

Comparison of PCA, Linear Regression and Logistic Regression method in terms of accuracy and error rate for breast cancer dataset

**Department of Electrical and Computer Engineering
ECE 503**

PROJECT REPORT

Submitted by

**Nikita Kapoor
(V00949256)**



**University
of Victoria**

Date: December 22, 2019

Abstract

In this project, I will be analyzing the efficiency of Principal Component Analysis, Linear Regression and Logistic Regression Method and comparing it in terms of error rate and number of misclassifications. I will be considering WDBC (Wisconsin Diagnostic Breast Cancer) dataset. The dataset has 30 features selected from each of 569 patients of which 357 patients were diagnosed as benign and 212 patients were diagnosed as malignant.

The objective of applying these methods is for automatic diagnosis of breast cancer and comparing the accuracy with the help of testing data. The accuracy is measured when a new unused sample is classified as benign or malignant.

Out of these three methods, Logistic Regression was applied in the lab experiment – 4 on Wisconsin Diagnostic Breast Cancer dataset. Now, I will apply, Principal Component Analysis and Linear Regression. The MATLAB code and the results obtained are also included in this report.

1. Dataset

The dataset has 30 features selected from each of 569 patients of which 357 patients were diagnosed as benign and 212 patients were diagnosed as malignant. The features in the dataset represent different characteristics having a variation in the parameters. To overcome this problem, the dataset is first normalized to a particular scale for all features.

We are using two data files namely D_bc_tr.mat and D_bc_te.mat for training and testing purpose respectively.

D_bc_tr.mat: This dataset is of size 31 * 480 in which first 30 rows contains 30 medical features for each of 480 patients. The corresponding labels are contained in its 31st row. Label “-1” is assigned to the samples diagnosed as benign, while label “1” is assigned to the samples diagnosed as malignant.

D_bc_te.mat: This dataset is of size 31 * 89 in which first 30 rows contains 30 medical features for each of 89 patients. The corresponding labels are contained in its 31st row.

For normalizing the data, we follow these steps:

1. Compute mean and variance of all columns of each row of D_bc_tr and D_bc_te as follows:

$$m_i = \frac{1}{N} \sum_{k=1}^N z_k^{(i)}, \quad \sigma_i^2 = \frac{1}{N-1} \sum_{k=1}^N (z_k^{(i)} - m_i)^2$$

2. Subtract mean from every element of the row and divide it by the variance.

$$\hat{z}_i = \frac{1}{\sigma_i} \begin{bmatrix} z_1^{(i)} - m_i & z_2^{(i)} - m_i & \cdots & z_N^{(i)} - m_i \end{bmatrix}$$

2. Methods Used

Classification is done using:

2.1 Principal Component Analysis

PCA is a means of feature extraction. It is a technique of eliminating the least important or redundant features and keeping the important ones called principal components. In this project, we are using PCA for binary classification.

For dimension reductionality, we follow the following steps

1. Compute the mean vector of the data by

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$$

2. Compute the covariance matrix of the data points.

$$\mathbf{C}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} (\mathbf{x}_i^{(j)} - \boldsymbol{\mu}_j)(\mathbf{x}_i^{(j)} - \boldsymbol{\mu}_j)^T \text{ for } j = 0, 1, \dots, 9$$

3. Compute q eigen vectors according to their eigen values in the decreasing order.
4. Choose first q eigen vectors for q dimensional features.

$$\mathbf{f}_i = \mathbf{U}_q^T (\mathbf{x}_i - \boldsymbol{\mu})$$

The Euclidian distance between x and its representation is computed using:

$$e_j = \|\mathbf{x} - \hat{\mathbf{x}}_j\|_2$$

The main advantage of PCA is that we can compress the data by reducing the dimensions without much loss of information. We construct a low rank approximation where S is a diagonal matrix of q largest singular values. The singular value decomposition of matrix gives low value approximations.

$$\mathbf{C}_j \approx \mathbf{U}_q^{(j)} \mathbf{S}_q^{(j)} \mathbf{U}_q^{(j)T}$$

where

$$\mathbf{U}_q^{(j)} = [\mathbf{u}_1^{(j)} \ \mathbf{u}_2^{(j)} \ \dots \ \mathbf{u}_q^{(j)}] \quad \text{for } j = 0, 1, \dots, 9$$

2.2 Linear Regression

Linear Regression is a method of supervised learning. In Linear regression there is a linear relationship between the independent and dependent variable. We have linear equation $y = mx + c$. Here our motive is to find best values for m and c .

For prediction, we follow the following steps:

A training dataset considered with the dimensions as follows:

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n), n=1,2,\dots,N\} \text{ with } \mathbf{x}_n \in R^{d \times 1} \text{ and } \mathbf{y}_n \in R^{1 \times 1}$$

We consider a linear model to predict or classify the new data with Weight \mathbf{W} and bias \mathbf{b} . These are the parameters optimized to achieve the best prediction/classification accuracy.

$$f(\mathbf{x}, \mathbf{W}, \mathbf{b}) = \mathbf{W}^T \mathbf{x} + \mathbf{b} ; \mathbf{W} \in R^{d \times 1} \text{ and } \mathbf{b} \in R^{1 \times 1} ;$$

We minimize the loss or objective function to measure the prediction error over the whole training data set

$$\underset{\mathbf{W}, \mathbf{b}}{\text{minimize}} \sum_{n=1}^N \|f(\mathbf{x}_n, \mathbf{W}, \mathbf{b}) - \mathbf{y}_n\|_2^2$$

The parameters \mathbf{W} , \mathbf{b} and \mathbf{y} are given by:

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_l], \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_l \end{bmatrix}, \text{ and } \mathbf{y}_n = \begin{bmatrix} y_1^{(n)} \\ y_2^{(n)} \\ \vdots \\ y_l^{(n)} \end{bmatrix}$$

The loss function is expressed as

$$\sum_{n=1}^N (\mathbf{w}_i^T \mathbf{x}_n + b_i - y_i^{(n)})^2 = \|\hat{\mathbf{X}} \hat{\mathbf{w}}_i - \hat{\mathbf{y}}_i\|_2^2 \quad \text{for } i = 1, 2, \dots,$$

with

$$\hat{\mathbf{X}} = \begin{bmatrix} \hat{\mathbf{x}}_1^T \\ \hat{\mathbf{x}}_2^T \\ \vdots \\ \hat{\mathbf{x}}_N^T \end{bmatrix} \quad \text{where } \hat{\mathbf{x}}_n = \begin{bmatrix} \mathbf{x}_n \\ 1 \end{bmatrix} \text{ for } n = 1, 2, \dots, N,$$

The objective or loss function is quadratic and convex, so it is a global minimizer. Altogether, the equations are solved as:

$$\begin{bmatrix} \hat{w}_1^* & \hat{w}_2^* & \cdots & \hat{w}_l^* \end{bmatrix} = (\hat{X}^T \hat{X})^{-1} \hat{X}^T \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_l \end{bmatrix} = (\hat{X}^T \hat{X})^{-1} \hat{X}^T \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_N^T \end{bmatrix}$$

The optimal parameters are obtained.

$$W^* = \begin{bmatrix} w_1^* & w_2^* & \cdots & w_l^* \end{bmatrix} \text{ and } b^* = \begin{bmatrix} b_1^* \\ b_2^* \\ \vdots \\ b_l^* \end{bmatrix}$$

So the optimal linear model obtained is

$$f(x, W^*, b^*) = W^{*T} x + b^*$$

In simple form,

$$f(x) = W^{*T} x + b^*$$

In Linear Regression model the output $f(x)$ is a continuous function of the input which is why it is called linear model. The function $f(x)$ is called discriminant function as it is used for both classification and prediction.

When doing binary classification, the new point x which is outside the dataset D :

$$\begin{aligned} &\text{belongs to class } \mathcal{P} \text{ if } x^T w^* + b^* > 0 \\ &\text{belongs to class } \mathcal{N} \text{ if } x^T w^* + b^* < 0 \end{aligned}$$

And the decision boundary is given by:

$$f(x) = w^{*T} x + b^* = 0$$

This term will define a flat plane which divides the input space into two parts. The points on the side of the plane will have $f(x)$ values with same sign and the points on the other side of the plane will have $f(x)$ values with opposite sign. In this way, classification is done in Linear Regression Method. Similarly, this can be extended for multi-class case.

2.3 Logistic Regression

Logistic regression is a classification model which is used to predict class labels with the help of input data or features. Logistic regression is applicable to binary classification problems and is used to predict a binary outcome.

The classification is performed in the following steps.

1. First, the objective function is minimized.

$$f(\hat{\mathbf{w}}) = \frac{1}{N} \sum_{n=1}^N \ln(1 + e^{-y_n \hat{\mathbf{w}}^T \hat{\mathbf{x}}_n})$$

Logistic regression uses conditional probability with optimal parameters \mathbf{w} and b that are obtained by solving the unconstrained problem. Once \mathbf{w} is obtained, the conditional probability for a new sample outside training data is evaluated by computing the conditional probability.

$$P(y | \mathbf{x}) = \frac{1}{1 + e^{-y(\hat{\mathbf{w}}^{*T} \hat{\mathbf{x}})}}$$

2. The new sample outside the training data is classified into positive class or negative class as it is a binary classification technique.

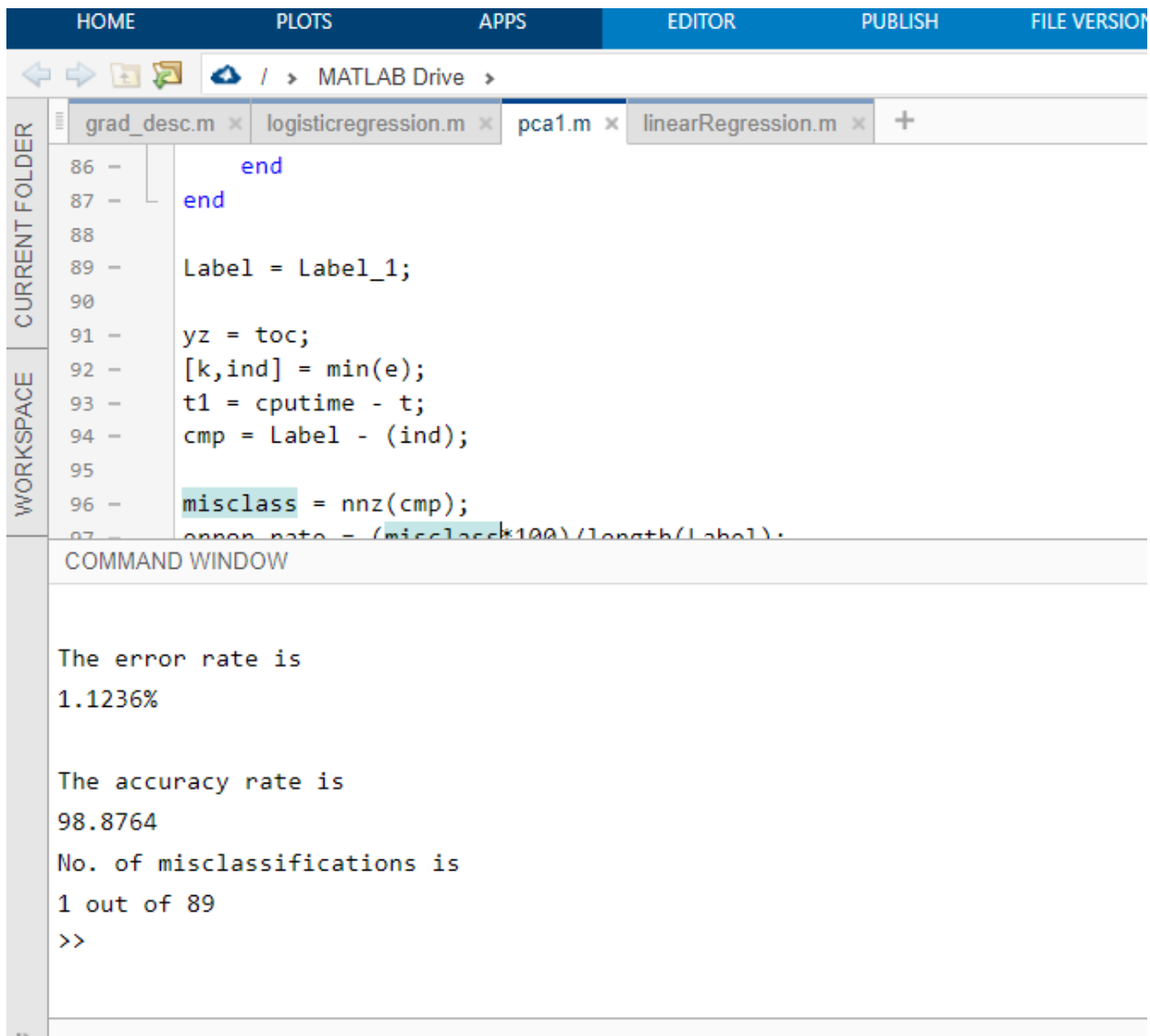
$$\begin{cases} \mathbf{x} \in \mathcal{N} & \text{if } \mathbf{w}^{*T} \mathbf{x} + b^* < 0 \\ \mathbf{x} \in \mathcal{P} & \text{if } \mathbf{w}^{*T} \mathbf{x} + b^* > 0 \end{cases}$$

We observe that the objective function is strictly convex, so it is a unique global minimizer and this minimizer is characterized when its gradient is zero. The convexity of the function also assures that the Gradient Descent algorithm is not sensitive to the choice of initial point.

3. Result

The error rate, accuracy and number of misclassifications are calculated for each experiment. The results are as follows:

Principal Component Analysis: First the dataset is converted to structured format. And then, I applied PCA on that dataset. The number of mis-classifications encountered were: 1 out of 89. The error rate and accuracy rate values are: 1.1236% and 98.8764% respectively. The result obtained is better than the Linear Regression method but it is same as compared to logistic regression for 75 iterations.



The image shows a MATLAB interface with the Editor window displaying a script named 'pca1.m'. The script contains MATLAB code for calculating misclassification and error rate. The Command Window shows the output of the script, which includes the error rate, accuracy rate, and the number of misclassifications.

```
86 - end
87 - end
88
89 - Label = Label_1;
90
91 - yz = toc;
92 - [k,ind] = min(e);
93 - t1 = cputime - t;
94 - cmp = Label - (ind);
95
96 - misclass = nnz(cmp);
97 - error_rate = (misclass*100)/length(Label);
```

COMMAND WINDOW

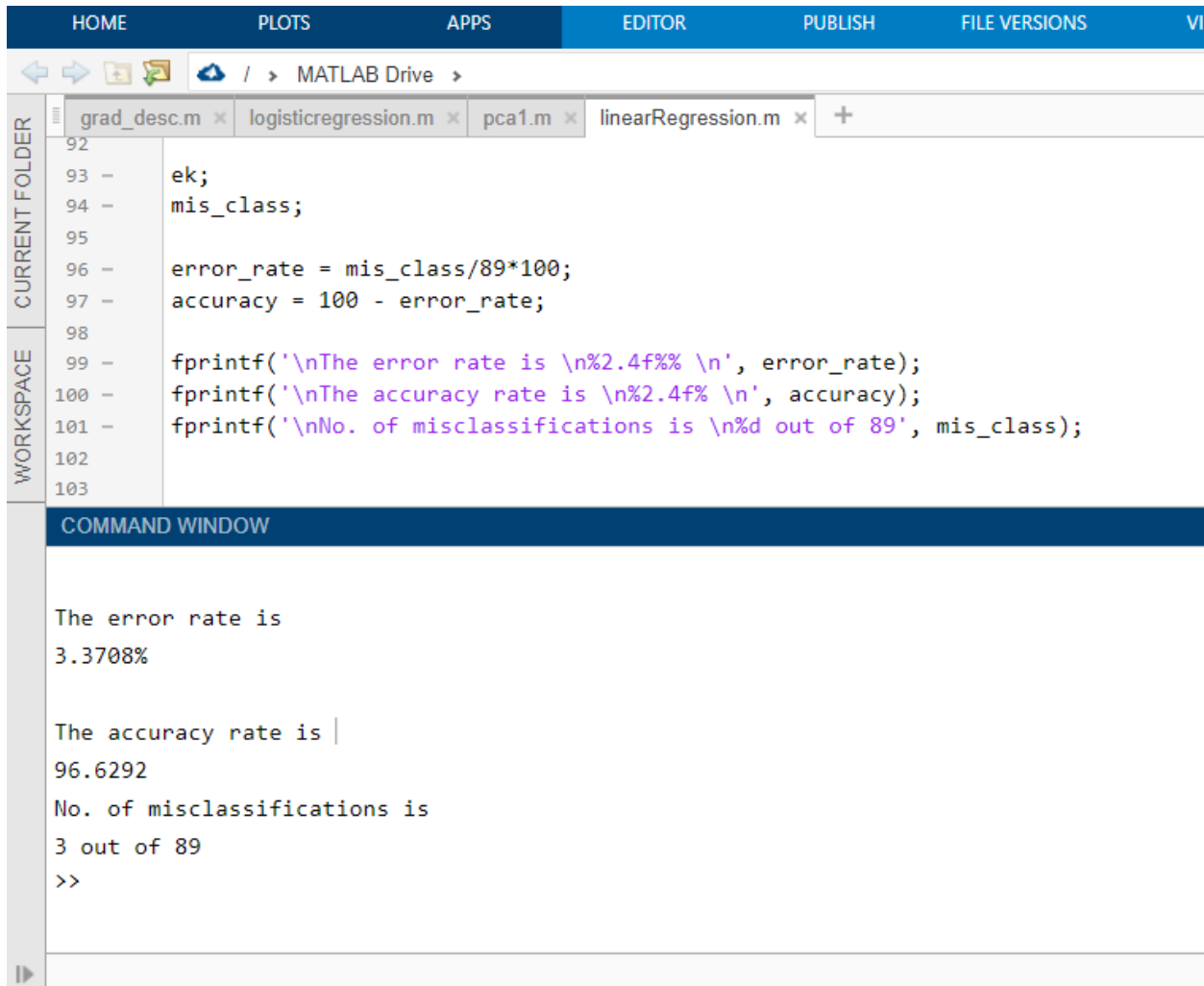
```
The error rate is
1.1236%

The accuracy rate is
98.8764

No. of misclassifications is
1 out of 89
>>
```

Fig 1: Command Window Output for PCA

Linear Regression: This Linear Regression is a simple 2 class classification problem. As the dataset contains only 2 classes. The number of misclassifications were: 3 out of 89. The error rate and accuracy rate values are: 3.3708% and 96.6292% respectively. The accuracy in this method is lowest as compared to other two methods as the number of misclassifications are more.



The image shows a MATLAB interface with the Editor window displaying a script for linear regression evaluation. The script calculates the error rate and accuracy based on the number of misclassifications (3 out of 89). The Command Window shows the output of the script, displaying the error rate as 3.3708%, the accuracy rate as 96.6292%, and the number of misclassifications as 3 out of 89.

```
92  
93 - ek;  
94 - mis_class;  
95  
96 - error_rate = mis_class/89*100;  
97 - accuracy = 100 - error_rate;  
98  
99 - fprintf('\nThe error rate is \n%2.4f%% \n', error_rate);  
100 - fprintf('\nThe accuracy rate is \n%2.4f%% \n', accuracy);  
101 - fprintf('\nNo. of misclassifications is \n%d out of 89', mis_class);  
102  
103
```

COMMAND WINDOW

```
The error rate is  
3.3708%  
  
The accuracy rate is |  
96.6292  
No. of misclassifications is  
3 out of 89  
>>
```

Fig 2: Command window output for Linear Regression

Logistic Regression: When number of iterations were 12, the count of misclassifications were: 3 out of 89, error rate and accuracy rate values were: 3.3708% and 96.6292% respectively. When number of iterations were 75, error rate and accuracy rate values were: 1.1236% and 96.6292% respectively. When number of iterations are 75, the accuracy percentage is same as of linear regression method but when iterations are 12, the accuracy is less.

For K = 12

WORKSPACE	COMMAND WINDOW
11 -	<code>Xtrain=zeros(30,480);</code>
12 -	<code>for i=1:30</code>
13 -	<code>xi=D_bc_tr(i,:);</code>
14 -	<code>mi=mean(xi);</code>
15 -	<code>vi=sqrt(var(xi));</code>
16 -	<code>Xtrain(i,:)=(xi-mi)/vi;</code>
17 -	<code>end</code>
	 K = 12 The error rate is 3.3708% The accuracy rate is 96.6292 No. of misclassifications is 3 out of 89 ..

Fig 3: Command window output for Logistic Regression

For K = 75

WORKSPACE	
12 -	for i=1:30
13 -	xi=D_bc_tr(i,:);
14 -	mi=mean(xi);
15 -	vi=sqrt(var(xi));
16 -	Xtrain(i,:)=(xi-mi)/vi;
17 -	end
18	

COMMAND WINDOW	
K = 75	
The error rate is 1.1236%	
The accuracy rate is	
98.8764	
No. of misclassifications is	
1 out of 89	
>>	

Fig 4: Command window output for Logistic Regression

4. Conclusion

In this project, we worked on Principal Component Analysis, Linear Regression and Logistic Regression method and calculated their error rates and accuracy. The comparison has been made in the table below.

The tabular form comparing the output results of all the methods is as follows:

S.No	Method Used	Misclassifications	Error Rate	Accuracy(%)
1.	Principal Component Analysis	1/89	1.1236	98.8764
2.	Linear Regression	3/89	3.3708	96.6292
3.	Logistic Regression	1/89	1.1236	98.8764

In all the 3 datasets, PCA and Logistic Regression are performing better than Linear Regression in terms of accuracy and less error rates. Therefore, we can say that PCA and Logistic Regression are better techniques than Linear Regression.

5. MATLAB Code

Code for all three algorithms used are given below.

Principal Component Analysis:

```
clc
clear all
close all

load('D_bc_tr.mat');
load('D_bc_te.mat');

hk = [];hj = [];

for i = 1 : 480
if D_bc_tr(31,i) == 1
hk1 = D_bc_tr(:,i);
hk = [hk hk1];
end
if D_bc_tr(31,i) == -1
hj2 = D_bc_tr(:,i);
hj = [hj hj2];
end
end

Xtr = [hk(1:30,:) hj(1:30,)];

varr = 48;
q = 8;

meann = [];U = [];var = [];
meann1 = [];U1 = [];var1 = [];
meann2 = [];U2 = [];var2 = [];

for j = 0:1
if j == 0

varr = 180;
X = Xtr(:,1:180);
meann = mean(X,2);
A = X - meann;
```

```

C = 1/ varr *(A*A');
[uq, Sq] = eigs(C,q);
meann1 = [meann1 meann];
U1 = [U1 uq];
var1 = [var1 X];
end

if j == 1
varr = 300;
X = Xtr(:,181:end);
meann = mean(X,2);
A = X - meann;
C = 1/ varr *(A*A');
meann2 = [meann2 meann];

[uq, Sq] = eigs(C,q);
U2 = [U2 uq];
var2 = [var2 X];
end
end

U = [U1 U2];
var = [var1 var2];
meann = [meann1 meann2];
D_test = D_bc_te(1:30,:);
t = cputime;
tic;
for j = 1:89
for i = 0:1
Uq = U(:,(i*q)+1:(i+1)*q);
f_j = Uq'*(D_test(:,j)-meann(:,i+1));
x_capp = Uq*f_j+meann(:,i+1);
e(i+1,j) = norm(D_test(:,j)-x_capp);
end
end

Lab = D_bc_te(31,:);
Lab_1 = zeros(1,89);

for i = 1 : 89
if Lab(:,i) == -1
Lab_1(1,i) = 2;
end
if Lab(:,i) == 1
Lab_1(1,i) = 1;
end
end

```

```
end
```

```
Lab = Label_1;
```

```
yz = toc;  
[k,ind] = min(e);  
t1 = cputime - t;  
cmp = Lab - (ind);
```

```
misclass = nnz(cmp);  
error_rate = (misclass*100)/length(Label);  
accuracy = 100 - error_rate;
```

```
fprintf('\n\nThe error rate is \n%2.4f%% \n', error_rate);  
fprintf('\n\nThe accuracy rate is \n%2.4f%% \n', accuracy);  
fprintf('\n\nNo. of misclassifications is \n%d out of 89', misclass);
```

Linear Regression:

```
clc
clear all
close all

load('D_bc_tr.mat');
load('D_bc_te.mat');

hk = [];
hj = [];

for i = 1 : 480
    if D_bc_tr(31,i) == -1
        be = D_bc_tr(:,i);
        hk = [hk be];
    end

    if D_bc_tr(31,i) == 1
        ma = D_bc_tr(:,i);
        hj = [hj ma];
    end
end

Xtr1 = hk(1:30,:);
Xtr2 = hj(1:30,:);
y1 = [ones(300,1);-ones(180,1)];
y2 = [ones(180,1);-ones(300,1)];

Xtr = [Xtr1 Xtr2];
x_11 = [Xtr1 Xtr2];
x_11(31,:) = [ones(1,300) ones(1,180)];

X11 = x_11';
w1star = pinv(X11)*y1;
w1 = w1star(1:30,:);
b1 = w1star(31,:);

x_22 = [Xtr2 Xtr1];
x_22(31,:) = [ones(1,180) ones(1,300)];
```



```

X22 = x_22';
w2star = pinv(X22)*y2;
w2 = w2star(1:30,:);
b2 = w2star(31,:);

W_star = [w1 w2];
b_star = [b1;b2];

Label = D_bc_te(31,:);

orig1 = [];orig2 = [];
XTe1 = [];XTe2 = [];

for i = 1 : 89

if Label(:,i) == -1
    orig2 = -1;
    orig2 = [orig2 orig2];
Xte = D_bc_te(1:30,i);
    XTe1 = [XTe1 Xte];
end

if Label(:,i) == 1
    orig1 = 1;
    orig1 = [orig1 orig1];
Xte = D_bc_te(1:30,i);
    XTe2 = [XTe2 Xte];
end

end

L = [orig1 orig2];
XTe = [XTe1 XTe2];

Xte = [XTe];
Label = [ones(1,57) ones(1,32)+1];

ek = zeros(2,89);
mis_class = 0;

```

```

for col = 1:89
    f = W_star'*Xte(:,col)+b_star;
    [maxvalue index] = max(f);
    ek(index,col) = 1;

    if index ~= Label(col)
        mis_class = mis_class +1;
    end
end

ek;
mis_class;

error_rate = mis_class/89*100;
accuracy = 100 - error_rate;

fprintf('\n\nThe error rate is \n%2.4f%% \n', error_rate);
fprintf('\n\nThe accuracy rate is \n%2.4f% \n', accuracy);
fprintf('\n\nNo. of misclassifications is \n%d out of 89', mis_class);

```

Logistic Regression:

```
clear;
clc;
close all;

load D_bc_tr;
load D_bc_te;

NumFeature=30;
K=12; % maximum efficiency at 75

Xtrain=zeros(30,480);
for i=1:30
    xi = D_bc_tr(i,:);
    mi = mean(xi);
    vi = sqrt(var(xi));
    Xtrain(i,:) = (xi-mi)/vi;
end

Xtest = zeros(30,89);
for I = 1:30
    xi = D_bc_te(i,:);
    mi = mean(xi);
    vi = sqrt(var(xi));
    Xtest(i,:) = (xi-mi)/vi;
end

ytrain = D_bc_tr(31,:);
ytest = D_bc_te(31,:);

Dtrain = [Xtrain;ytrain]
Dtest = [Xtest;ytest]

Eps = 10^-9;
W = zeros(NumFeature+1,1);

F = zeros(1,K);
K = 1;
Gk = g_wdbc(w,Dtrain);
```

```

Dk = -gk;
alpha_k = bt_lsearch2019(w,dk, 'f_wdbc', 'g_wdbc',Dtrain);

while((norm(alpha_k*dk)>=eps) && (k<K))
w = w+alpha_k*dk;
gk = g_wdbc(w,Dtrain);
dk = -gk;
alpha_k=bt_lsearch2019(w,dk, 'f_wdbc', 'g_wdbc',Dtrain);
f(k)=f_wdbc(w,Dtrain);
k=k+1;
end

Dt=[Xtest;ones(1,89)];
Result=w'*Dt;
TestLabel=zeros(1,length(Result));
FalsePos=0;
FalseNeg=0;
for ii=1:length(Result)
if Result(ii)<0
TestLabel(ii)=-1;
if ytest(ii)>0
FalseNeg=FalseNeg+1;
end
else
TestLabel(ii)=1;
if ytest(ii)<0
FalsePos=FalsePos+1;
end
end
end

mis_class = (FalsePos+ FalseNeg);
error_rate = mis_class/89*100;
accuracy = 100 - error_rate;

fprintf('\nK = %d', K);
fprintf('\nThe error rate is %2.4f%% \n', error_rate);
fprintf('\nThe accuracy rate is %2.4f%% \n', accuracy);
fprintf('\nNo. of misclassifications is %d out of 89', mis_class);

```

References

- [1] 'ECE-403/ 503-Lab Manual', by Dr. Wu-Sheng Lu, 2019.
- [2] 'ECE-403/ 503-Course Notes', by Dr. Wu-Sheng Lu, 2019.
- [3] 'Breast Cancer Detection: An Application of Machine Learning Algorithms', Abien et al, 2019.