Pattern Recognition Theory 18794 Homework 2

Mengwen He (Alex)

October 23, 2016

Problem 1

PCA vs LDA. Given the following dataset for a 2-class problem with 2-D features:

$$c_1 = \left\{ \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\},\ c_2 = \left\{ \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \begin{bmatrix} 6 \\ 4 \end{bmatrix} \right\}$$

1. Using global PCA, find the best direction onto which the data will be projected on.

```
figure;
hold on;

%load data
c1=[2,2,2;1,2,3];
c2=[4,5,6;3,3,4];
c=[c1,c2];

%(1) calculate mean
mu=mean(c,2);

%draw data points
plot(c1(1,:),c1(2,:),'rx',c2(1,:),c2(2,:),'bo');
axis([0,7,0,7]);
axis equal;

%(2) centralize data points
cc1=c1-repmat(mu,1,size(c1,2));
cc2=c2-repmat(mu,1,size(c2,2));
cc=[cc1,cc2];

%(3) calculate covariant matrix S
S=cc*cc'./size(cc,2);
%(4) calculate eigenvectors and eigenvalues of S;
[V,D]=eig(S);
%(5) sort eigenvalues and eigenvectors;
[E,I]=sort(diag(D),'descend');
V=V(:,I);

%(6) calcuate w, the normal vector of the projection line
tempw=cross([0;0;1],[V(:,1);0]);
w=tempw(1:2,1);
w=w/norm(w); %the normal vector of the desired projection line
%(7) calculate w0
w0=-w'*mu;

%draw projection line
fh=@(x1,x2) w(1)*x1+w(2)*x2+w0;
ezplot(fh,[0,7,0,7]);
axis equal;

%projected points
pc=V(:,1)*(V(:,1)'*cc)+repmat(mu,1,size(cc,2));

%draw projections
for i=1:size(pc,2)
```

```
    plot([c(1,i);pc(1,i)],[c(2,i);pc(2,i)],'black');
end

%(8) calculate MSE
MSE=sum(sum((c-pc).^2,1),2)/size(c,2)

%(9) calculate Fisher ratio
pcc1=V(:,1)'*cc1;
pcc2=V(:,1)'*cc2;
pmu1=mean(pcc1,2);
pmu2=mean(pcc2,2);
pccc1=(pcc1-pmu1);
pccc2=(pcc2-pmu2);
ps1=(pccc1*pccc1')/size(pccc1,2);
ps2=(pccc2*pccc2')/size(pccc2,2);
FR=(pmu1-pmu2)^2/(ps1+ps2)

%set title
title(sprintf('[%f,%f]*x+%f=0; MSE=%f, FR=%f',w(1),w(2),w0,MSE,FR));
```

./matlab/P_1_1.m

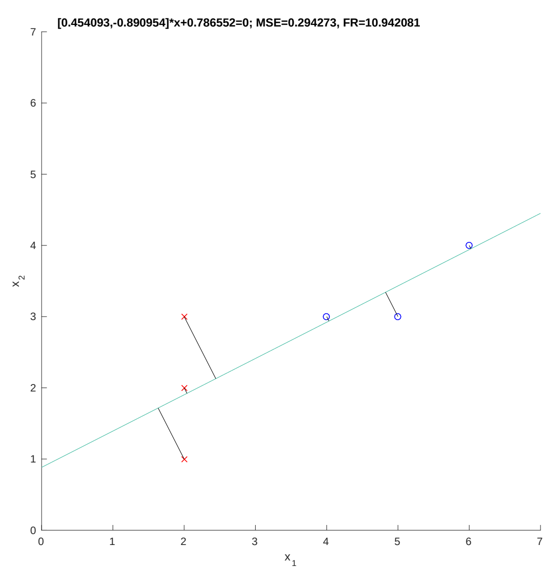
(a) Express the equation of the line along this direction passing through the samples mean in the following form: $\vec{w}^T\vec{x} + w_0 = 0$, where

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ and } \vec{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

and w_0 is a scalar known as the bias. Plot the $\vec{w}^T\vec{x} + w0 = 0$ line along with all the sample points as well the line along $\vec{w}$ which recall starts from the mean.

$$\vec{w} = \begin{bmatrix} 0.4541 \\ -0.8910 \end{bmatrix}, \; w_0 = 0.7866$$

(b) Project and reconstruct all the sample points. Plot the reconstructed points.



(c) Find the total mean square error MSE for all sample points (between the original points and the reconstructed points).

$$MSE = 0.2943$$

(d) Find the Fisher Ratio for this projection defined by

$$FR = \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where m_i is the mean of the projected samples of class i, and σ_i^2 is the equivalent variance. You can compute the FR on the projected 1-D points (rather than the reconstructed points which are 2-D vectors).

$$FR = 10.9421$$

2. Using Fisher Linear Discriminant Analysis (LDA), determine the best one-dimensional space onto which the above data should be projected.

```
figure;
hold on;
%load data
c1=[2,2,2;1,2,3];
c2=[4,5,6;3,3,4];
c=[c1,c2];

%(1) calculate global mean and local means
mu=mean(c,2);
mu1=mean(c1,2);
mu2=mean(c2,2);

%(2) draw data points
plot(c1(1,:),c1(2,:),'rx',c2(1,:),c2(2,:),'bo');
axis([0,7,0,7]);
axis equal;

%(3) calcuate Sb
Sb=(mu1-mu2)*(mu1-mu2)';

%(4) calculate S1
S1=zeros(size(c1,1));
for i=1:size(c1,2)
    S1=S1+(c1(:,i)-mu1)*(c1(:,i)-mu1)';
end
S1=S1./size(c1,2);
%(5) calculate S2
S2=zeros(size(c2,1));
for j=1:size(c2,2)
    S2=S2+(c2(:,j)-mu2)*(c2(:,j)-mu2)';
end
S2=S2./size(c2,2);
%(6) calculate Sw
Sw=S1+S2;

%(7) calcuate generalized eigenvectors and eigenvalues of Sb*V=Sw*V*D
%[V,D]=eigs(inv(Sw)*Sb);
[V,D]=eigs(Sb,Sw);

%(8) sort eigenvalues and eigenvectors
[E,I]=sort(diag(D),'descend');
V=V(:,I);

%(9) normalize the desired eigenvector
V(:,1)=V(:,1)/norm(V(:,1));
%(10) calcuate w, the normal vector of the projection line
tempw=cross([0;0;1],[V(:,1);0]);
w=tempw(1:2,1);
w=w/norm(w); %the normal vector of the desired projection line
%(11) calcuate w0
w0=-w'*mu;

%draw projection line
fh=@(x1,x2) w(1)*x1+w(2)*x2+w0;
ezplot(fh,[0,7,0,7]);
axis equal;

%projected points
pc=V(:,1)*(V(:,1)'*cc)+repmat(mu,1,size(cc,2));

%draw projections
```

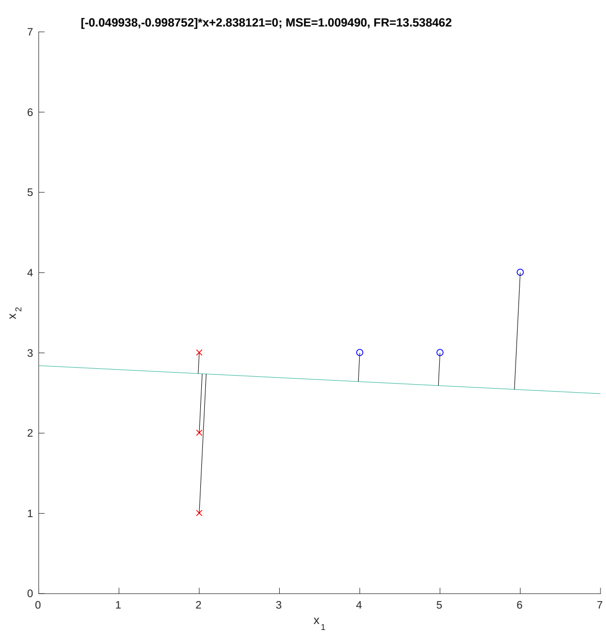
```matlab
for i=1:size(pc,2)
    plot([c(1,i);pc(1,i)],[c(2,i);pc(2,i)],'black');
end

%(12) calcuate MSE
MSE=sum(sum((c-pc).^2,1),2)/size(c,2)

%(13) calcuate Fisher ratio
cc1=c1-repmat(mu,1,size(c1,2));
cc2=c2-repmat(mu,1,size(c2,2));
cc=[cc1,cc2];

pcc1=V(:,1)'*cc1;
pcc2=V(:,1)'*cc2;
pmu1=mean(pcc1,2);
pmu2=mean(pcc2,2);
pccc1=(pcc1-pmu1);
pccc2=(pcc2-pmu2);
ps1=(pccc1*pccc1')/size(pccc1,2);
ps2=(pccc2*pccc2')/size(pccc2,2);
FR=(pmu1-pmu2)^2/(ps1+ps2)

%set title
title(sprintf('[%f,%f]*x+%f=0; MSE=%f, FR=%f',w(1),w(2),w0,MSE,FR));
```

./matlab/P_1_2.m

(a) Express the equation of the line along this direction passing through the samples mean in the following form: $\vec{w}^T\vec{x} + w0 = 0$. Plot the $\vec{w}^T\vec{x} + w0 = 0$ line along with all the sample points.

$$\vec{w} = \begin{bmatrix} -0.0499 \\ -0.9988 \end{bmatrix}, \; w_0 = 2.8381$$

(b) Project and reconstruct all the sample points. Plot the reconstructed points. Make sure they lie on a line.



(c) Find the total MSE for all sample points (between the original points and the reconstructed points). Compare that to the total MSE found in the first question.

$$MSE = 1.0095$$

(d) Find the Fisher Ratio for this projection. You can compute the FR on the projected points (rather than the reconstructed points which are 2-D vectors). Compare that to the FR found in part 1. Interpret your result.

$$FR = 13.5385$$

Problem 2

Another goal of PCA is to obtain the linear subspace which minimizes the projection error caused by dimension reduction. The two goals, maximum variance and minimum error, have the same formulation as PCA.

1. Assume all data samples $\{\vec{x}_1, \ldots, \vec{x}_n\}$ are centered. Formulate the optimization problem to solve for a vector $\vec{w}$ spanning the linear subspace which minimizes the sum of the squared reconstruction errors:

$$\sum_{i=1}^{n} d_i^2$$

where d_i, the reconstruction error of $\vec{x}_i$, is defined as the distance between $\vec{x}_i$ and its projection onto the linear subspace. Assume that $\vec{w}$ has a unit norm since we are interested only in the direction of the vector $\vec{w}$. The answer must be written in terms of $\vec{w}$ and $\vec{x}_i$ without d_i.

Answer:

$\because$

$$\begin{array}{rcl}
d_i^2(\vec{w}) & = & ||\vec{x}_i - \vec{w}\vec{x}_i^T\vec{w}||_2^2 \\
& = & (\vec{x}_i - \vec{w}\vec{x}_i^T\vec{w})^T(\vec{x}_i - \vec{w}\vec{x}_i^T\vec{w}) \\
& = & \vec{x}_i^T\vec{x}_i - 2\vec{x}_i^T\vec{w}\vec{x}_i^T\vec{w} + \vec{w}^T\vec{x}_i\vec{x}_i^T\vec{w} \\
& = & \vec{x}_i^T\vec{x}_i - 2\vec{w}^T(\vec{x}_i\vec{x}_i^T)\vec{w} + \vec{w}^T(\vec{x}_i\vec{x}_i^T)\vec{w} \\
& = & \vec{x}_i^T\vec{x}_i - \vec{w}^T(\vec{x}_i\vec{x}_i^T)\vec{w} \\
\sum_{i=1}^n d_i^2(\vec{w}) & = & \sum_{i=1}^n \vec{x}_i^T\vec{x}_i - \vec{w}^T\sum_{i=1}^n(\vec{x}_i\vec{x}_i^T)\vec{w} \\
\text{where } ||\vec{w}|| & = & 1 \\
& = & \vec{w}^T\vec{w} = 1
\end{array}$$

$\therefore$ use Lagrange Multipliers to get the optimized $\vec{w}$:

$$\begin{array}{rcl}
L(\vec{w}, \lambda) & = & \sum_{i=1}^n d_i^2(\vec{w}) - \lambda(1 - \vec{w}^T\vec{w}) \\
& = & \sum_{i=1}^n \vec{x}_i^T\vec{x}_i - \vec{w}^T\sum_{i=1}^n(\vec{x}_i\vec{x}_i^T)\vec{w} - \lambda(1 - \vec{w}^T\vec{w}) \\
\frac{\partial L}{\partial \lambda} & = & \vec{w}^T\vec{w} - 1 = 0 \\
\frac{\partial L}{\partial \vec{w}} & = & -2(\sum_{i=1}^n \vec{x}_i\vec{x}_i^T)\vec{w} + 2\lambda\vec{w} = 0 \\
& \Rightarrow & \left(\sum_{i=1}^n \vec{x}_i\vec{x}_i^T\right)\vec{w} = \lambda\vec{w}
\end{array}$$

$\therefore$ λs are the eigenvalues, and $\vec{w}$s are the corresponding eigenvectors of $\sum_{i=1}^n \vec{x}_i\vec{x}_i^T$.

$\because$

$$\left(\sum_{i=1}^{n} \vec{x}_i\vec{x}_i^T\right)\vec{w} = \lambda\vec{w} \Rightarrow \vec{w}^T\left(\sum_{i=1}^{n} \vec{x}_i\vec{x}_i^T\right)\vec{w} = \lambda$$

and

$$\sum_{i=1}^{n} \vec{x}_i^T\vec{x}_i \text{ is constant}$$

$\therefore$ to minimize $\sum_{i=1}^n d_i^2(\vec{w})$, we need to choose the $\vec{w}^*$ to maximize $\vec{w}^T\left(\sum_{i=1}^n \vec{x}_i\vec{x}_i^T\right)\vec{w} = \lambda = \lambda_{max}$.

2. From your answer in (a), show that the optimization in part (a) is equivalent to the PCA optimization using the following equivalence:

$$\Sigma = \sum_{i}^{n} \vec{x}_i\vec{x}_i^T$$

Answer:

$\because$ PCA is to choose the $\vec{w}^*$ to maximize the variant after projection $\vec{w}^T\Sigma\vec{w} = \lambda = \lambda_{max}$.

$\therefore$ they are equivalent.

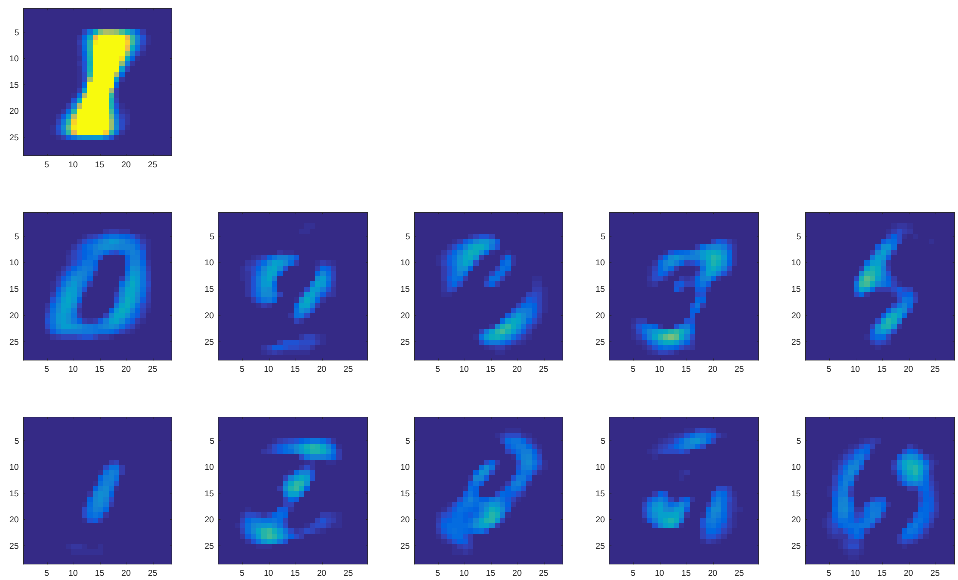
Problem 3

You are to implement PCA from scratch and not use the MATLAB built in functions. You are allowed to use SVD and/or eigenvalue decomposition. Please submit a print out of your code aw well.

1. Plot the mean image of digit 1 and plot the first 5 global PCA vectors corresponding to the dataset corresponding to two cases: using Gram Matrix Trick and without Gram Matrix trick. Do not forget to remove the mean before you run PCA. Measure the time taken for the PCA vector computation in each case.

```matlab
load('mnist_all.mat');

figure;

%calculate digit 1's mean image and reshape it
mu1=mean(train1,1)';
mu1image=reshape(mu1,[28,28])';

%draw digit 1's mean image
subplot(3,5,1);
image(mu1image);

%collect part of training data, controlled by samplenum
samplenum=100;
train=double([train0(1:samplenum,:);train1(1:samplenum,:);train2(1:samplenum,:);train3(1:
    samplenum,:);train4(1:samplenum,:);
    train5(1:samplenum,:);train6(1:samplenum,:);train7(1:samplenum,:);train8(1:samplenum,:);
    train9(1:samplenum,:)]');
%calculate the global mean
mu=mean(train,2);

%centralize all data
ctrain=train-repmat(mu,1,size(train,2));

% Withouth Gram trick PCA to get the first 5 eigenvectos
tic
S=ctrain*ctrain'./size(ctrain,2);
[V,D]=eigs(S,5);
toc

%draw the first 5 eigenvector images
for i=1:5
    eigenimage=reshape(V(:,i),[28,28])'.*255;
    subplot(3,5,i+5);
    image(eigenimage);
end

% With Gram trick PCA to get the first 5 eigenvectos
tic
G_S=ctrain'*ctrain./size(ctrain,2);
[G_U,G_D]=eigs(G_S,5);
G_V=normc(ctrain*G_U);
toc

%draw the first 5 eigenvector images
for i=1:5
    eigenimage=reshape(G_V(:,i),[28,28])'.*255;
    subplot(3,5,i+10);
    image(eigenimage);
end
```

./matlab/P_3_1.m



The first row is the digit 1's mean image; the second row is the first 5 eigenvector images without Gram trick; the third row is the first 5 eigenvector images with Gram trick.

2. Comment on the time taken in each case. Does it make sense to use the Gram trick for this particular dataset?
Answer:

   - When samplenum=1000 (total number is 1000*10=10000): w/o Gram costs 0.100550s, and w/ Gram costs 4.579334s.
   - When samplenum=100 (total number is 100*10=1000): w/o Gram costs 0.037760s, and w/ Gram costs 0.038084s.
   - When samplenum=10 (total number is 10*10=100): w/o Gram costs 0.013963s, and w/ Gram costs 0.042149s.
   - When samplenum=1 (total number is 10*10=100): w/o Gram costs 0.008933s, and w/ Gram costs 0.003651s.

   Therefore, if $N << d$, Gram trick is helpful for the PCA algorithm speed.

3. Pick any random image from the dataset which can be any image from any class and project onto the eigen space (the global PCA eigen space that you obtained in part 1 of this problem) and reconstruct it using the first n eigen vectors (not just the nth vector but all of them up until n) (for the two cases, Gram trick and without it) where $n = \{1, 2, 5, 10, 20\}$. Do not forget to remove the mean before projecting the image, and also to add it back once it has been reconstructed using only the first n eigen vectors.

   For each of the 5 reconstructions, compute the mean square error (MSE) of the reconstructed image. Note that the MSE between two vectors $\vec{a}$ and $\vec{b}$ is given by

$$MSE(\vec{a}, \vec{b}) = ||\vec{a} - \vec{b}||_2^2$$

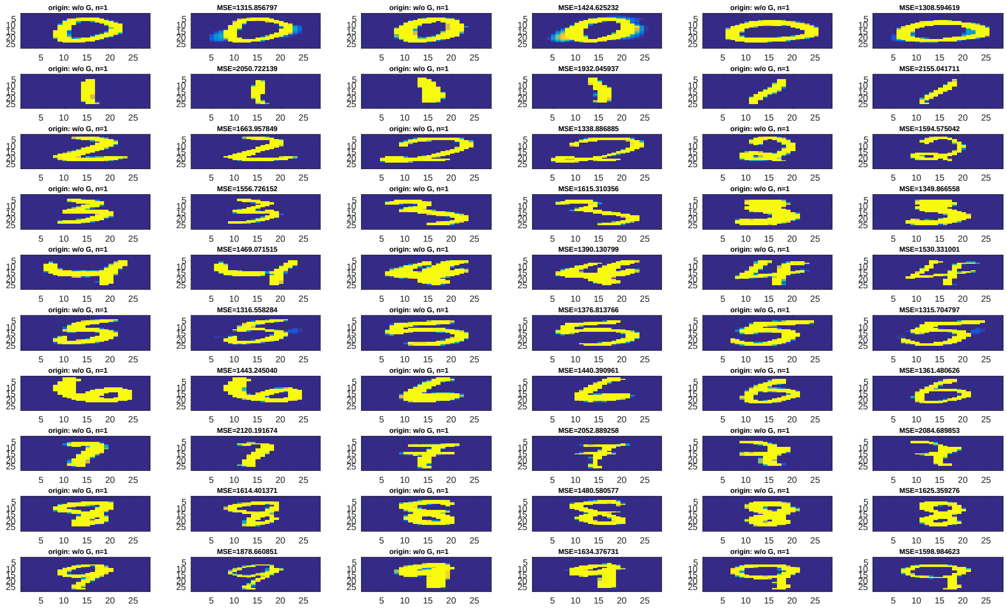
   Display all reconstructed images and the original in one figure, with the corresponding MSE value as the title of each sub-figure

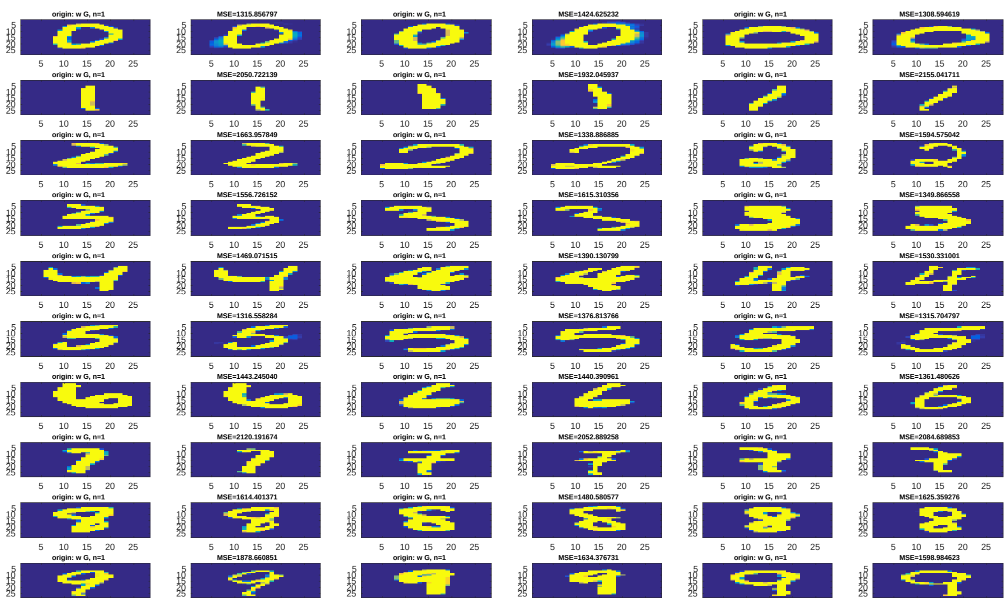
```
%select maxn first eigenvectors for reconstruction
n=[1,2,5,10,20];
maxn=max(n);

%collect part of training data, controlled by samplenum
samplenum=100;
```

```
train=double([train0(1:samplenum,:);train1(1:samplenum,:);train2(1:samplenum,:);train3(1:
    samplenum,:);train4(1:samplenum,:);
    train5(1:samplenum,:);train6(1:samplenum,:);train7(1:samplenum,:);train8(1:samplenum,:);
    train9(1:samplenum,:)]');
mu=mean(train,2);
ctrain=train-repmat(mu,1,size(train,2));

%PCA without Gram trick
S=ctrain*ctrain'./size(ctrain,2);
[V,D]=eigs(S,maxn);
%PCA with Gram trick
G_S=ctrain'*ctrain./size(ctrain,2);
[G_U,G_D]=eigs(G_S,maxn);
G_V=normc(ctrain*G_U);

%set how many test samples randomly extracted from each test data
m=3;

%collect test data
images=zeros(size(mu,1),10*m);
index=1;
images(:,index:index+m-1)=test0(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test1(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test2(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test3(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test4(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test5(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test6(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test7(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test8(randi([1 samplenum],1,m),:)';
index=index+m;
images(:,index:index+m-1)=test9(randi([1 samplenum],1,m),:)';

%For PCA without Gram trick
for nid=1:size(n,2)
    figure;
    %For all digits
    for i=1:10
        %For all test data of each digit
        for j=1:m
            id=(i-1)*m+j;
            %draw original image firstly
            subplot(10,2*m,2*(id-1)+1);
            originimageshow=reshape(images(:,id),[28,28])';
            image(originimageshow);
            title(sprintf('origin: w/o G, n=%d',n(nid)),'FontSize',8);
            %reconstruct image with n(nid) first eigenvectos
            muimage=images(:,id)-mu;
            eigenimage=muimage;
            for k=1:n(nid)
                coeff=V(:,k)'*muimage;
                eigenimage=eigenimage+V(:,k)*coeff;
            end
            %draw reconstrcuted image secondly
            subplot(10,2*m,2*(id-1)+2);
            eigenimageshow=reshape(eigenimage,[28,28])';
            image(eigenimageshow);
            %calculate MSE
            MSE=norm(images(:,id)-eigenimage);
            title(sprintf('MSE=%f',MSE),'FontSize',8);
        end
    end
end
```

```
%For PCA with Gram trick
for nid=1:size(n,2)
    figure;
    %For all digits
    for i=1:10
        %For all test data of each digit
        for j=1:m
            id=(i-1)*m+j;
            %draw original image firstly
            subplot(10,2*m,2*(id-1)+1);
            originimageshow=reshape(images(:,id),[28,28])';
            image(originimageshow);
            title(sprintf('origin: w G, n=%d',n(nid)),'FontSize',8);
            %reconstruct image with n(nid) first eigenvectos
            muimage=images(:,id)-mu;
            eigenimage=muimage;
            for k=1:n(nid)
                coeff=V(:,k)'*muimage;
                eigenimage=eigenimage+G_V(:,k)*coeff;
            end
            %draw reconstrcuted image secondly
            subplot(10,2*m,2*(id-1)+2);
            eigenimageshow=reshape(eigenimage,[28,28])';
            image(eigenimageshow);
            MSE=norm(images(:,id)-eigenimage);
            %calculate MSE
            title(sprintf('MSE=%f',MSE),'FontSize',8);
        end
    end
end
```

./matlab/P_3_3.m


origin: w/o G, n=1
MSE=1315.856797
MSE=1424.625232
MSE=1308.594619
MSE=2050.722139
MSE=1932.045937
MSE=2155.041711
MSE=1663.957849
MSE=1338.886885
MSE=1594.575042
MSE=1556.726152
MSE=1615.310356
MSE=1349.866558
MSE=1469.071515
MSE=1390.130799
MSE=1530.331001
MSE=1316.558284
MSE=1376.813766
MSE=1315.704797
MSE=1443.245040
MSE=1440.390961
MSE=1361.480626
MSE=2120.191674
MSE=2052.889258
MSE=2084.689853
MSE=1614.401371
MSE=1480.580577
MSE=1625.359276
MSE=1878.660851
MSE=1634.376731
MSE=1598.984623


origin: w G, n=1
MSE=1315.856797
MSE=1424.625232
MSE=1308.594619
MSE=2050.722139
MSE=1932.045937
MSE=2155.041711
MSE=1663.957849
MSE=1338.886885
MSE=1594.575042
MSE=1556.726152
MSE=1615.310356
MSE=1349.866558
MSE=1469.071515
MSE=1390.130799
MSE=1530.331001
MSE=1316.558284
MSE=1376.813766
MSE=1315.704797
MSE=1443.245040
MSE=1440.390961
MSE=1361.480626
MSE=2120.191674
MSE=2052.889258
MSE=2084.689853
MSE=1614.401371
MSE=1480.580577
MSE=1625.359276
MSE=1878.660851
MSE=1634.376731
MSE=1598.984623