PROBLEM-2

January 16, 2020

In this problem we have worked with the grayscale image of P.C. Mahalanobis. Then we have performed image compression using Vector Quantization by the help of the clustering algorithm K-means which is also known as "Llyod's algorithm" in computer science and engineering.

Before proceeding let us check the K-means algorithm and image vector quantization.

K-MEANS ALGORITHM

K-means is one of the most popular partitioning methods of clustering. It performs a non-hierarchical clustering and groups the observations of a dataset into a given number of clusters with the view to minimize the within-cluster scatter/variation due to the clustering process.

Suppose we have n observations $x_1, x_2, ..., x_n \epsilon \mathbb{R}^p$ and a measure $d_{ij} = d(x_i, x_j)$ of dissimilarity between x_i and x_j, for every $i, j = 1(1)n$. We may wish to group the n observations into K clusters. A clustering of the above n observations can be thought of as a function C which assigns the cluster number for an observation, i.e, for an observation x_i, $C(x_i) = k \epsilon \{1, 2, ..., K\}$ denotes the cluster k to which x_i is assigned during the clustering process. We may, instead, simply denote by $C(i)$, the cluster to which x_i is assigned.

Now, the within cluster scatter due to the above clustering is given by

$$W = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k,\, j:C(j)=k} d_{ij}$$

where n_k is the number of observations in thek^{th} cluster, $k = 1(1)K$.

In particular, we may take $d_{ij} = \|x_i - x_j\|^2$, the squared Euclidean Distance between the observations x_i and x_j. Then, we have

$$W = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k\,,j:C(j)=k} \|x_i - x_j\|^2$$

$$= \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k} \|x_i - \bar{x_k}\|^2$$

where

$$\bar{x_k} = \frac{1}{n_k} \sum_{i:C(i)=k} x_i$$

is the mean of the observations in the k^{th} cluster.

This W is the within-cluster variation due to the above clustering. Equivalently, we may choose to minimize

$$W = \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i:C(i)=k} \|x_i - c_k\|^2$$

with respect to the clustering method C and $c_1, c_2, ..., c_K$.

K-means attempts to find the clustering method C so as to approximately minimize this within cluster variation. It runs in the following way:

- We start with an initial guess for $c_1, c_2, ..., c_K$ (e.g., pick K points at random over the range of $x_1, ..., x_K$), then:

1. Minimize over C: for each $i = 1(1)n$, find the cluster center c_k closest to x_i , and let $C(i) = k$
2. Minimize over $c_1, ..., c_K$: for each $k = 1(1)K$, let $c_k = \bar{x_k}$.

We repeat this process and stop when W does not change any more.

Thus, one gets hold of a clustering method C to cluster the above n observations.

VECTOR QUANTIZATION

The K-means clustering algorithm represents a key tool in the apparently unrelated area of image and signal compression, particularly in vector quantization or VQ. There are three stages to image vector quantization:

- (Prototype Creation) Generally an image of mn pixels is represented in m rows and n columns. But here first we create arrays by grouping together q local pixels by row (that is q adjacent pixels row-wise). Then we have a collection of vectors V = {Vi, i = 1,2, 3,...., mn/q} where each Vi is a q dimensional vector. This V is called index and the elements Vi are called index entry. Some of these vectors,Vi belonging to V , will be similar in some respects. For example, many vectors may be extracted from a uniform area of an image. This leads to the following idea.

– Cluster vectors in V into r groups where(r << mn/q). The smaller the choice of r, the greater is the compression.

– In other words, the set of vectors, V is partitioned into r distinct sets, S1,S2,....,Sr corresponding to each cluster.

– Each subset Si is then been represented by a suitable representative/prototype vector (may be the cluster mean/mediod). The set of prototype vectors constitutes the codebook. Each of the member in the codebook is called codeword .

– Thus codebook contains r vectors, each of which has an address,(1,2,... r). Because the number of selected elements is much smaller than the number of vectors in the image, the number of bits required to represent the address of a prototype vector is much smaller than the number of bits required to represent an image vector.

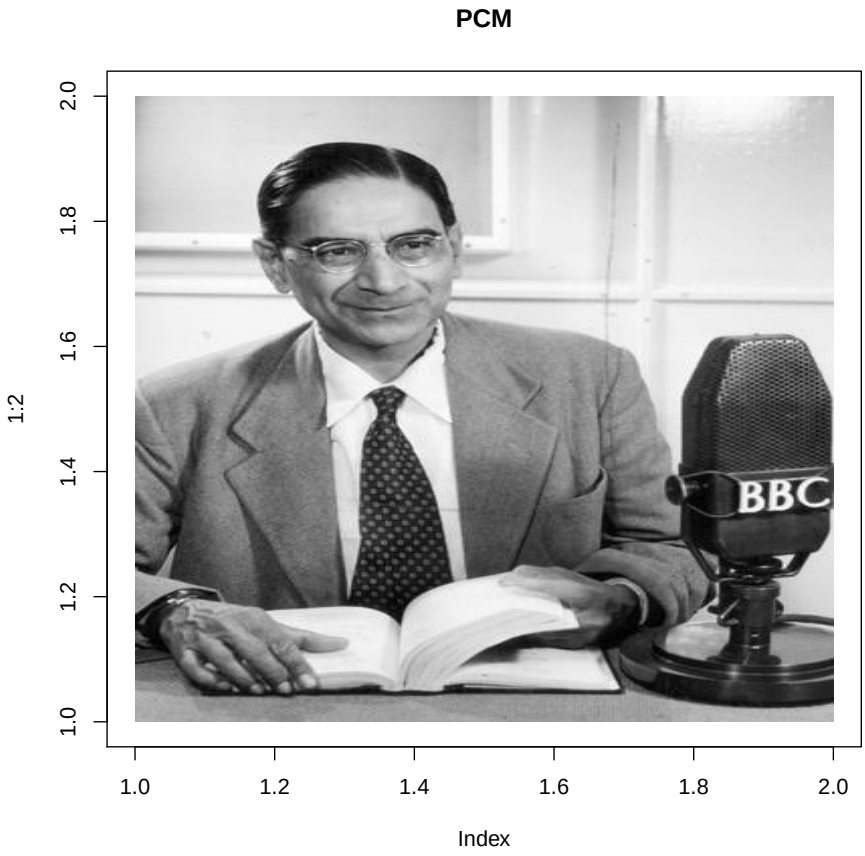
- (image compression) Now that the prototype vectors have been determined, the next step of VQ consists of image compression and transmission. Each vector in the original image is compared one-at-a-time to each of the prototype vectors in the codebook. The prototype that most closely resembles the input vector is selected, and its address is transmitted/represented. That is the compressed version of the image is only the codebook consisting of r vectors (instead of mn/q vectors) each of which is q dimensional.

- (image decompression) The final step of VQ consists of getting back the image, which will be obviously an approximation to the original one. i.e., a sequence of mn=q addresses, and decompressing it. Each of themn=q vectors will be approximated by the corresponding prototype vector in the codebook (that is the cluster center which it belongs to).

Now we proceed with the grayscale image of P.C. Mahalanobis.

1. First we read the image in R and write a function to plot the original image in R.

```
Loading required package:  jpeg
Warning:  package 'jpeg' was built under R version 3.5.3
```

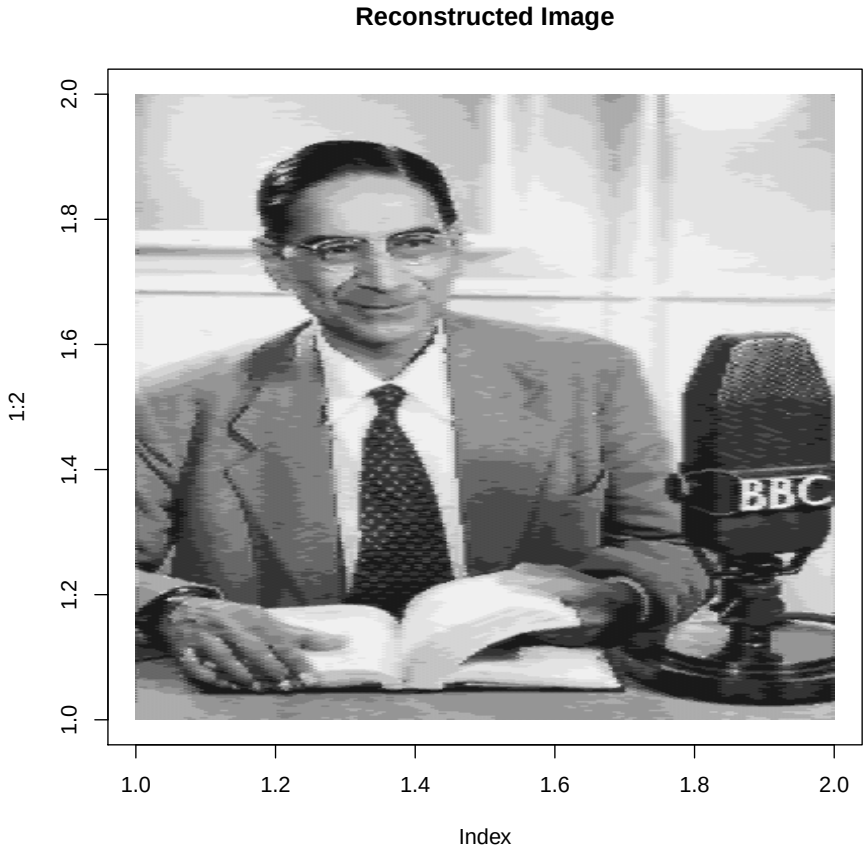


The R-Code for this is:-

```
require(jpeg)
x=as.vector(readJPEG("C:\\Users\\Souvik Saha\\Documents\\Study material\\Presidency Universi
drawPic = function(y,title)
{
dim(y) = c(430,346)
plot(1:2,ty='n',main=title)
rasterImage(as.raster(y),1,1,2,2)
}
drawPic(x,"PCM")
```

2. Now we implement a version of VQ with the choice of q=4 and r=15.

```
[1] 430 346
```



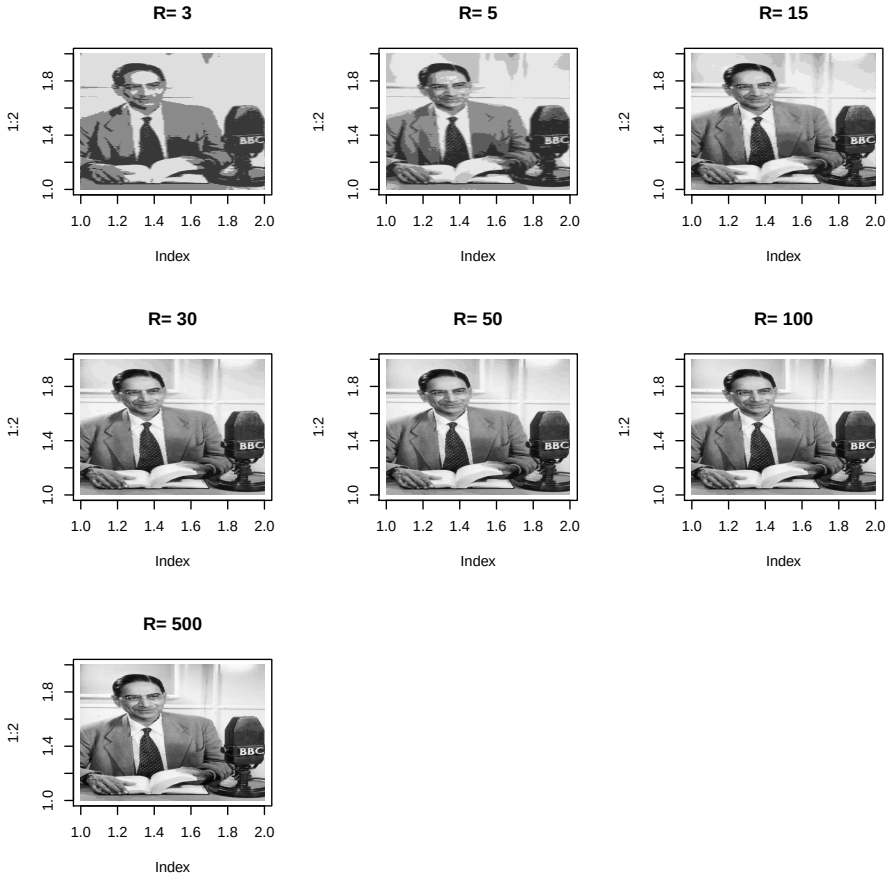
The R code for this is:-

```
#Prototype Creation
m=nrow(x)
n=ncol(x)
q=4
V=matrix(as.vector(t(x)),nrow=(m*n)/q,ncol=q,byrow=T)
r=15
C=kmeans(V,r,algorithm="Lloyd",iter=600)
codebook=C$centers
#Image Compression & Decomposition
Z=codebook[C$clus,]
v.new=as.vector(t(Z))
x.new=matrix(v.new,nrow=m,byrow=T)
dim(x.new)
#Drawing the reconstructed image
drawPic(x.new,"recon_photo")
```

Comments:-

Since we are finding an approximation of the original image the reconstructed image is a little blurred,hazy and less prominent than the actual one.

3.Now we repeat the process of reconstructing the image for some choices of r say 5,3,15,30,50,100,500.



The R-Code for this is:-

```
recon.pic<-function(A,q,r)
{
m=nrow(A)
n=ncol(A)
V=matrix(as.vector(t(A)),nrow=(m*n)/q,ncol=q,byrow=T)
C=kmeans(V,r,algorithm="Lloyd",iter=600)
codebook=C$centers
Z=codebook[C$clus,]
```

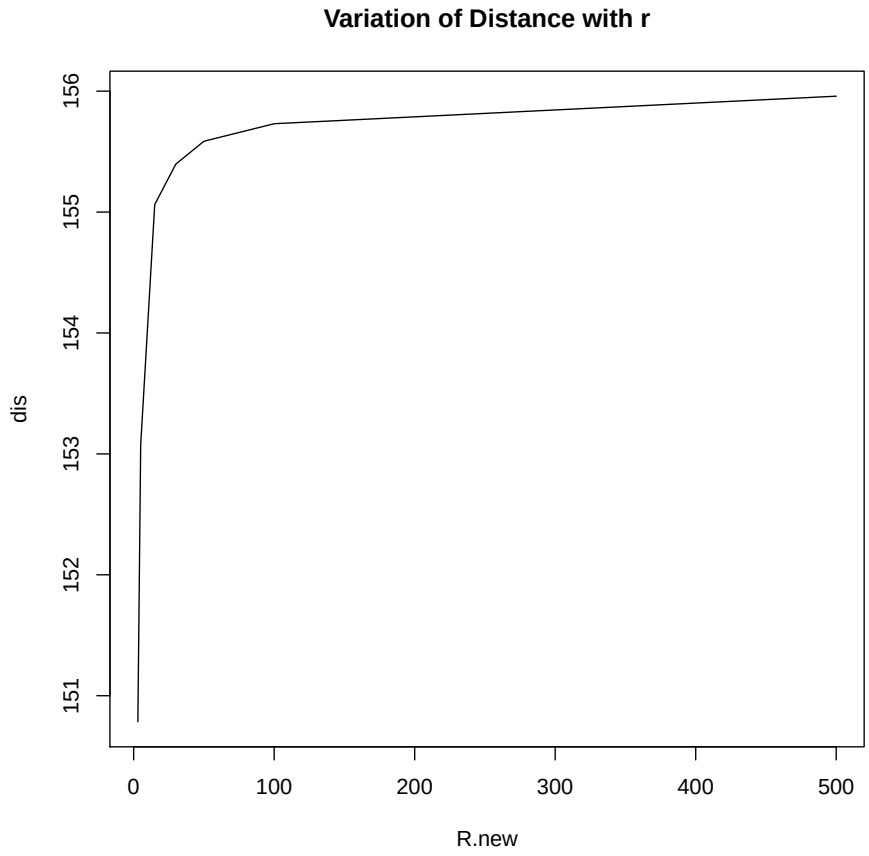
```
v.new=as.vector(t(Z))
x.new=matrix(v.new,nrow=m,byrow=T)
dim(x.new)
PIC=drawPic(x.new,paste("R=",r))
return(PIC)
}
par(mfrow=c(3,3))
R=c(3,5,15,30,50,100,500)
for(i in 1:length(R))
{
        recon.pic(x,4,R[i])
}
```

Comments:-

We know a small value of r indicates large compression but greater approximation.Hence with the increase in r we see the reconstructed image is getting clearer and exact with the original image i.e the blurriness in the image is decreasing.

4.Now we calculate the Euclidean distance between the original image and the estimated image and study the variation with changes in the value of r.

```
[1] 150.7843 153.0724 155.0621 155.3966 155.5856 155.7307 155.9583
```



The R code for doing this is:-

```
#First we create a function for calculating the distance
euclid<-function(a,b)
{
        Dis=sqrt(sum((a-b)^2))
        return(Dis)
}
#Now using this function we create another function for finding the Euclidean distance for
euclid.pic<-function(A,q,r)
{
x=as.vector(t(A))
m=nrow(A);n=ncol(A)
V=matrix(as.vector(t(A)),nrow=(m*n)/q,ncol=q,byrow=T)
C=kmeans(V,r,algorithm="Lloyd",iter=600)
codebook=C$centers
Z=codebook[C$clus,]
```

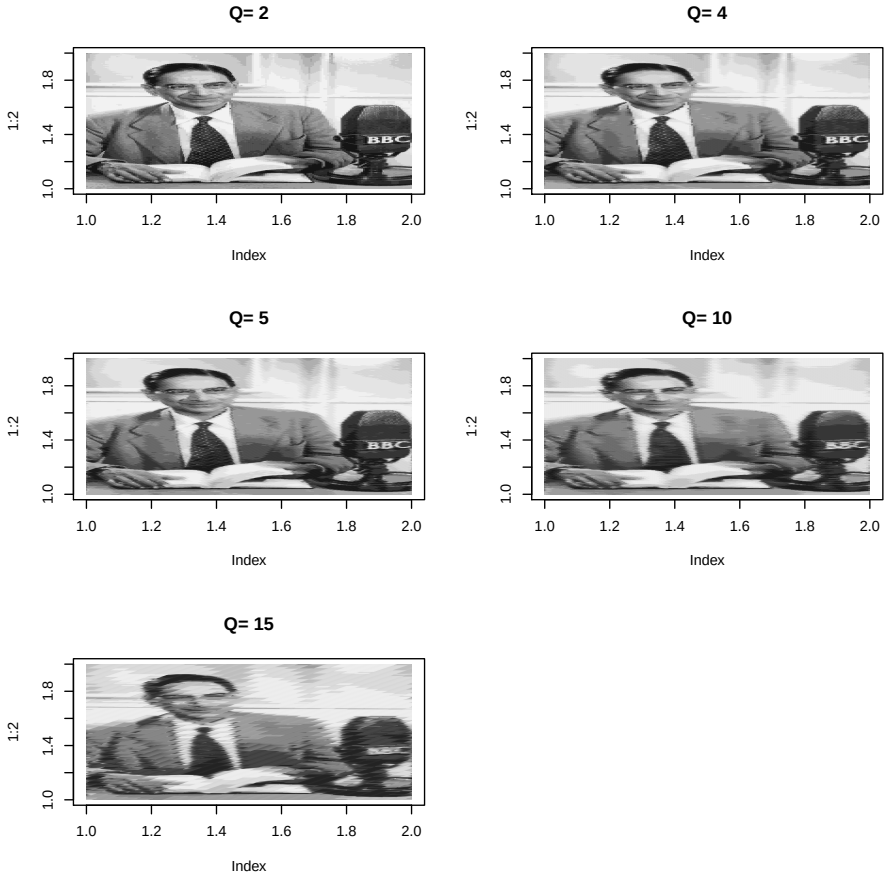
```
v.new=as.vector(t(Z))
x.new=matrix(v.new,nrow=m,byrow=T)
D=euclid(x,x.new)
return(D)
}
R.new=c(3,5,15,30,50,100,500)
dis=NULL
q=4
for(i in 1:length(R.new))
{
        dis[i]=euclid.pic(x,q,R.new[i])
}
dis
plot(R.new,dis,type="l",,main='Variation of Distance with r')
```

Comments:-

The graph shows a sharp increase of the Euclidean distance between the original and estimated images with increase in r.

5.Now we repeat the process VQ keeping r fixed say at 15 and for different values of q say (2,4,5,10,15)

```
Warning in matrix(as.vector(t(A)), nrow = (m * n)/q, ncol = q, byrow
= T): data length [148780] is not a sub-multiple or multiple of the
number of rows [9918]
Warning in matrix(v.new, nrow = m, byrow = T): data length [148770]
is not a sub-multiple or multiple of the number of rows [430]
```



The R Code for this is:-

```
recon.pic<-function(A,q,r)
{
m=nrow(A);n=ncol(A)
V=matrix(as.vector(t(A)),nrow=(m*n)/q,ncol=q,byrow=T)
C=kmeans(V,r,algorithm="Lloyd",iter=600)
codebook=C$centers
Z=codebook[C$clus,]
v.new=as.vector(t(Z))
x.new=matrix(v.new,nrow=m,byrow=T)
dim(x.new)
PIC=drawPic(x.new,paste("Q=",q))
return(PIC)
}
Q=c(2,4,5,10,15)
r.new=15
```

```
par(mfrow=c(3,2))
for(i in 1:length(Q))
{
        recon.pic(x,Q[i],r.new)
}
```

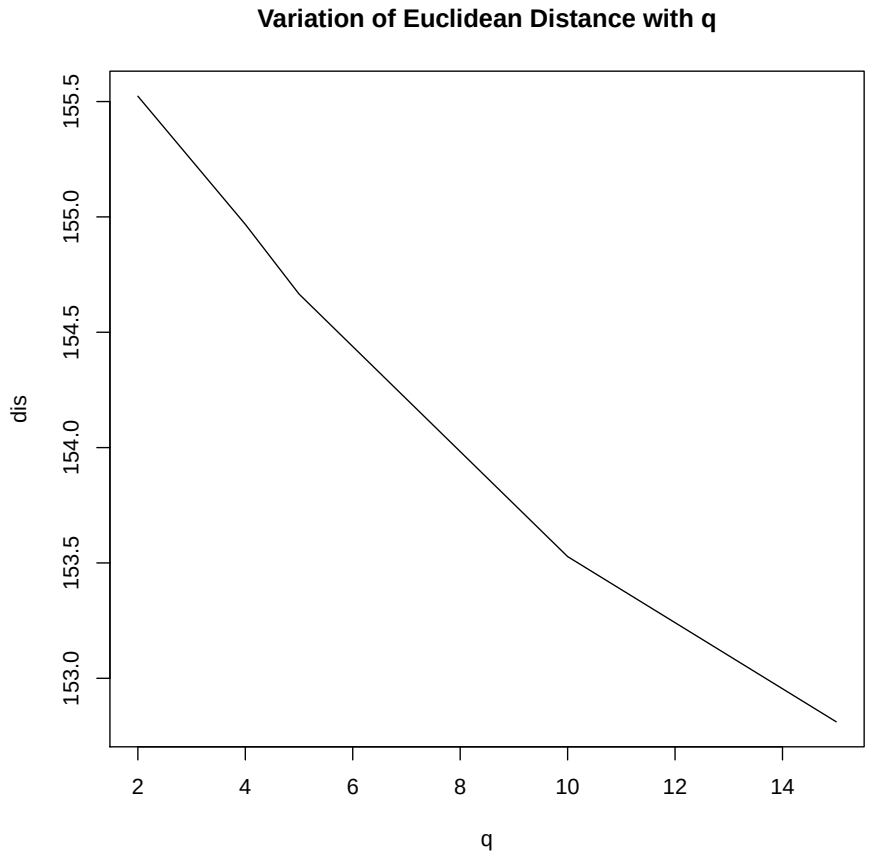
Comments:-

With the increase in the value of q we see the reconstructed images become more hazy,blurred and less prominent with respect to the original image.

6.Now we calculate the Euclidean distance between the original and reconstructed images for various values of q.

```
Warning in matrix(as.vector(t(A)), nrow = (m * n)/q, ncol = q, byrow
= T): data length [148780] is not a sub-multiple or multiple of the
number of rows [9918]
Warning in matrix(v.new, nrow = m, byrow = T): data length [148770]
is not a sub-multiple or multiple of the number of rows [430]

[1] 155.5232 154.9678 154.6657 153.5276 152.8114
```



The R code for this is:-

```
R.new=15
dis=NULL
q=c(2,4,5,10,15)
for(i in 1:length(q))
{
        dis[i]=euclid.pic(x,q[i],R.new)
}
dis
plot(q,dis,type="l",main='Variation of Euclidean Distance with q')
```

Comments:-

There is a sharp decrease in the Euclidean distance between original and estimated images with increase in the value of q.

REFERENCES USED FOR THIS PROBLEM:-

(a) The Elements of Statistical Learning, Hastie, Tibshirani, et.al.

(b) Image compression using learned vector quantization, K. Ferens, W.Lehn, and W. Kinsner.

ACKNOWLEDGEMENT

I would like to thank Prof Atanu Kumar Ghosh and the authors of the books and papers I have used as references for this project.