each category for scenario1 and 100 samples 50 for each category for scenario2); and use them to get 100 models for each of five methods. Apply fitted models to make a prediction for the test set (including 1000 sample points for each category) to get the error rate for each method. Finally, we build a matrix to stored the error rates for each classification methods.

In each of two scenarios, there were two predictors (p=2). The scenarios were as follow:

Two Data Generation Procedures (DGPs)

Scenario1: Data simulated with apparent linear boundary. There were 20 training observations in each of two classes. The data (X_1 and X_2) of two different classes (shown in green and blue below) is generated from a multivariate normal distribution with the same covariance matrix $\Sigma = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$ and different mean vectors $\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and $\mu_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Run the procedure to generate 5000 data for each category and plot it. The red line in the plot is the Bayes' boundary.

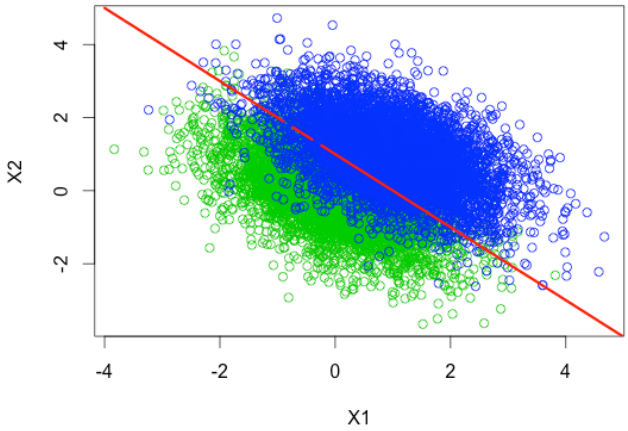

Figure 1. Scatterplot for data from the scenario1.

Scenario2: Data simulated with non-linear boundary. In this scenario, the data (X_1 *and* X_2) from two different classes is generated from a combination of variables from multivariate normal distribution with the same covariance matrix and mean vectors as scenario1. The plot for this scenario is shown below. Using Bayes' classifier, the boundary for this scenario was non-linear.

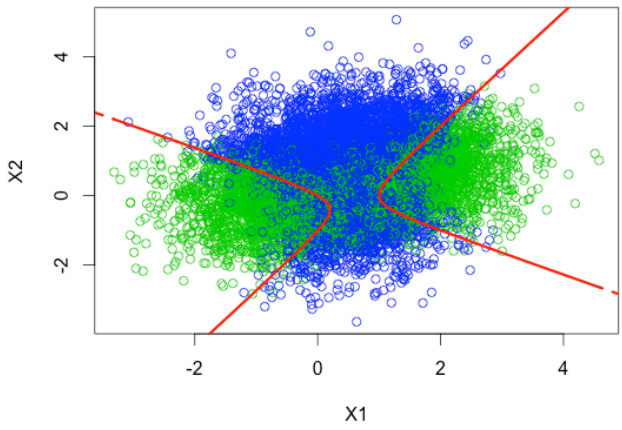

Figure 2. Scatterplot for data from the scenario2.

Calculation function for Error rates

For input, we need a dataset to be our training data to fit a model for further prediction except KNN. Also, we need to input the test data set, which is from the same distribution with training data, to evaluate error rates for each method.

The error rate used to compare the classification efficiency is the sum of False Negative Rate (the response variable assigned to 0 with actual category of 1) and False Positive Rate (the response variable assigned to 1 with actual category of 0).

Outcomes

There are the boxplots of the error rates for LR1, LDA, QDA, KNN-1 and KNN-10 under two different population distributions of data. It shows that, for scenario1 (shown in Figure.3), LR1 and LDA performed better (with an average error rate of 17%) in terms of their boundaries (straight lines). In contrast, more flexible methods such as QDA, KNN-1 and KNN-10 did not perform well.

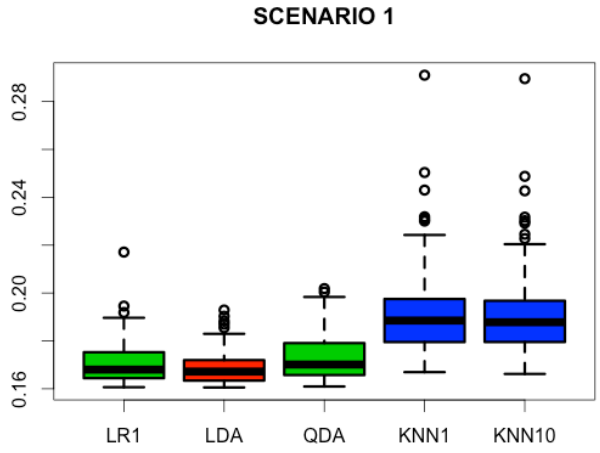

Figure 3. Boxplot for error rates of models under scenario1.

For scenario2 (see figure.4), LR1 and QDA performed very bad and had almost the same error rate for minimal (34%), maximal (46%) and average (37.5%). QDA performed best with the mean of 26%, which then followed by KNN-1 and KNN-10 with same distribution but a little higher position in the plot.

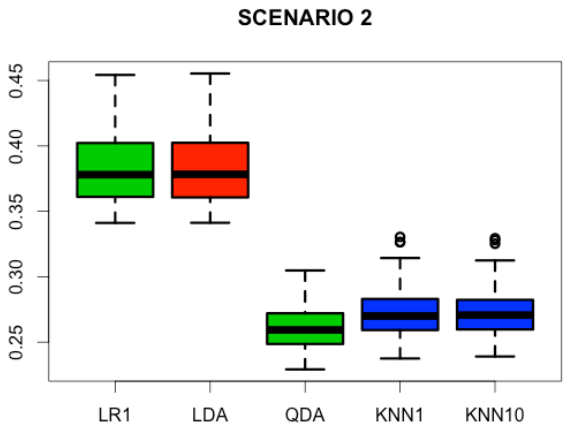

Figure 4. Boxplot for error rates of models under scenario2.

Comparing two scenarios, the average error rate ranges from 16% to 21% between the most efficient method and the less efficient method for scenario1; while, the average range for scenario2 obviously higher than scenario1 with boundary from 26% to 17.5% and even reach 45% for maximum of all methods which almost guess randomly. Besides, the difference of performance among five methods in scenario1 is not as much as that of scenario2.

Results

According to the outcome we get above, we get that those five methods can not dominant others in both two situations. When the true decision boundary is linear like in scenario1, LR1 and LDA perform better than other three more flexible methods. And when the true decision boundaries are moderately non-linear like in scenario2, QDA may give us a smaller error rate and KNN performed well too. From the outcomes, the performance of KNN-1 and KNN-10 are almost the same. KNN-1 spreads more than KNN-10.

Discussion

Logistic Regression and LDA tend to have the same performance since they both assume linear decision boundaries. LDA further assumes the distribution of the data is Gaussian with the same covariance matrix in each group. In scenario1, the data with linear Bayes' boundary did

generated from multivariate Gaussian distribution with the same covariance matrix. Thus we can see that actually LDA performed better than LR1 in terms of spread and average.

KNN is a non-parametric method and it has no assumptions about the distribution of data. KNN is also the most flexible one among the five. KNN-1 is even more flexible than KNN-10 since every single point in training data will matters significantly. Thus we can notice that in scenario1, KNN-1 spread more than KNN-10. When some assumptions are met, like in scenario1, they would perform badly, since it focused more on reduce the bias for test set, the variance raises dramatically at the same time.

The flexibility of QDA is between LDA and Logistic Regression and KNN. It makes some assumptions about the distribution of the data, and it also have a quadratic form which could capture more flexible pattern. Considering the two scenarios, QDA is the one that performed good in both situations.

Conclusion

The different assumptions of the method really matter for the effective of the classification. Considering the five classification methods studied in this research, when we can learn some knowledge about the distribution of the data, we should choose a method according to this knowledge, namely a linear boundary method Logistic Regression and LDA. KNN is good to reduce the training error rate while sometimes too fitted to training data, which is not good for further prediction. QDA may be a good choice with both flexibility and moderate balance between bias and variance trade-off.

References

Alan Genz, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl, Torsten Hothorn (2016). mvtnorm: Multivariate Normal and t Distributions. R package version 1.0-5. URL http://CRAN.R-project.org/package=mvtnorm

Alan Genz, Frank Bretz (2009), Computation of Multivariate Normal and t Probabilities. Lecture Notes in Statistics, Vol. 195., Springer-Verlag, Heidelberg. ISBN 978-3-642-01688-2

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: With applications in R.

R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

Appendix

```
library(mvtnorm)


### Function for data generation from scenario 1
dg2 <- function(ss) {
  sig <- matrix(c(1, -.5, -.5, 1), 2, 2)
  tr <- rbind(rmvnorm(ss, mean = c(0,0), sigma = sig),
              rmvnorm(ss, mean = c(1,1), sigma = sig))
  tr <- cbind(rep(c(0,1), each = ss), tr)
  tr <- data.frame(tr)
  names(tr) <- c("class", "X1", "X2")
  tr
}
d2 <- dg2(5000)
plot(d2[,2:3], col = d2$class + 3, pch = 1)


pDens1 <- function(xy, clss = "green")
{
  ### This function takes an argument xy and applies the density for
  ### class green or class blue


  if(clss == "green") {mm <- c(0,0); ss <- matrix(c(1, -.5, -.5, 1), 2, 2)}
  if(clss == "blue") {mm <- c(1,1); ss <- matrix(c(1, -.5, -.5, 1), 2, 2)}
```

```
  out <- dmvnorm(x = xy, mean = mm, sigma = ss)
  out
}

dom <- seq(-5, 5, .1)
grid <- expand.grid(dom, dom)
dfgrid <- data.frame(grid)
names(dfgrid) <- c("X1", "X2")

### Create a vector to store results
outputG <- numeric(dim(dfgrid)[1])
outputB <- numeric(dim(dfgrid)[1])

### Run the function for each x,y pair in the domain "dom"
for (i in 1:dim(dfgrid)[1])
{
  outputG[i] <- pDens1(xy = dfgrid[i,], clss = "green")
  outputB[i] <- pDens1(xy = dfgrid[i,], clss = "blue")
}

### Posterior probabilities
posts <- outputG*(1/2) / (outputG*(1/2) + outputB*(1/2))
```

```
### Bayes decision boundary for dg4.
pmat <- matrix(posts, length(dom))
contour(dom, dom, pmat, levels=0.5, labels="", xlab="", ylab="",
      axes=FALSE, add = TRUE, lwd = 3, col = "red")


### Function for data generation from scenario 2
dg5 <- function(ss) {
 sig <- diag(2)
 tr <- rbind(rmvnorm(ss, mean = c(0,0), sigma = sig),
           rmvnorm(ss, mean = c(1,1), sigma = sig))
 tr <- data.frame(tr)
 names(tr) <- c( "X1", "X2")
 attach(tr)
 probs <- exp(X1 + X2 - X1^2 + X2^2 - X1*X2) /
  (1 + exp(X1 + X2 - X1^2 + X2^2 - X1*X2))
 inds <- rbinom(n = length(probs), size = 1, prob = probs)
 detach(tr)
 tr <- cbind(inds, tr)
 tr <- data.frame(tr)
 names(tr) <- c("class", "X1", "X2")
 tr
}
d5 <- dg5(5000)
```

```
plot(d5[,2:3], col = d5$class + 3, pch = 1)


### Bayes decision boundary for dg5.
dom <- seq(-4, 6, .1)
grid <- expand.grid(dom, dom)
dfgrid <- data.frame(grid)
names(dfgrid) <- c("X1", "X2")
attach(dfgrid)
pgrid <- exp(X1 + X2 - X1^2 + X2^2 - X1*X2) /
  (1 + exp(X1 + X2 - X1^2 + X2^2 - X1*X2))
detach(dfgrid)
pmat <- matrix(pgrid, length(dom))
contour(dom, dom, pmat, levels=0.5, labels="", xlab="", ylab="",
        axes=FALSE, add = TRUE, lwd = 3, col = "red")


### Function to get overall error from a confusion table
err <- function(v1, v2) {
  tab <- table(v1, v2)
  err <- (tab/sum(tab))[1,2] + (tab/sum(tab))[2,1]
  err
}


### Function to get the error rates from 5 models
```

```
simFun <- function(dat, test) {
  ### The argument dat is a data frame generated by one of the
  ### "dg" functions defined above.

  ### Logistic regression with only linear predictors
  lr1 <- glm(class ~ X1 + X2, data = dat, family = "binomial")
  lr1P <- predict(lr1, newdata = test, type = "response") # probabilities
  lr1C <- as.numeric(lr1P >= .5)                          # 0s and 1s
  ### LDA
  lda <- lda(class ~ X1 + X2, data = dat)
  ldaC <- predict(lda, newdata = test)$class
  ldaC <- as.numeric(as.character(ldaC))
  ### QDA
  qda <- qda(class ~ X1 + X2, data = dat)
  qdaC <- predict(qda, newdata = test)$class
  qdaC <- as.numeric(as.character(qdaC))
  ### KNN-1
  knn1 <- knn(train = dat[,2:3], cl = dat[,1], test = test[,2:3],
              k = 10, prob = TRUE)
  knn1 <- as.numeric(as.character(knn1))
  ### KNN-10
  knn10 <- knn(train = dat[,2:3], cl = dat[,1], test = test[,2:3],
               k = 10, prob = TRUE)
```

```
  knn10 <- as.numeric(as.character(knn10))
  errs <- c(err(lr1C, test[,1]),
            err(ldaC, test[,1]), err(qdaC, test[,1]),
            err(knn1, test[,1]), err(knn10, test[,1]) )
  errs
}



### Load packages
library(MASS) # for LDA and QDA
library(class) # for KNN

################################################################################
#######
### SCENARIO 2
#####################################################################
################################################################################
#######

### Set seed for reproducibility
set.seed(294850)
```

```
### Generate testing set
test2 <- dg2(10000)

### How many replications?
R <- 100
### Create a placeholder to store output
out2 <- matrix(0, R, 5)

### Generate 100 training data sets. For each data set determine
### the error rate on the test set using each method
for (i in 1:R) {
  out2[i,] <- simFun(dat = dg2(20), test = test2)
}

out2 <- data.frame(out2)
names(out2) <- c("LR1", "LDA", "QDA", "KNN1", "KNN10")
boxplot(out2, col = c(3, 2, 3, 4, 4), lwd = 3, main = "SCENARIO 1")

##############################################################################
#######
### SCENARIO 5
#####################################################################
```

```
        ##########################################################################
#######

        ### Set seed for reproducibility
        set.seed(222222)

        ### Generate testing set
        test5 <- dg5(10000)

        ### How many replications?
        R <- 100
        ### Create a placeholder to store output
        out5 <- matrix(0, R, 5)

        ### Generate 100 training data sets. For each data set determine
        ### the error rate on the test set using each method
        for (i in 1:R) {
          out5[i,] <- simFun(dat = dg5(50), test = test5)
        }

        out5 <- data.frame(out5)
        names(out5) <- c("LR1", "LDA", "QDA", "KNN1", "KNN10")
boxplot(out5, col = c( 3, 2, 3, 4, 4), lwd = 3, main = "SCENARIO 2")
```