PURDUE UNIVERSITY

ECE 637 Laboratory Exercise 5
Eigen-decomposition of Images

Tong Shen

February 24, 2017

1 MULTIVARIATE GAUSSIAN DISTRIBUTIONS AND WHITENING

1.1 GENERATING GAUSSIAN RANDOM VECTORS

In this section, our goal will be to use Matlab to generate independent Gaussian random vectors, which have certain covariance. The formula is:

$$X = E\Lambda^{1/2}W \tag{1.1}$$

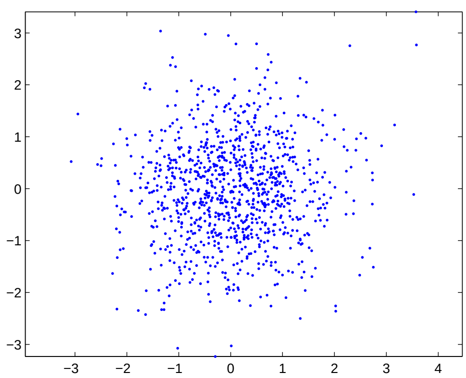

Figure 1.1: Plot of X, uncorrelated Guassian Randome Vector

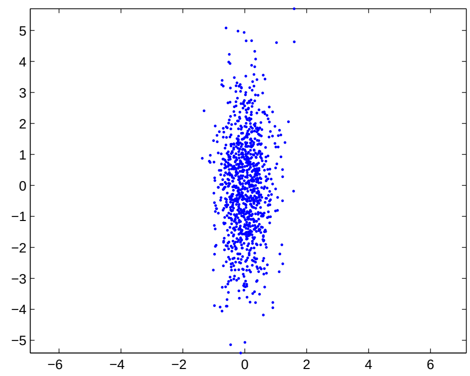

Figure 1.2: Plot of $\tilde{X}$, Scaled Randome Vector

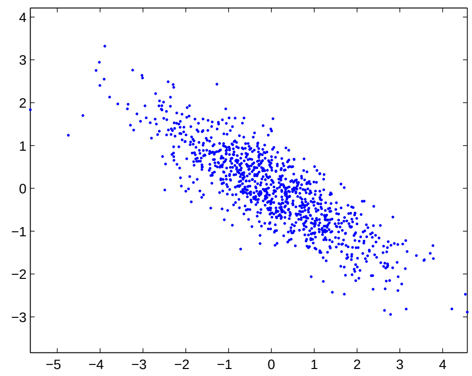

Figure 1.3: Plot of W, Randome Vectors with a Covariance R

1.2 Covariance Estimation and Whitening

In this section, we will use some method to estimate the covariance and get the decorrelation done.
The given theoretical value of the covariance matrix, R_x is as following:

$$R_x = \begin{bmatrix} 2 & -1.2 \\ -1.2 & 1 \end{bmatrix} \tag{1.2}$$

While my numerical covariance estimate $\hat{R}_X$ is:

$$\hat{R}_x = \begin{bmatrix} 1.989 & -1.147 \\ -1.147 & 0.9414 \end{bmatrix} \tag{1.3}$$

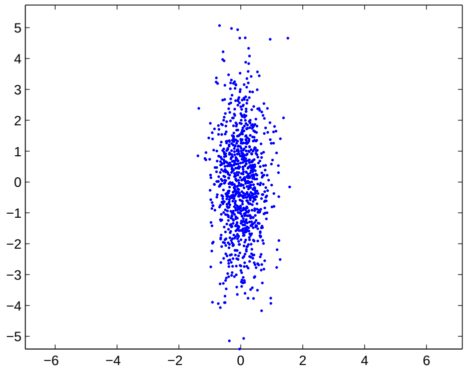

Figure 1.4: Plot of $\tilde{X}$, the decorrelated random vectors

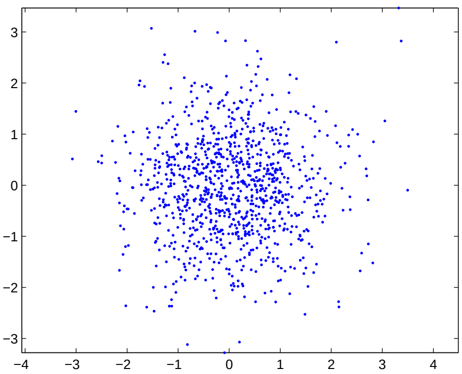

Figure 1.5: Plot of W_i, whittened random vectors

The numerical covariance estimate $\tilde{R}_W$ is:

$$\tilde{R}_w = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1.4}$$

2 Estimation of Eigenvectors and Eigenvalues Using the Singular Value Decomposition

In this part, nothing is needed to report.

3 Eigenimages, PCA, and Data Reduction

In this exercise we will compute eigenvectors associated with images (also called eigenimages) of typed English letters.

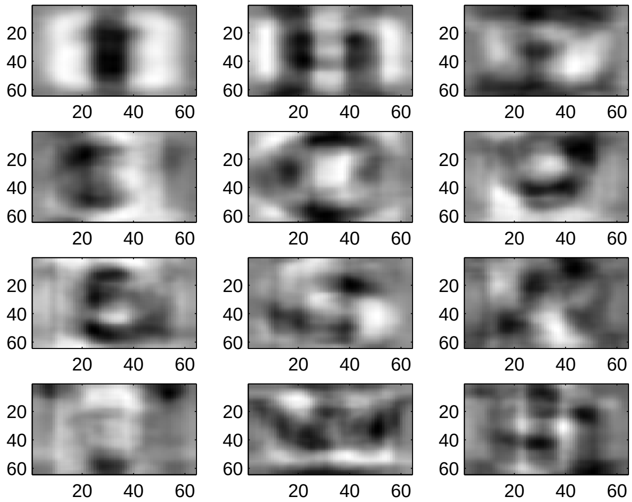

Figure 3.1: Display of the First 12 Eigenimages

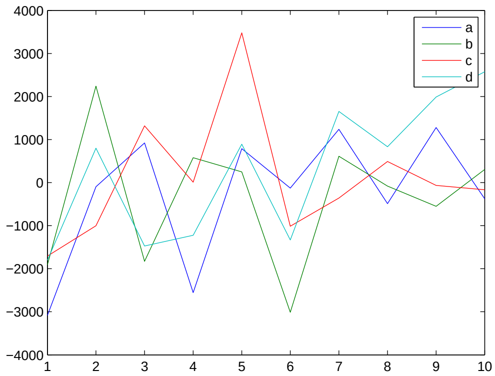

Figure 3.2: the Plots of Projection Coefficients vs. Eigenvector Number

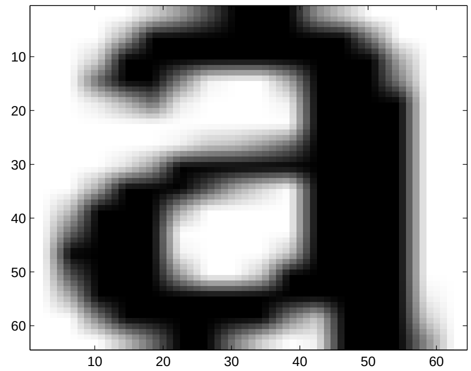

Figure 3.3: the Original Image

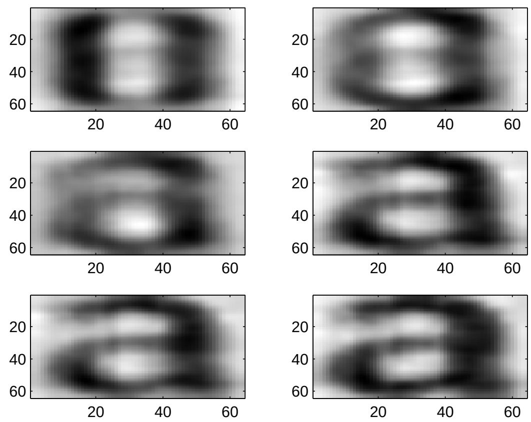

Figure 3.4: the 6 Resynthesized Versions.

4 IMAGE CLASSIFICATION

In this exercise, we will implement a classifier using the text character images from the last section as a training set. In this context the classifier will accept an input image, assumed to be of a lower-case English letter, and determine which of the 26 English letters it represents.

Table 4.1: mis-classified input image for $\mathrm{R_x}$

Actual Letter Image	Mis-classified Letter Image
d	a
j	y
l	i
n	v
p	e
q	a
u	a
y	v

Table 4.2: mis-classified input image of Λ_x

Actual Letters Images	Mis-classified Letters Images
i	l
y	v

Table 4.3: mis-classified input image of R_{wc}

Actual Letters Images	Mis-classified Letters Images
g	q
y	v

Table 4.4: mis-classified input image of Λ

Actual Letters Images	Mis-classified Letters Images
f	t
y	v

Table 4.5: mis-classified input image of I

Actual Letters Images	Mis-classified Letters Images
f	t
g	q
y	v

5 Code Listing

```
W = zeros(2,1000);
for i = 1:1000
W(1,i) = normrnd(0,1);
W(2,i) = normrnd(0,1);
end
figure(1)
plot(W(1,:),W(2,:),'.')
axis('equal')

R_x = [2 -1.2
-1.2 1];
[eigV eigs] = eig(R_x);

X_hat = eigs.^(1/2)*W;
figure(2)
plot(X_hat(1,:),X_hat(2,:),'.')
axis('equal')

X_i = eigV * X_hat;
figure(3)
plot(X_i(1,:),X_i(2,:),'.')
axis('equal')

R_n = zeros(2);
for i = 1:1000
R_n(1,1) = R_n(1,1) + X_i(1,i)*X_i(1,i)/999;
R_n(1,2) = R_n(1,2) + X_i(1,i)*X_i(2,i)/999;
R_n(2,1) = R_n(2,1) + X_i(2,i)*X_i(1,i)/999;
R_n(2,2) = R_n(2,2) + X_i(2,i)*X_i(2,i)/999;
end

[eigV_Rn eig_Rn] = eig(R_n);
Xi_uncorr = eigV_Rn'*X_i;
figure(4)
plot(Xi_uncorr(1,:),Xi_uncorr(2,:),'.')
axis('equal')

Xi_hat = inv(eig_Rn).^(1/2)*Xi_uncorr;
```

```
figure(5)
plot(Xi_hat(1,:),Xi_hat(2,:),'.')
axis('equal')

X_hi = zeros(2);
for i = 1:1000
X_hi(1,1) = X_hi(1,1) + Xi_hat(1,i)*Xi_hat(1,i)/999;
X_hi(1,2) = X_hi(1,2) + Xi_hat(1,i)*Xi_hat(2,i)/999;
X_hi(2,1) = X_hi(2,1) + Xi_hat(2,i)*Xi_hat(1,i)/999;
X_hi(2,2) = X_hi(2,2) + Xi_hat(2,i)*Xi_hat(2,i)/999;
end

[height length] = size(X);
meanImage = mean(X,2);
X_center = zeros(4096,312);
for i = 1:312
X_center(:,i) = X(:,i) - meanImage;
end
%X_center = X_center/sqrt(311);

[U S V] = svd(X_center,0);

figure(6)
for i = 1:12
X_hat12 = U(:,i);
temp = vec2mat(X_hat12,64)';
colormap(gray(256));
subplot(4,3,i)
imagesc(temp);
end


figure(7)
for i = 1:4
temp = vec2mat(X(:,i),64)';
colormap(gray(256));
subplot(2,2,i)
imagesc(temp);
end

figure(8)
Projection_co = zeros(4,10);
for i = 1:10
for j = 1:4
```

```
Projection_co(j,i) = U(:,i)'*(X(:,j) - meanImage);
end
end
t = 1:10;
plot(t,Projection_co);
legend('a','b','c','d')

m = [1 5 10 15 20 30];
figure(9)
for i = 1:6
temp = m(i);
X_resyn = U(:,1:temp)*U(:,1:temp)'*X_center(:,1);
x = vec2mat(X_resyn + meanImage,64)';
colormap(gray(256))
subplot(3,2,i)
imagesc(x);
end

figure(10)
x = vec2mat(X(:,1),64)';
colormap(gray(256))
imagesc(x);

A = U(:,1:10);

empty_cell=cell(26,2);
params=cell2struct(empty_cell,{'M','R'},2);

for j = 1:26
Y = zeros(10,12);
u = zeros(10,1);
for i = 1:12
Y(:,i) = A'*X_center(:,(i-1)*26 + j);
u = mean(Y,2);
R = 0;
for p = 1:12
R = R + (Y(:,p) - u)*(Y(:,p) - u)';
end
R = R/11;
end
params(j).M = u;
params(j).R = R;
end

```

```
datachar='abcdefghijklmnopqrstuvwxyz';
k = 1;
X_test = zeros(4096,26);
for ch = datachar
fname=sprintf('test_data/veranda/%s.tif',ch);
test_image = reshape(imread(fname),1,4096);
X_test(:,k) = test_image;
k = k + 1;
end
result = zeros(26,1);
Cri = zeros(26,26);
for i = 1:26
y = A'*(X_test(:,i) - meanImage);
for j = 1:26
t = y - params(j).M;
Cri(i,j) = t'*inv(params(j).R)*t + log(det(params(j).R));
end
[x loc] = min(Cri(i,:));
result(i) = loc;
end

result_1 = zeros(26,1);
Cri = zeros(26,26);
for i = 1:26
y = A'*(X_test(:,i) - meanImage);
for j = 1:26
t = y - params(j).M;
R_diag = diag(diag(params(j).R));
Cri(i,j) = t'*inv(R_diag)*t + log(det(R_diag));
end
[x loc] = min(Cri(i,:));
result_1(i) = loc;
end

result_2 = zeros(26,1);
Cri = zeros(26,26);
R_wc = zeros(10);
for i = 1:26
R_wc = R_wc + params(i).R;
end
R_wc = R_wc/26;

for i = 1:26
y = A'*(X_test(:,i) - meanImage);
```

```
for j = 1:26
t = y - params(j).M;
Cri(i,j) = t'*inv(R_wc)*t + log(det(R_wc));
end
[x loc] = min(Cri(i,:));
result_2(i) = loc;
end

R_wcDiag = diag(diag(R_wc));
result_3 = zeros(26,1);
Cri = zeros(26,26);
for i = 1:26
y = A'*(X_test(:,i) - meanImage);
for j = 1:26
t = y - params(j).M;
Cri(i,j) = t'*inv(R_wcDiag)*t + log(det(R_wcDiag));
end
[x loc] = min(Cri(i,:));
result_3(i) = loc;
end

result_4 = zeros(26,1);
Cri = zeros(26,26);
for i = 1:26
y = A'*(X_test(:,i) - meanImage);
for j = 1:26
t = y - params(j).M;
Cri(i,j) = t'*eye(10)*t + 1;
end
[x loc] = min(Cri(i,:));
result_4(i) = loc;
end
temp = zeros(26,2);
display('This is the misclassified letter using R_x')
for i = 1:26
if result(i) ~= i
temp(i,1) = (datachar(i));
temp(i,2) = (datachar(result(i)));
char(temp(i,:))
end
end

display('This is the misclassified letter using Lambda_x')

```

```
for i = 1:26
if result_1(i) ~= i
temp(i,1) = (datachar(i));
temp(i,2) = (datachar(result_1(i)));
char(temp(i,:))
end
end

display('This is the misclassified letter using R_wc')

for i = 1:26
if result_2(i) ~= i
temp(i,1) = (datachar(i));
temp(i,2) = (datachar(result_2(i)));
char(temp(i,:))
end
end

display('This is the misclassified letter using Lambda')

for i = 1:26
if result_3(i) ~= i
temp(i,1) = (datachar(i));
temp(i,2) = (datachar(result_3(i)));
char(temp(i,:))
end
end

display('This is the misclassified letter using I, the unit matrix')

for i = 1:26
if result_4(i) ~= i
temp(i,1) = (datachar(i));
temp(i,2) = (datachar(result_4(i)));
char(temp(i,:))
end
end
```