STAT37400-Final Project

Linear Discriminant Analysis using Bayesian Theorm with parametric and nonparametric inference

Suchan Park (12173246)

Linear Discriminant Analysis(LDA) is a traditional statistical method for classification. There is two way to make the classification model, one is using Fisher's linear discriminant and the other is using a bayesian rule. For both methods, the main assumption of the model is that data follow a multivariate normal distribution. Therefore, the probability density function is required for the model.

When data does not follow a multivariate normal distribution, the main assumption is not satisfied and the model will get worse. By using nonparametric inference, LDA can be made without the assumption. Instead of using multivariate normal distribution, we can estimate the distribution from the data nonparametrically.

In the main body, there is only a relevant theories and a result of an experiment. Every Rcode is given in an Appendix.
Keywords: LDA, Bayes Theorm, Nonparametric Inference

Contents

1. Description of the data **3**
1.1 Preprocess 3
1.2 Relevant function 3

2. Parametric analysis **3**
2.1 Assumption 3
2.2 Estimate 4
2.3 Classification Model 4
2.4 Uncertainty 4
2.4.1 Uncertainty for estimated parameter mu 5
2.4.2 Uncertainty for estimated parameter S0 5
2.4.3 Uncertainty for the model 6
2.5 Result 7

3. Nonparametric analysis **8**
3.1 Assumption 8
3.2 Density Estimate 8
3.2.1 Kernel density estimate 8
3.2.2 Optimal bandwidth using Silverman's rule of thumb 8
3.3 Classification 9
3.4 Uncertainty 9
3.4.1 Uncertainty for the estimated kernel density 9
3.4.2 Uncertainty for the model 10
3.5 Result 12

4. Discussion **12**

5. Reference **13**

6. Appendix **14**

1. Description of the data

For the project, I will use iris dataset in r which is a popular dataset for classification problem. Data has 4 dimensional of [Sepal.Length , Sepal.Width, Petal.Length, Petal.Width] and categorical variable [Species]. Before I start, I want to preprocess the data and make some functions which I will use for the analysis.

1.1 Preprocess

I will separate the target variable(species) to the numeric input variable. After that, we need to classify the index of each species to train a model. Since we only have 150 data, it is hard to make a sufficient train set and data set. I will use 130 randomly chosen data as a train set, the others as a test set.

1.2 Relevant function

Before I start the classification, I make a required function for the model. The first function is to compute the risk of the model and the second one is a function to subtract a vector to a matrix in each row.

2. Parametric analysis

My goal is to make classification model using **Linear Discriminant Analysis with Bayesian rule**. At first, I will make the model in the sense of parametric statistics.

Let's consider Linear Discriminate Analysis with a bayesian rule. When w_i is a class, i=1,2,,,c and x is an input data. We want to find that when x is given, which class should be assigned to the x. Thus, we need to compute a conditional probability of w_i given x for all i and find the maximum of $p(w_i|x)$. By using a bayes rule, we can find the probability.

$$p(w_i|x) \approx p(x|w_i)p(w_i)$$

The prior probability of W can be easily computed by the ratio of the class. Thus, remaining issue is how to compute $P(x|w_i)$.

2.1 Assumption

The main assumption of **Linear Discriminant Analysis** is that data are from multivariate normal(MVN) distribution and each class has the same covariance matrix $\sum$. Each data from the same class will be independent. By estimating parameters of MVN, $\hat{\mu}$ and $\hat{\sum}$, we can compute $P(x|w_i)$. Thus, we need to estimate the parameters first. I will use maximum likelihood estimates of μ and $\sum$.

$$\hat{\mu}_{MLE} = \bar{x}, \qquad \hat{\sum}_{MLE} = S_0$$

where

$$S_0 = \frac{1}{n}(X - \bar{X})(X - \bar{X})^t = \frac{S}{n}$$

Note that S is called Scatter matrix and S will be used later.

2.2 Estimate

```
## [1] "Estimated mu of class1"
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##    4.9951220    3.4048780    1.4536585    0.2390244
```

```
## [1] "Estimated mu of class2"
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.929787     2.763830     4.253191     1.321277
```

```
## [1] "Estimated mu of class3"
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     6.507143     2.957143     5.454762     2.002381
```

```
## [1] "Estimated covariance matrix"
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length       0.6054     -0.0405       1.1337      0.4598
## Sepal.Width       -0.0405      0.1750      -0.3122     -0.1165
## Petal.Length       1.1337     -0.3122       2.8498      1.1914
## Petal.Width        0.4598     -0.1165       1.1914      0.5414
```

2.3 Classification Model

Let's set $\sigma_i(x) = p(w_i|x)$. From the above process, we can compute $\sigma_i(x)$ for i=1,2,3. For input x, the class is $\arg\max_i \sigma_i(x)$.

As an example, I want to run the model for a single input.

```
## [1] "True class is 2"
```

```
##        [,1]  [,2]   [,3]
## [1,] 0.0072 0.041 0.0299
```

We can find that the model works well. Since I don't consider constant in the model, the sum of probability does not add to 1.
The result of classification is the following:

```
##   [1] 1 3 1 1 2 1 1 2 3 2 3 1 1 3 1 1 1 1 1 2 2 2 3 3 2 1 1 2 1 2 2 1 3 2 1
##  [36] 1 3 2 3 2 3 2 3 3 1 2 1 2 2 2 3 3 2 2 3 2 2 3 1 2 1 2 2 3 2 3 2 2 2 2
##  [71] 1 2 2 3 3 3 2 2 3 3 3 1 1 1 1 1 3 1 2 1 2 1 2 3 2 3 2 2 2 2 1 2 3 2 1
## [106] 2 3 1 2 1 1 3 2 2 3 3 1 1 3 1 2 2 2 3 3 2 3 3 2 1
```

2.4 Uncertainty

Uncertainty can be measured by a variance or risk. Since I estimate parameters, there are an Uncertainty for each parameter, $\hat{\mu}_i$, and $\hat{\sum}$. In addition, there is an Uncertainty for the model, risk.

2.4.1 Uncertainty for estimated parameter mu

We can find a variance of μ_i by using bootstrap. With the variance, we can get a confidence interval for the parameters.

```
## [1] "CI of mu1"
```

```
##               Lower   Mean  Upper
## Sepal.Length 4.7766 4.9951 5.2137
## Sepal.Width  3.2903 3.4049 3.5195
## Petal.Length 0.9803 1.4537 1.9270
## Petal.Width  0.0339 0.2390 0.4442
```

```
## [1] "CI of mu2"
```

```
##               Lower   Mean  Upper
## Sepal.Length 5.7187 5.9298 6.1408
## Sepal.Width  2.6476 2.7638 2.8800
## Petal.Length 3.7986 4.2532 4.7078
## Petal.Width  1.1230 1.3213 1.5195
```

```
## [1] "CI of mu3"
```

```
##               Lower   Mean  Upper
## Sepal.Length 6.3049 6.5071 6.7093
## Sepal.Width  2.8400 2.9571 3.0742
## Petal.Length 5.0056 5.4548 5.9039
## Petal.Width  1.8079 2.0024 2.1969
```

2.4.2 Uncertainty for estimated parameter S0

We can get a variance of MLE estimate of the covariance matrix in multivariate normal distribution by using **Wishart distribution**. Wishart distribution is the sampling distribution of the maximum-likelihood estimator of the covariance matrix of a multivariate normal distribution. Therefore, we can use the variance of $\sum$ and use bootstrap to compute the confidence interval.

$$S \sim W_p(\sum, n-1)$$

where S is scatter matrix which I mentioned above.
Thus,

$$Var(S_0) = \frac{Var(S)}{n^2} = \frac{n-1}{n^2}\left[s_{ij}^2 + s_{ii}s_{jj}\right] \qquad i,j = 1,2,3,4$$

Estimated covariance matrix

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length       0.6054     -0.0405       1.1337      0.4598
## Sepal.Width       -0.0405      0.1750      -0.3122     -0.1165
## Petal.Length       1.1337     -0.3122       2.8498      1.1914
## Petal.Width        0.4598     -0.1165       1.1914      0.5414
```

Upper bound of the estimated covariance matrix

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length       0.6237     -0.0353       1.1709      0.4754
## Sepal.Width       -0.0353      0.1802      -0.2975     -0.1105
## Petal.Length       1.1709     -0.2975       2.9364      1.2283
## Petal.Width        0.4754     -0.1105       1.2283      0.5577
```

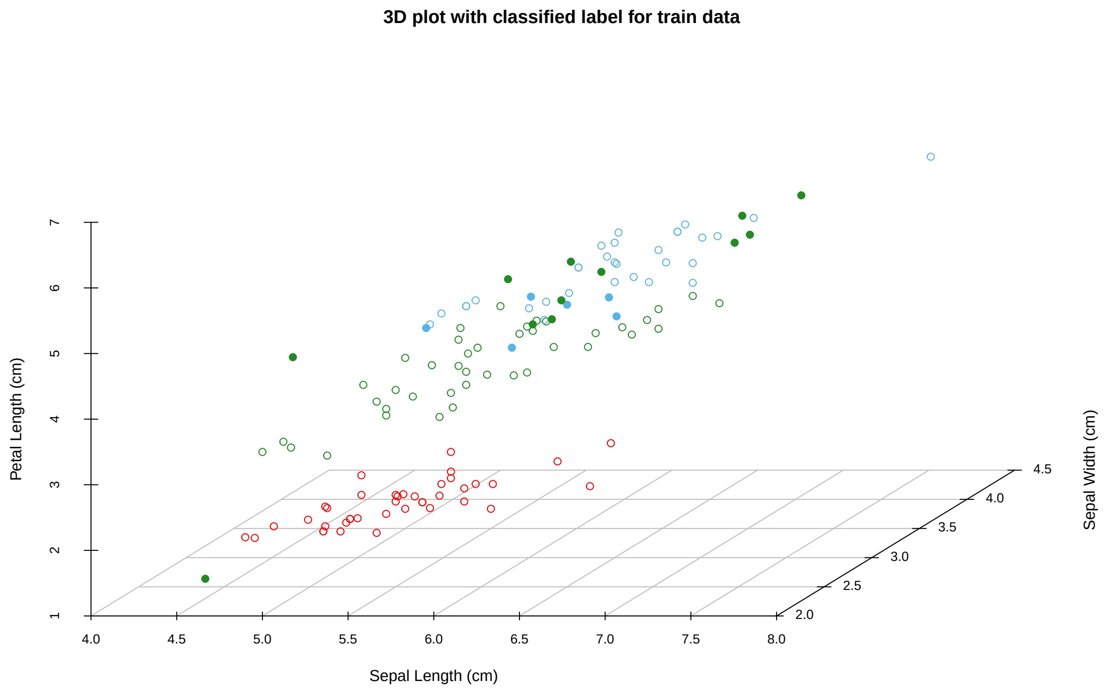
Lower bound of the estimated covariance matrix

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length       0.5871     -0.0457       1.0965      0.4442
## Sepal.Width       -0.0457      0.1698      -0.3269     -0.1225
## Petal.Length       1.0965     -0.3269       2.7632      1.1545
## Petal.Width        0.4442     -0.1225       1.1545      0.5251
```

2.4.3 Uncertainty for the model

We can also measure an uncertainty of the model by computing the risk of classification result. Let's check the performance of the model with the true label.

```
## [1] "The error rate of the model for train is about 13.8462 %"
```

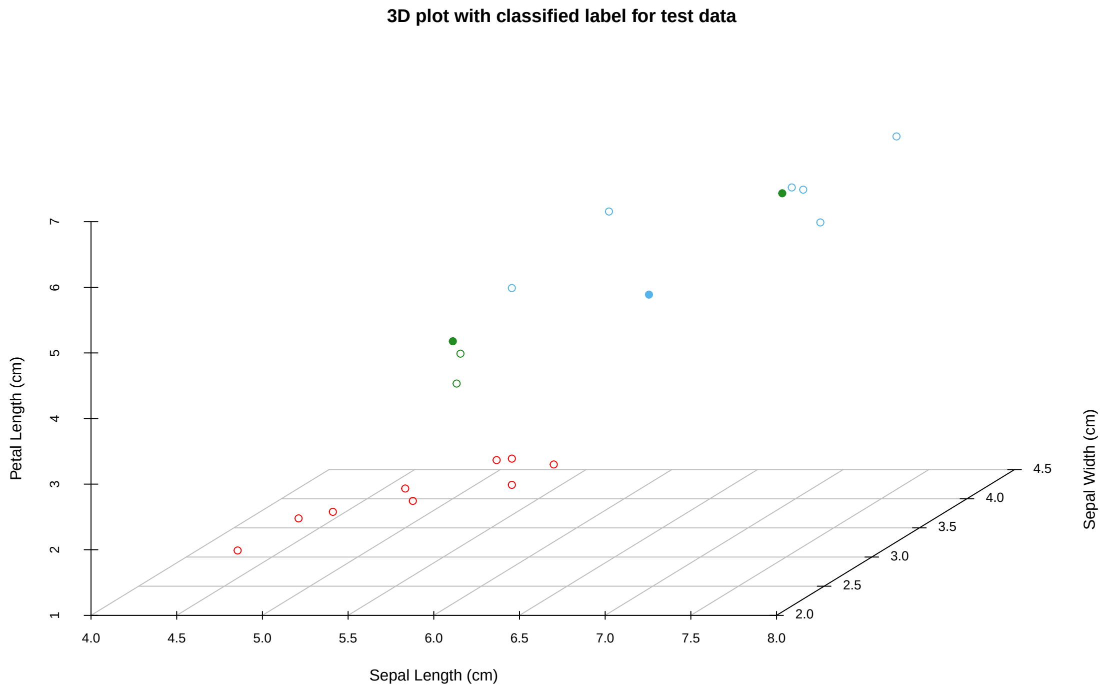


Above plot shows the result of classification. Since the dimension of the data is 4, I only show the plot with 3 dimension. Color means species for each data. The filled circle means the misclassified case.
Above risk is a train error so we need to check a test error.

```
## [1] "The result of classification for test data"
```

```
##  [1] 1 1 1 1 1 1 1 1 1 3 2 2 3 3 3 3 2 2 3 3
```

```
## [1] "The error rate of the model for test is about 15 %"
```



We can find that test error is higher than train error but the errors are similar. If we can use more data, the result will be convincing.
In conclusion, we can find that classification pretty works well but there is some misclassification around the border of two species. We need to check the cause of the misclassification and update the model.

2.5 Result

From the model, we get about 13.8% of training error and 15% of test error. When we consider that the number of data is not very large, the error rate is relative higher than we expected. One of the reason of the poor performance will be related with the assumption we made. The main assumption of the classification model is that data follows a multivariate normal distribution. We can check that data really follows the distribution using **Henze-Zirkler's MVN test**.

```
##            Test       HZ      p value MVN
## 1 Henze-Zirkler 2.108086 1.554312e-15  NO
```

From the test, We can find that the given data does not follow a multivariate normality. Since our main assumption is wrong, we need to modify the model. We can use the same model without the multivariate normal assumption by estimating kernel density.

3. Nonparametric analysis

From the above process, we can find that there is a room for an improvement in the model. We can update the classification model using Nonparametric inference.

3.1 Assumption

The main structure of the classification model is the same with the above. The Only difference is that we will use an estimated kernel density instead of multivariate normal density. To compute the multivariate density estimate, we need to decide an appropriate kernel and find optimal bandwidth. We will use **Gaussian Kernel** and compute the bandwidth by using **Silverman's rule of thumb**.

3.2 Density Estimate

3.2.1 Kernel density estimate

$$\hat{f}_H(x) = \frac{1}{n}\sum K_H(x - x_i)$$

where

$$K_H(x) = |H|^{-\frac{1}{2}}K(H^{-\frac{1}{2}}x)$$

and x is 4-dimensional vector and H is 4x4 matrix.

3.2.2 Optimal bandwidth using Silverman's rule of thumb

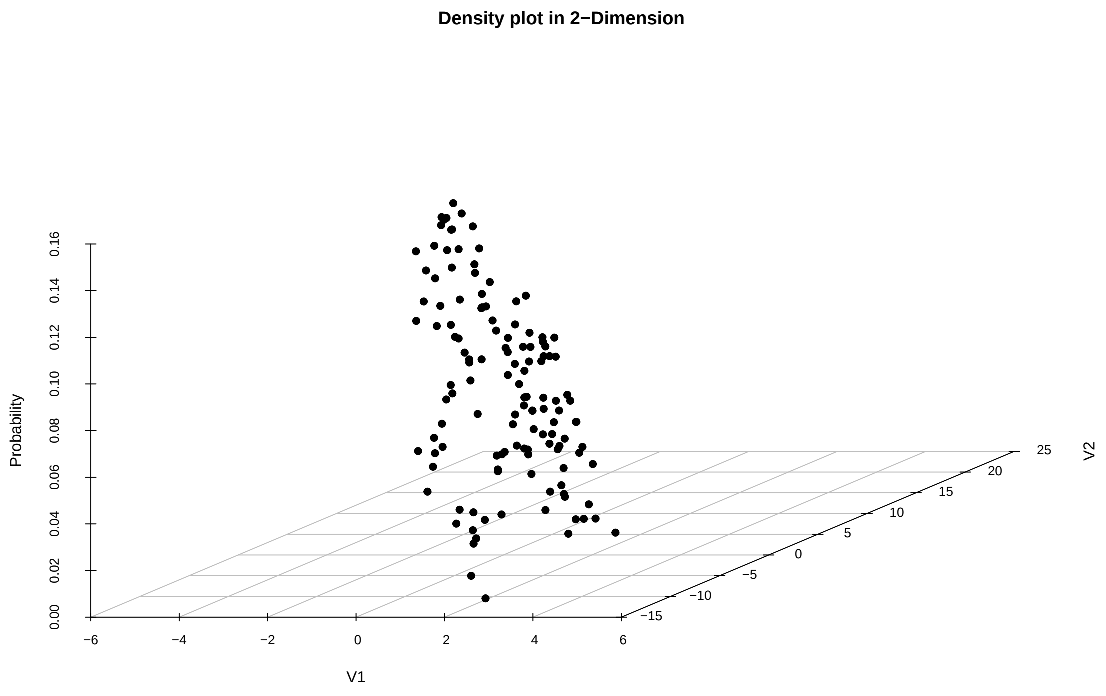
$$\sqrt{H}_{ii} = (\frac{4}{d+2})^{\frac{1}{d+4}} n^{\frac{-1}{d+4}} \sigma_i, \quad 0 \quad if \quad i \neq j$$

```
est.density <- rep(0,N)
est.sd <- rep(0,N)
for(i in 1:N)
  est.density[i]<-Kernel4D(train[i,],train)
```

We can compute the estimated density using the Kernel4D function. Since the dimension of data is 4, it is impossible to show a plot of the density directly. Thus, We will use t-SNE method to reduce the dimension of input data x.

```
##             V1         V2
## [1,] -5.064502  16.989547
## [2,]  2.803492 -13.179164
## [3,] -4.353928  18.486632
## [4,] -5.671935  18.568404
## [5,]  2.070266  -3.655195
## [6,] -5.218886  19.535116
```

We can see that 4-dimensions input data is reduced to 2-dimension. With the modified input, we can plot the estimated density.



3.3 Classification

Similar to the above process, we can get $\sigma_i(x)$ for i=1,2,3 and the class of a input is $\arg\max_i \sigma_i(x)$.

The result of classification is following:

```
##   [1] 1 3 1 1 2 1 1 2 3 3 3 1 1 3 1 1 1 1 1 2 2 2 3 3 3 1 1 2 1 2 3 1 3 2 1
##  [36] 1 3 3 3 2 3 2 2 3 1 2 1 2 2 2 3 2 2 2 3 2 2 3 1 2 1 3 2 3 2 2 2 3 2 2
##  [71] 1 2 2 3 3 3 2 2 3 3 3 1 1 1 1 1 3 1 2 1 2 1 3 3 2 3 2 1 2 2 1 2 3 2 1
## [106] 2 2 1 2 1 1 3 2 3 2 3 1 1 3 1 2 2 2 3 3 2 3 2 2 1
```

3.4 Uncertainty

From the nonparametric model, uncertainty comes from the estimated kernel and the model.

3.4.1 Uncertainty for the estimated kernel density

A sample analog of the exact finite-sample variance of $f_n(x)$ is provided by following.

$$s_n^2 = \frac{1}{nh^2}\sum_{i=1}^{n} K(\frac{(x-X_i)}{h_h})^2 - \frac{\hat{f}_n(x)^2}{n}$$

Hence, a studentized form of $z_n(x)$ for asymptotic confidence interval is defined by

$$t_n(x) = \frac{\hat{f}_n(x) - E\hat{f}_n(x)}{s_n(x)}$$

We can estimate $t_n(x)$ using bootstrap.

$$t_n^* = \frac{\hat{f}_n(x) - \hat{f}_n(x)}{s_n^*(x)}$$

where $\hat{f}_n(x)$ is a bootstrap estimator of f and s_n^* is the bootstrap estimator of s_n. Then, the upper limit is given by

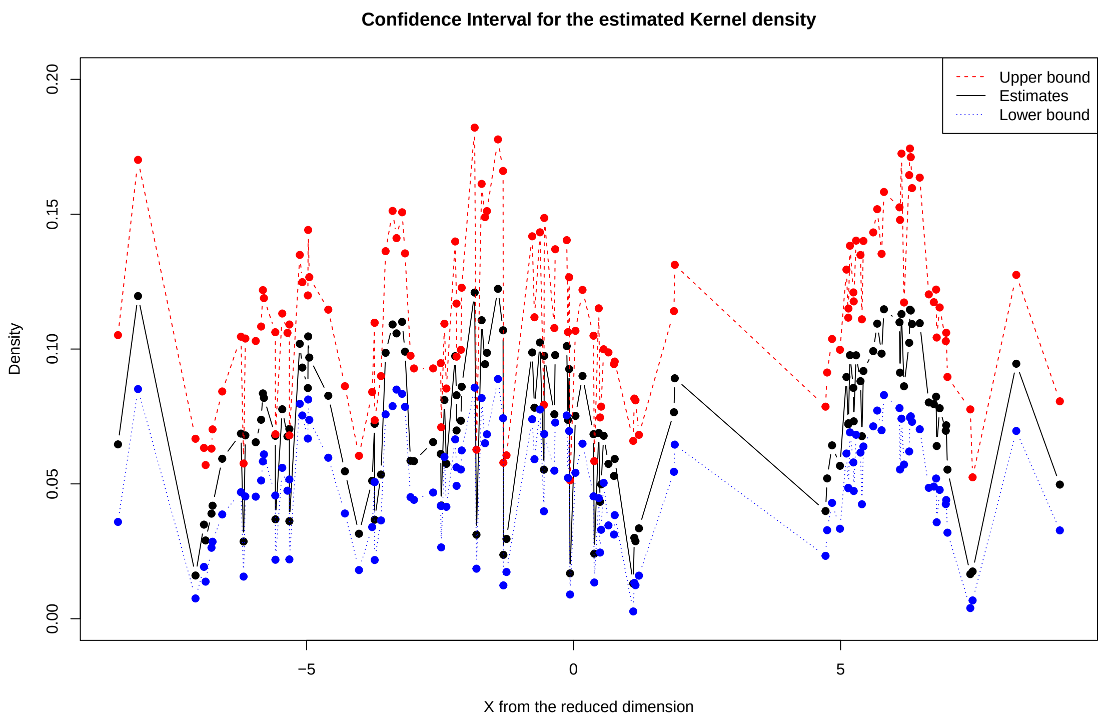
$$E\hat{f}_n(x)^U = \hat{f}_n(x) - s_n(x)u_{\alpha/2}^*$$

and the lower limit is given by

$$E\hat{f}_n(x)^L = \hat{f}_n(x) - s_n(x)u_{1-(\alpha/2)}^*$$

where, $u_{\alpha/2}^*$ is the bootstrap estimate of the quantile defined by $P(t_n^* \leq u_{\alpha/2}^*) = \alpha/2$ and $u_{1-(\alpha/2)}^*$ is the bootstrap estimate of the quantile defined by $P(t_n^* \leq u_{1-(\alpha/2)}^*) = 1 - (\alpha/2)$. (Hall [1992b, 679] and Davidson and MacKinnon [2003, chapter 5]).

Following is the C.I for the estimated kernel density. To visualize the result, we will use dimension-reduction from 4 to 1.

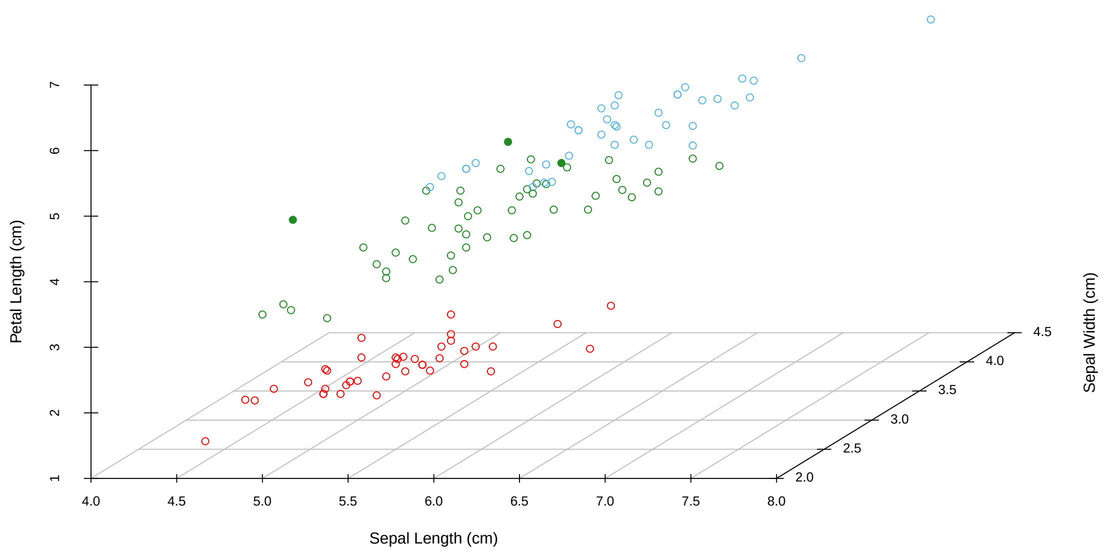


3.4.2 Uncertainty for the model

Based on the choice of kernel and bandwidth, the classifier model and the risk of the model will be changed. Thus, We can check the risk of the model with the true label.

```
## [1] "The error rate of the model for train is about 2.3077 %"
```

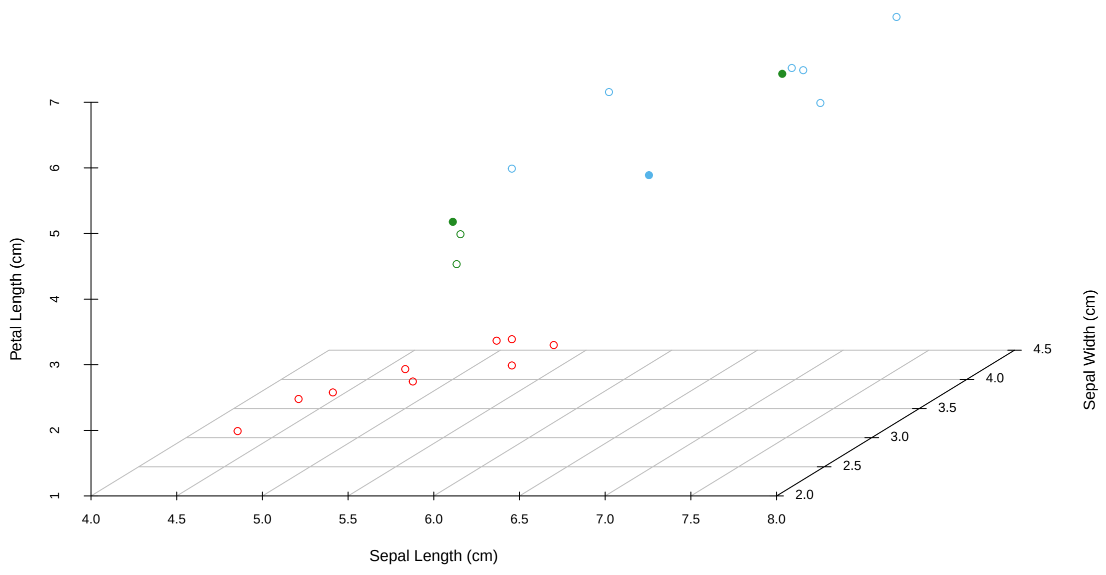
3D plot with classified label for train data



We also need to check the new model for test dataset.

```
## [1] "The error rate of the model for test is about 10 %"
```

3D plot with classified label for test data



3.5 Result

From the updated model, we get about 2.3% of training error and 10% of test error which shows that the case of misclassification decreases remarkably. It means that the update of the model is the right decision.

Since we don't know the distribution of the data, it is a strength to use a nonparametric method to estimate kernel density. However, the nonparametric model has several problems. It depends on the choice of hyperparameters such as Kernel or bandwidth so the mischoice of hyperparameters can get a worse result. In addition, the nonparametric model requires much computation than the parametric model and need more data as the dimension of data increases.

4. Discussion

From the experiments, we can find that nonparametric model shows the better performance than those of parametric model. As we discussed above, the reason might be the assumption of the model. The parametric model requires assumption for a given data. We show that a parametric model does not work well when the assumption is not fit to the given data. Since we cannot figure out the true distribution for a model, the nonparametric method is appropriate for this case.

It does not mean that the a nonparametric method is better than the a parametric method. If the size of data is very large, the data usually follows normal distribution by the central limit theorem. In addition, a nonparametric method usually requires a large computing cost, it can be inefficient. We also consider the embedded bias of a nonparametric model.

Thus we need to know both parametric and nonparametric methods and should use the proper model for a situation.

5. Reference

Amir Atiya, Gaussian Processes for Classification, Cairo University, 2010

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

C. V. Fiorio, Confidence intervals for kernel estimation, The Stata Journa, 2004

Hall. p, The Bootstrap and Edgeworth Expansion, Annals of Statistics 20, 1992

Multivarate Kernel Density Estimate. (n.d.). wikipedia. Retrieved Nov 23, 2018 from https://en.wikipedia.org/wiki/Multivariate_kernel_density_estimation#Rule_of_thumb/

Nathaniel E. Helwig, Multivariate Linear Regression, University of Minnesota, 2017

Wasserman L, All of nonparametric statistics. New York: Springer, 2010

Wischart Distribution. (n.d.). wikipedia. Retrieved Nov 23, 2018 from https://en.wikipedia.org/wiki/Wishart_distribution#Estimator_of_the_multivariate_normal_distribution/

6. Appendix

R code is attached for the relevent section

1.1 Preprocess

```r
data <-as.matrix(iris[,c(1:4)])
data.class <- as.numeric(iris$Species)

train.idx <- sample(1:150,130,replace=FALSE)
train <- data[train.idx,]
test <- data[-train.idx,]
train.class <- data.class[train.idx]
test.class <- data.class[-train.idx]

data <- train
data.class <- train.class

idx1 <- which(data.class==1)
idx2 <- which(data.class==2)
idx3 <- which(data.class==3)
d <- dim(data)[2]      # Dimension of input x
N <- dim(data)[1]      # The number of data
```

1.2 Relevant function

```r
# Risk function
Acc.rate <- function(y,yh)
  return(round(sum(y!=yh)/length(y)*100,4))
```

```r
# Matrix subtraction function
Mat.subtract <- function(M,x)
{
  N <- dim(M)[1]
  temp <- M
  for(i in 1:N)
    temp[i,]<-M[i,]-x
  return(temp)
}
```

2.2 Estimate

```r
mu0 <- colMeans(data)
mu1 <- colMeans(data[idx1,])   # Estimated mu of class1
mu2 <- colMeans(data[idx2,])   # Estimated mu of class2
mu3 <- colMeans(data[idx3,])   # Estimated mu of class3

xx <- Mat.subtract(data,mu0)
S0 <- round((t(xx)%*%xx)/N,4)  # Estimated covariance matrix
```

2.3 Classification Model

```
# Prior Probability
W1 <- length(idx1)/N
W2 <- length(idx2)/N
W3 <- length(idx3)/N
```

```
# Model to classify the data.
LDA_bayes_param <- function(x)
{
  sigma1 <- dmvnorm(x,mu1,S0)*W1
  sigma2 <- dmvnorm(x,mu2,S0)*W2
  sigma3 <- dmvnorm(x,mu3,S0)*W3
  return(matrix(c(sigma1,sigma2,sigma3),1,3))
}
```

```
estimate.class1 <- rep(0,N)
for(i in 1:N)
  estimate.class1[i]  <-which.max(LDA_bayes_param(data[i,]))
estimate.class1
```

2.4 Uncertainty

2.4.1 Uncertainty for estimated parameter mu_i

```
bootstrap <- function(target,B=1000,a=0.05)
{
  boot.result <- matrix(rep(0,B*d),B,d)
  for(i in 1:B)
  {
    temp <- as.matrix(rmvnorm(50,target,S0))
    boot.result[i,] <- colMeans(temp)
  }
  boot.mean <- colMeans(boot.result)
  err <- sqrt(colMeans(Mat.subtract(boot.result,boot.mean)^2))*qnorm(1-a/2)
  return(err)
}
```

```
mu.CI <- function(target)
{
  err <- bootstrap(target)
  temp <- round(cbind(target-err,target,target+err),4)
  colnames(temp)<- c("Lower","Mean","Upper")
  return(temp)
}
```

```
print("CI of mu1")
mu.CI(mu1)
print("CI of mu2")
mu.CI(mu2)
```

```r
print("CI of mu3")
mu.CI(mu3)
```

2.4.2 Uncertainty for estimated parameter S0

```r
bootstrap.s <- function(S,B=1000,a=0.05)
{
  boot.result <- matrix(rep(0,B*d*d),B,d*d)
  for(k in 1:B)
  {
    temp <- (rWishart(1,N-1,S)/N)[,,1]
    temp.err <- S
    for(i in 1:d)
      for(j in 1:d)
        temp.err[i,j]<-sqrt((N-1)/N^2*(temp[i,j]^2+temp[i,i]*temp[j,j]))
    boot.result[k,] <- matrix(temp.err,1,d*d)
  }
  boot.mean <- colMeans(boot.result)
  err <- sqrt(colMeans(Mat.subtract(boot.result,boot.mean)^2))*qnorm(1-a/2)
  err <- matrix(err,d,d)
  return(err)
}
```

```r
S0.err <- bootstrap.s(S0)
S0.upper <- round(S0+S0.err,4)
S0.lower <- round(S0-S0.err,4)
```

```r
# Estimated covariance matrix
print(S0)
```

```r
# Upper bound of the estimated covariance matrix
S0.upper
```

```r
# Lower bound of the estimated covariance matrix
S0.lower
```

2.4.3 Uncertainty for the model

```r
# Compute the risk of model
risk1<-Acc.rate(data.class,estimate.class1)
print(paste("The error rate of the model is about",risk1,"%"))
```

```r
colors <- c("#999999", "#E69F00", "#56B4E9")
pchs <- c(19,1)
colors <- colors[estimate.class1]
pchs <- pchs[as.numeric(data.class==estimate.class1)+1]
scatterplot3d(data[,1:3],color = colors, pch=pchs, box=FALSE,
              main="3D plot",
              xlab = "Sepal Length (cm)",
              ylab = "Sepal Width (cm)",
              zlab = "Petal Length (cm)")
```

2.5 Discussion

```
MVN.result <- mvn(data,mvnTest="hz")
MVN.result$multivariateNormality
```

3.2 Density Estimate

3.2.1 Kernel density estimate

```
# Create H using normal reference rule.
CreateHii <- function(dat)
{
  sig <- sd(dat)
  n <- length(dat)
  Hii <- (4/(d+2))^(1/(d+4))*n^(-1/(d+4))*sig
  return(Hii^2)
}
H11 <- CreateHii(data[,1])
H22 <- CreateHii(data[,2])
H33 <- CreateHii(data[,3])
H44 <- CreateHii(data[,4])
H <- matrix(rep(0,16),4,4)
diag(H)<-c(H11,H22,H33,H44)
```

```
# Create K_H(x)
KernelH <- function(x,H)
{
  const <- (2*pi)^(-d/2)*det(H)^(-1/2)
  ex <- -1/2*(x%*%solve(H)%*%x)
  return(const*exp(ex))
}
```

```
# 4-D Kernel Density Estimate
Kernel4D <- function(x, dataset,sd=FALSE)
{
  if(sd==FALSE)
  {
    temp.store <- 0
    n <- length(dataset[,1])
    for(j in 1:n)
    {
      temp.x <- dataset[j,]
      temp <- KernelH(x-temp.x,H)
      temp.store <- temp.store + temp
    }
    return(temp.store/n)
```

```r
  }
  else   # compute the sd of the kernel.
  {
    temp.store <- 0
    n <- length(dataset[,1])
    for(j in 1:n)
    {
      temp.x <- dataset[j,]
      temp <- KernelH(x-temp.x,H)
      temp.store <- temp.store + temp^2
    }
    return(temp.store/n)
  }
}
```

3.3 Classification

```r
LDA_bayes_nonparam <- function(x)
{
  sigma1 <- Kernel4D(x,data[idx1,])*W1
  sigma2 <- Kernel4D(x,data[idx2,])*W2
  sigma3 <- Kernel4D(x,data[idx3,])*W3
  return(matrix(c(sigma1,sigma2,sigma3),1,3))
}
```

```r
estimate.class2 <- rep(0,N)
for(i in 1:N)
  estimate.class2[i]  <-which.max(LDA_bayes_nonparam(data[i,]))
estimate.class2
```

3.4 Uncertainty

3.4.1 Uncertainty for the estimated kernel density

```r
# SD for kernel density estimation
est.sd2 <- rep(0,N)
for(i in 1:N)
  est.sd2[i]<-Kernel4D(train[i,],train,sd=TRUE)
```

```r
# bootstrap
B <- 100
d <- 4
h <- min(c(H11,H22,H33,H44))
R <- rep(0,B)
za <- qnorm(0.975)
boot.result.sd <- matrix(rep(0,B*N),B,N)
boot.result.tn <- matrix(rep(0,B*N),B,N)
for(i in 1:B)
{
  boot.train <- as.matrix(sample_n(as.data.frame(train),N,replace=TRUE))
  boot.density <- as.matrix(rep(0,N),N,1)
  boot.sd2 <- as.matrix(rep(0,N),N,1)
  boot.tn <- as.matrix(rep(0,N),N,1)
  for(j in 1:N)
  {
    boot.density[j]<-Kernel4D(train[j,],boot.train) # Bootstrap estimator of f
    boot.sd2[j] <- Kernel4D(train[j,],boot.train,sd=TRUE)-boot.density[j]^2/N
    boot.tn[j] <- (boot.density[j]-est.density[j])/sqrt(boot.sd2[j])
  }
  boot.result.sd[i,]<-sqrt(boot.sd2)
  boot.result.tn[i,]<-boot.tn
}
```

```
# Compute upper and lower bound
fxu <- rep(0,N)
fxl <- rep(0,N)
for(i in 1:N)
{
  temp <- boot.result.tn[,i]
  u <- quantile(temp,probs=0.025)
  l <- quantile(temp,probs=0.975)
  fxu[i] <- est.density[i]-sqrt(est.sd2[i])*u
  fxl[i] <- est.density[i]-sqrt(est.sd2[i])*l
}
```

```
# Dimension Reduction to 1
tsne1 <- Rtsne(as.matrix(train), check_duplicates=FALSE, pca=TRUE, perplexity=30, theta=0
d_tsne1 <- as.matrix(tsne1$Y)
colnames(d_tsne1) <- c("V1")
```

```
# Reorder the data
density.order <- order(d_tsne1)
boot.result <- cbind(d_tsne1[density.order], fxl[density.order], est.density[density.orde
```

```
# Plot the result
plot(boot.result[,1],boot.result[,3],pch=19,type="b",ylim=c(0,0.2), xlab="X from the redu
points(boot.result[,1],boot.result[,4],pch=19,lty=2, type="b", col="red")
points(boot.result[,1],boot.result[,2],pch=19,lty=3,type="b", col="blue")
legend("topright",legend=c("Upper bound","Estimates","Lower bound"),col=c("red","black","
```

3.4.2 Uncertainty for the model

```
# Compute the risk of model, train
risk2<-Acc.rate(data.class,estimate.class2)
print(paste("The error rate of the model for train is about",risk2,"%"))
```

```
colors <- c("#ff0000", "#228B22", "#56B4E9")
pchs <- c(19,1)
colors1 <- colors[estimate.class2]
```

```r
pchs1 <- pchs[as.numeric(data.class==estimate.class2)+1]
scatterplot3d(data[,1:3],color = colors1, pch=pchs1, box=FALSE,
              main="3D plot with classified label for train data",
              xlab = "Sepal Length (cm)",
              ylab = "Sepal Width (cm)",
              zlab = "Petal Length (cm)")
```

```r
estimate.class22 <- rep(0,length(test.class))
for(i in 1:length(test.class))
  estimate.class22[i]  <-which.max(LDA_bayes_nonparam(test[i,]))
```

```r
# Compute the risk of model, test
risk22<-Acc.rate(test.class,estimate.class22)
print(paste("The error rate of the model for test is about",risk22,"%"))
```

```r
colors <- c("#ff0000", "#228B22", "#56B4E9")
pchs <- c(19,1)
colors <- colors[estimate.class11]
pchs <- pchs[as.numeric(test.class==estimate.class11)+1]
scatterplot3d(test[,1:3],color = colors, pch=pchs, box=FALSE,
              main="3D plot with classified label for test data",
              xlab = "Sepal Length (cm)",
              ylab = "Sepal Width (cm)",
              zlab = "Petal Length (cm)")
```