

## ECE 637: Lab 5

Rahul Deshmukh  
[deshmuk5@purdue.edu](mailto:deshmuk5@purdue.edu)

March 3, 2021

### Section 2.1 Report

Scatter plots of  $W$ ,  $\tilde{X}$  and  $X$

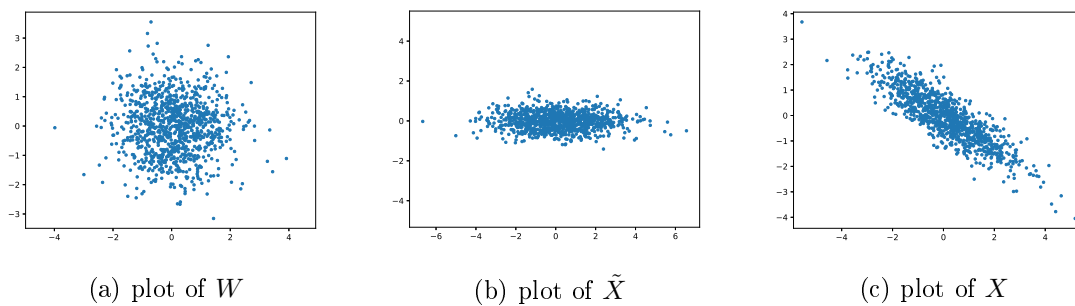


Figure 1: Scatter plots

For python code refer to Listing 7 at page 11.

## Section 2.2 Report

1. Theoretical value of  $R_x = \begin{bmatrix} 2 & -1.2 \\ -1.2 & 1 \end{bmatrix}$
2. Numerical listing of  $\hat{R}_x$  can be found in lines 5-7 of Listing 1 at page 2
3. Scatter plots of  $\tilde{X}_i$  and  $W_i$

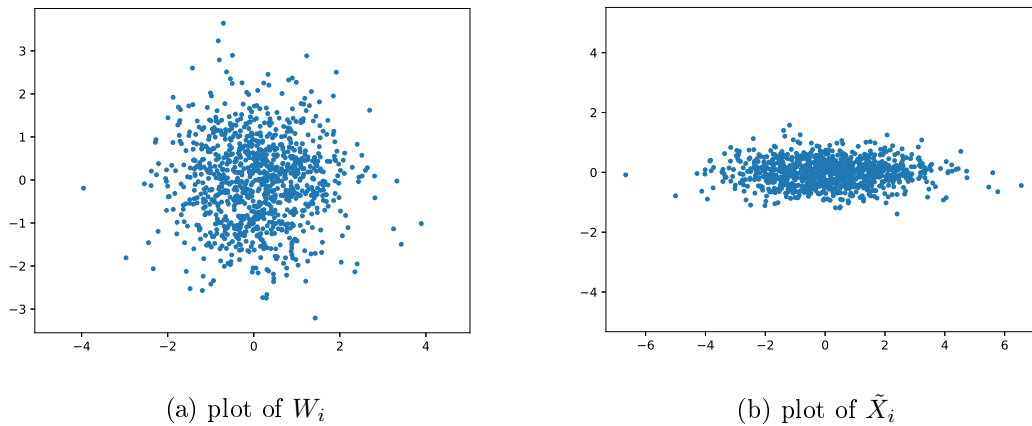


Figure 2: Scatter plots

4. Numerical listing of  $\hat{R}_W$  can be found in lines 13-15 of Listing 1 at page 2

Listing 1: output log for section2

```

1 Input covariance(R) :
2 [[ 2.  -1.2]
3  [-1.2  1.  ]]
4
5 Estimated covariance(R_hat) :
6 [[ 2.0016139  -1.23182429]
7  [-1.23182429  1.024182  ]]
8
9 Difference in covariances(R_hat-R) :
10 [[ 0.0016139  -0.03182429]
11  [-0.03182429  0.024182  ]]
12
13 Estimated covariance of R_W_hat:
14 [[ 1.00000000e+00 -1.32739165e-15]
15  [-1.32739165e-15  1.00000000e+00]]

```

## Section 4 Report

1. Eigen images:

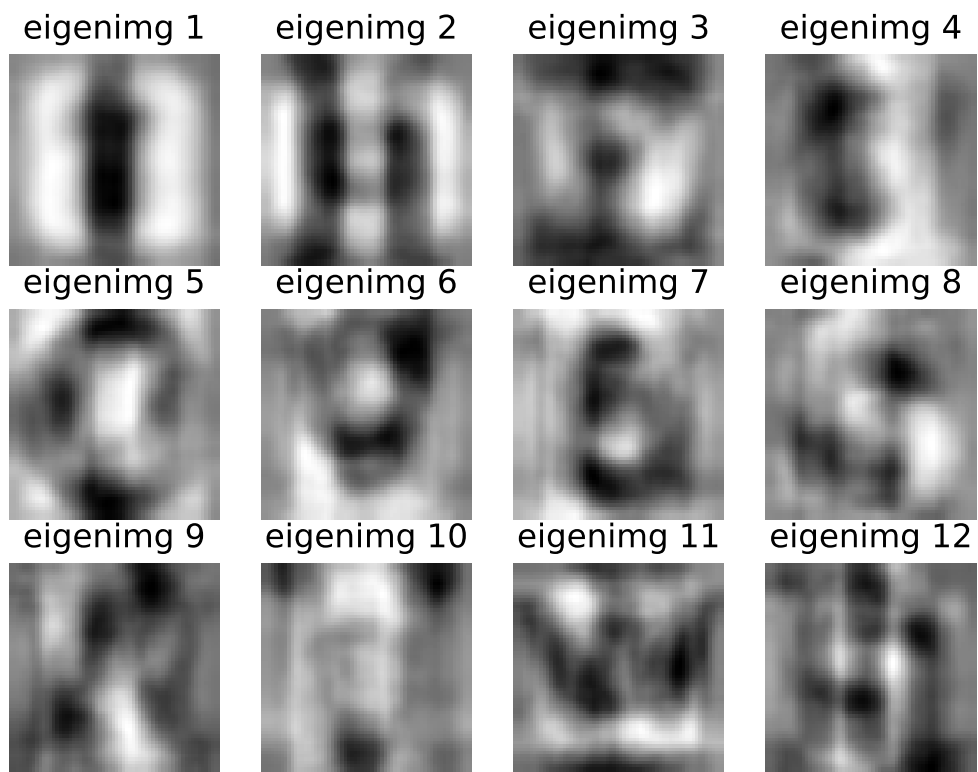
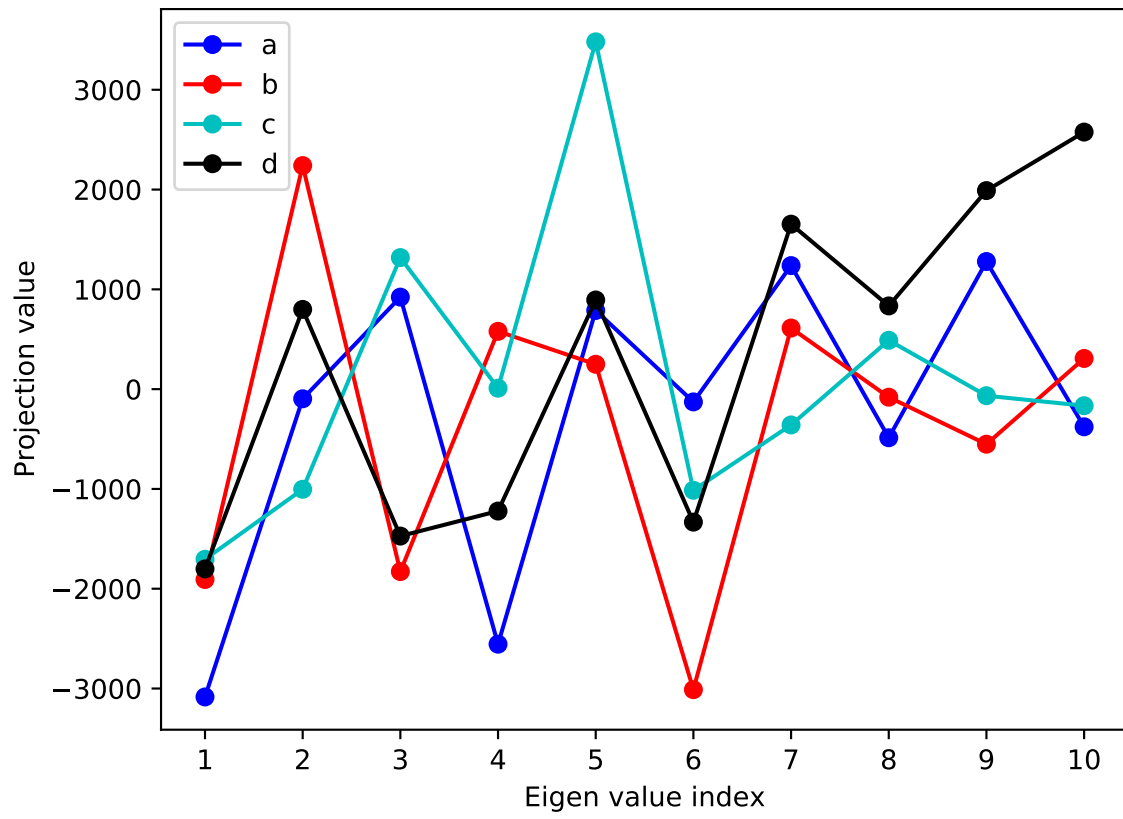
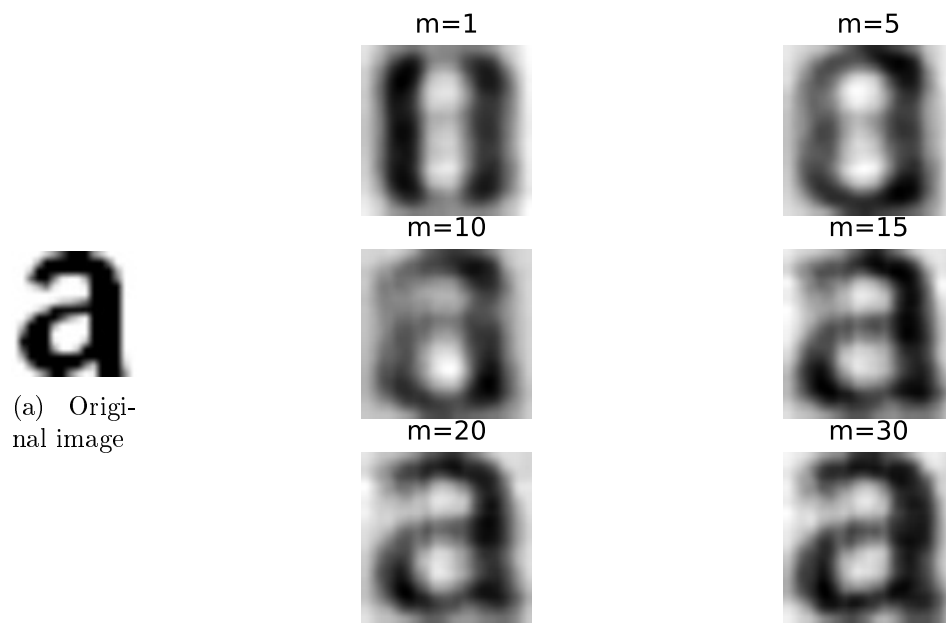


Figure 3: First 12 eigen images with title

2. Plot of projection coefficients vs eigen vector number:



3. Original image vs 6 resynthesized versions:



For python code refer to Listing [8](#) at page [12](#)

## Section 5.1 Report

The model has an accuracy of 69.231%. The mis-classified images are indicated with 'x' in the third column of Listing 2 at page 6

Listing 2: output log for classification

```
1 option-0
2 True    Predicted
3 a    a
4 b    b
5 c    c
6 d    a    x
7 e    e
8 f    f
9 g    g
10 h    h
11 i    i
12 j    y    x
13 k    k
14 l    i    x
15 m    m
16 n    v    x
17 o    o
18 p    e    x
19 q    a    x
20 r    r
21 s    s
22 t    t
23 u    a    x
24 v    v
25 w    w
26 x    x
27 y    v    x
28 z    z
29 Accuracy: 69.231
30
```

## Section 5.2 Report

The classification results for all modifications can be found at Listings 3-6.

1. From listings 3-6 we can observe that modifications #1,#2,#3 perform the best with same accuracy (92.308%) on testing data.
2. When we constrain the covariance ( $R_k$ ) we essentially are trying to obtain well-conditioned  $B_k$  which helps in computing the inverse ( $B_k$ ) for computing class scores. This solution is particularly helpful when we have limited training data which can lead to singular ML estimates of  $R_k$ . Here is a discussion on the effect of constraining for each modification:

- (a) Modification #1: Accuracy= 92.308%

$$B_k = \Lambda_k$$

We make the assumption that we have a pure quadratic classifier (ie only  $y_i^2$  terms). Also since we chose only the diagonal terms there are more chances that  $B_k$  is non-singular (ie  $[\Lambda_k]_{ii} \neq 0 \quad \forall \quad i \in [1, 2, \dots, n]$ ). This explains the improvement in accuracy compared to using  $R_k$ .

- (b) Modification #2: Accuracy= 92.308%

$$B_k = R_{wc} = \frac{1}{K} \sum_{k=1}^K R_k$$

We make the assumption that we have the same well-conditioned quadratic classifier  $B_k = R_{wc}$  for all  $K$  classes. Since we compute  $B_k$  using a sum, the estimate has higher chances of being non-singular. This explains the improvement in accuracy compared to using  $R_k$ .

- (c) Modification #3: Accuracy= 92.308%

$$B_k = \Lambda$$

We make the assumption that we have the same well-conditioned pure-quadratic classifier  $B_k$  for all  $K$  classes using only the diagonal terms of  $R_{wc}$  to construct  $\Lambda$ . This explains the improvement in accuracy compared to using  $R_k$ .

- (d) Modification #4: Accuracy= 88.462%

$$B_k = I$$

We make the assumption that we have a linear classifier as  $B_k = I$ . Even though  $B_k$  is non-singular in this case but we have reduced the complexity of our model. This explains the improvement in accuracy compared to using  $R_k$ . However the model does not have enough expressivity to get accuracy close to a quadratic classifier (Modification #1-#3).

Listing 3: output log for modification #1

```

1 option-1
2 True    Predicted
3 a    a
4 b    b
5 c    c
6 d    d
7 e    e
8 f    f
9 g    g
10 h   h
11 i   l   x
12 j   j
13 k   k
14 l   l
15 m   m
16 n   n
17 o   o
18 p   p
19 q   q
20 r   r
21 s   s
22 t   t
23 u   u
24 v   v
25 w   w
26 x   x
27 y   v   x
28 z   z
29 Accuracy: 92.308
30

```

Listing 4: output log for modification #2

```

1 option-2
2 True    Predicted
3 a    a
4 b    b
5 c    c
6 d    d
7 e    e
8 f    f
9 g    q   x
10 h   h
11 i   i
12 j   j
13 k   k
14 l   l
15 m   m
16 n   n
17 o   o
18 p   p
19 q   q
20 r   r
21 s   s
22 t   t
23 u   u
24 v   v
25 w   w

```



```

26 x x
27 y v x
28 z z
29 Accuracy: 92.308
30

```

Listing 5: output log for modification #3

```

1 option-3
2 True Predicted
3 a a
4 b b
5 c c
6 d d
7 e e
8 f t x
9 g g
10 h h
11 i i
12 j j
13 k k
14 l l
15 m m
16 n n
17 o o
18 p p
19 q q
20 r r
21 s s
22 t t
23 u u
24 v v
25 w w
26 x x
27 y v x
28 z z
29 Accuracy: 92.308
30

```

Listing 6: output log for modification #4

```

1 option-4
2 True Predicted
3 a a
4 b b
5 c c
6 d d
7 e e
8 f t x
9 g q x
10 h h
11 i i
12 j j
13 k k
14 l l
15 m m
16 n n
17 o o
18 p p

```

19	q	q	
20	r	r	
21	s	s	
22	t	t	
23	u	u	
24	v	v	
25	w	w	
26	x	x	
27	y	v	x
28	z	z	
29	Accuracy: 88.462		
30			

## Appendix

Got to [git repo](#) for complete code.

Listing 7: Python code for section 2

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Mon Mar  1 17:46:46 2021
5
6  @author: rahul
7  course: ece637 DIP-1
8  lab5: 2.1 and 2.2
9  """
10
11 import sys
12 import numpy as np
13 import matplotlib.pyplot as plt
14
15 np.random.seed(seed=637)
16 Rx = np.array([[2.0, -1.2], [-1.2, 1.0]])
17
18 def plot_pts(W, name):
19     plt.figure()
20     plt.plot(W[0, :], W[1, :], '.')
21     plt.axis('equal')
22     plt.savefig(name+'.eps', format='eps')
23     plt.close()
24
25 def estimate_mean_cov(X):
26     "unbiased estimate of mean and cov (MLE)"
27     dim, num_pts = X.shape
28     mu_hat = X.sum(axis=1)/num_pts
29     R_hat = np.zeros((dim, dim))
30     for i in range(num_pts): R_hat += np.outer(X[:, i] - mu_hat, X[:, i] - mu_hat)
31     R_hat /= num_pts - 1.0
32     return mu_hat, R_hat
33
34
35 def main(num_pts):
36     #Section 2.1
37     size, _ = Rx.shape
38     I = np.eye(size)
39     mu = np.zeros(size)
40     Lam, E = np.linalg.eig(Rx)
41     W = np.random.multivariate_normal(mu, I, num_pts).T
42     X_scaled = np.dot(np.diag(np.sqrt(Lam)), W)
43     X = np.dot(E, X_scaled)
44     plot_pts(W, 'W')
45     plot_pts(X_scaled, 'X_scaled')
46     plot_pts(X, 'X')
47
48     #Section 2.2
49     mu_hat, R_hat = estimate_mean_cov(X)
50     print('Input covariance(R):')
51     print(Rx); print()
52     print('Estimated covariance(R_hat):')
53     print(R_hat); print()
```

```

54     print( 'Difference in covariances(R_hat-R): ' )
55     print( R_hat-Rx ); print ()
56
57     Lam_hat,E_hat = np.linalg.eig(R_hat)
58     X_scaled_hat = np.dot(E_hat.T,X)
59     W_hat = np.dot(np.diag(np.sqrt(1/Lam_hat)), X_scaled_hat)
60     mu_W_hat, R_W_hat = estimate_mean_cov(W_hat)
61     print( 'Estimated covariance of R_W_hat: ' )
62     print(R_W_hat); print ()
63
64     plot_pts(X_scaled_hat, 'X_scaled_hat')
65     plot_pts(W_hat, 'W_hat')
66
67 if __name__=="__main__":
68     num_pts = int( sys.argv[1])
69     main(num_pts)

```

Listing 8: Python code for PCA

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Mon Mar  1 20:19:53 2021
5
6  @author: rahul
7  course: ece637 DIP-1
8  lab: 5 section 4
9  """
10
11 import numpy as np
12 import matplotlib.pyplot as plt
13 from read_data import read_data
14
15 Ht=64
16 Wd=64
17 m = [1,5,10,15,20,30]
18
19 def display_eigen_images(U,Lam):
20     fig, axs = plt.subplots(3, 4)
21     for k in range(12):
22         img=np.reshape(U[:,k],(Wd,Ht))
23         axs[k//4,k%4].imshow(img,cmap=plt.cm.gray, interpolation='none')
24         axs[k//4,k%4].set_title( 'eigenimg '+str(k+1))
25         axs[k//4,k%4].axis( 'off ')
26     plt.savefig( 'eigen_images.pdf')
27     plt.close()
28
29 def plot_projections(Y):
30     c=10
31     x = np.arange(1,c+1,1)
32     plt.figure()
33     plt.xticks(x)
34     plt.plot(x,Y[:,0], '-ob', label='a')
35     plt.plot(x,Y[:,1], '-or', label='b')
36     plt.plot(x,Y[:,2], '-oc', label='c')
37     plt.plot(x,Y[:,3], '-ok', label='d')
38     plt.xlabel( 'Eigen value index')
39     plt.ylabel( 'Projection value')
40     plt.legend()

```

```

41     plt.savefig('projections.pdf')
42
43
44 def main():
45     X = read_data()
46     p, n = X.shape
47     mu_hat = np.sum(X, axis=1)/n
48     X = (X.T - mu_hat).T
49     Z = X/np.sqrt(n)
50     U, S, Vt = np.linalg.svd(Z, full_matrices=False)
51     Lam = S**2
52     display_eigen_images(U, Lam)
53
54     #projection of images
55     Y = np.dot(U.T, X[:, :4])
56     plot_projections(Y)
57
58     #reconstruction
59     recons = np.zeros((p, len(m)))
60     for i, im in enumerate(m):
61         recons[:, i] = np.dot(U[:, :im], Y[:, im, 0])
62     recons = (recons.T + mu_hat).T
63     fig, axs=plt.subplots(3, 2)
64     for k in range(6):
65         img=np.reshape(recons[:, k], (Wd, Ht))
66         axs[k//2, k%2].imshow(img, cmap=plt.cm.gray, interpolation='none')
67         axs[k//2, k%2].set_title('m='+str(m[k]))
68         axs[k//2, k%2].axis('off')
69     plt.savefig('reconstruction.pdf')
70     plt.close()
71
72 if __name__=="__main__":
73     main()

```

Listing 9: Python code for image classification using PCA

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Tue Mar  2 00:48:48 2021
5
6  @author: rahul
7  course: ece637 DIP-1
8  lab-5 section 5
9  """
10 import glob, os, sys
11 import numpy as np
12 from PIL import Image
13 import matplotlib.pyplot as plt
14 from read_data import read_data
15
16 Ht=64; Wd=64;
17 eig_size=10
18 test_dir='../test_data/'
19
20 def option014(Y, option):
21     params = []
22     for k in range(26):
23         samples = Y[:, k::26]

```

```

24     _, num_samples = samples.shape
25     mu = samples.sum(axis=1)/(1.0*num_samples)
26     if option!=4:
27         cov = np.zeros(( eig_size , eig_size ))
28         for i in range(num_samples):
29             cov += np.outer(samples[:,i]-mu,samples[:,i]-mu)
30         cov /= num_samples-1.0
31         if option==1:
32             cov = np.diag(np.diagonal(cov)) #only diagonal elements
33     else:
34         cov = np.eye(eig_size)
35     dic = { 'mean':mu, 'cov':cov }
36     params.append(dic)
37     return params
38
39 def option23(Y,option):
40     params = []
41     for k in range(26):
42         samples = Y[:,k::26]
43         _, num_samples = samples.shape
44         mu = samples.sum(axis=1)/(1.0*num_samples)
45         cov = np.zeros(( eig_size , eig_size ))
46         for i in range(num_samples):
47             cov += np.outer(samples[:,i]-mu,samples[:,i]-mu)
48         cov /= num_samples-1.0
49         dic = { 'mean':mu, 'cov':cov }
50         params.append(dic)
51     R_wc = np.zeros(( eig_size , eig_size ))
52     for k in range(26): R_wc += params[k][ 'cov ' ]
53     R_wc /= 26
54     if option==2:
55         for k in range(26): params[k][ 'cov ' ] = R_wc
56     elif option==3:
57         Lam = np.diag(np.diagonal(R_wc))
58         for k in range(26): params[k][ 'cov ' ] = Lam
59     return params
60
61 def training(option):
62     X = read_data()
63     p,n = X.shape
64     mu_x = np.sum(X,axis=1)/n
65     X = (X.T - mu_x).T
66     Z = X/np.sqrt(n)
67     U,S,Vt = np.linalg.svd(Z,full_matrices=False)
68     A = U[:, : eig_size ]
69     Y = np.dot(A.T,X)
70
71     if (np.any(np.array([0,1,4])==option)): params = option014(Y,option)
72     elif (np.any(np.array([2,3])==option)): params = option23(Y,option)
73
74     return params, A, mu_x
75
76 def testing(params, A, mu_x):
77     #read images
78     filenames = glob.glob(test_dir+'*/*.tif')
79     true_labels = []
80     for i,f in enumerate(filenames):
81         class_label = os.path.basename(f).split('.')[0]
82         true_labels.append(class_label)

```

```

83     #classify data
84     predicted_labels = []
85     for i, f in enumerate(filenamees):
86         im = Image.open(f)
87         img = np.array(im)
88         x = np.reshape(img, Ht*Wd) - mu_x
89         y = np.dot(A.T, x)
90         class_scores = np.zeros(26)
91         for k in range(26):
92             mu_k = params[k][ 'mean' ]
93             cov_k = params[k][ 'cov' ]
94             class_scores[k] = (np.dot((y-mu_k), np.dot(np.linalg.inv(cov_k), (y-mu_k)))
95                               +
96                               np.log(np.abs(np.linalg.det(cov_k))))
97             label = np.argmin(class_scores)
98             char = chr(ord('a') + label)
99             predicted_labels.append(char)
100     return true_labels, predicted_labels
101
102 if __name__ == "__main__":
103     option = int(sys.argv[1])
104     params, A, mu_x = training(option)
105     true_labels, predicted_labels = testing(params, A, mu_x)
106     # print results
107     print('True\t Predicted\t')
108     count = 0.0
109     for i in range(len(true_labels)):
110         flag_correct = (true_labels[i]==predicted_labels[i])
111         if flag_correct: count+=1
112         print("%s\t %s"%(true_labels[i], predicted_labels[i]), end='\t')
113         if not flag_correct:
114             print('x', end='')
115         print('')
116     acc = (count/len(true_labels))*100
117     print('Accuracy: %0.3f'%(acc))

```

Listing 10: Python code for data reading utility

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Mon Feb 15 19:32:39 2021
5  ECE637
6  Prof. Charles A. Bouman
7  Image Processing Laboratory: Eigenimages and Principal Component Analysis
8
9  Description:
10
11  This is a Matlab script that reads in a set of training images into
12  the Matlab workspace. The images are sets of English letters written
13  in various fonts. Each image is reshaped and placed into a column
14  of a data matrix, "X".
15  @author: Wenrui Li
16  """
17
18  import sys
19  import numpy as np
20  from PIL import Image
21  import matplotlib.pyplot as plt

```

```

22
23 # The following are strings used to assemble the data file names
24 datadir='.././training_data/' # directory where the data files reside
25 dataset=['arial','bookman_old_style','century','comic_sans_ms','courier_new',
26 'fixed_sys','georgia','microsoft_sans_serif','palatino_linotype',
27 'shruti','tahoma','times_new_roman']
28 datachar='abcdefghijklmnopqrstuvwxyz'
29
30 def read_data():
31     """
32     Read in all these training images into columns of a single matrix X.
33
34     Returns:
35         X: Image column matrix.
36
37     """
38     Rows=64 # all images are 64x64
39     Cols=64
40     n=len(dataset)*len(datachar) # total number of images
41     p=Rows*Cols # number of pixels
42
43     X=np.zeros((p,n)) # images arranged in columns of X
44     k=0
45     for dset in dataset:
46         for ch in datachar:
47             fname='/'.join([datadir,dset,ch])+'.tif'
48             im=Image.open(fname)
49             img = np.array(im)
50             X[:,k]=np.reshape(img,(1,p))
51             k+=1
52     return X
53
54 # display samples of the training data
55 def display_samples(X,ch):
56     """
57     Display samples.
58
59     Args:
60     X (ndarray) : Image column matrix.
61     ch (char) : A char 'a'~'z'.
62
63     Returns:
64
65     """
66     ind = ord(ch)-ord('a')
67     fig, axs = plt.subplots(3, 4)
68     for k in range(len(dataset)):
69         img=np.reshape(X[:,26*(k-1)+ind],(64,64))
70
71         axs[k//4,k%4].imshow(img,cmap=plt.cm.gray, interpolation='none')
72         axs[k//4,k%4].set_title(dataset[k])
73
74     plt.show()
75
76 if __name__ == "__main__":
77     ch = sys.argv[1]
78     X = read_data()
79     display_samples(X,ch)

```



Listing 11: Bash code for running python code

```

1  #!/ bin/bash
2
3  #Section 2.1 and 2.2
4  python3 ex2.py 1000 | tee sec2.log
5  mv ./*.eps output/section_2/
6  mv ./sec2.log output/section_2/
7  echo 'section 2 done'
8
9  #section 4
10 python3 ex4.py
11 mv ./*.pdf output/section4/
12 echo 'section 4 done'
13
14 #section 5
15 for opt in {0..4}
16 do
17 {
18     echo option-"$opt";
19     python3 ex5.py $opt;
20     echo "_____";
21 } | tee ex5_opt_"$opt".log
22 done
23 mv ./*.log output/section5/
24 echo 'section 5 done'

```