“Statistical Pattern Recognition”

Shervin Halat

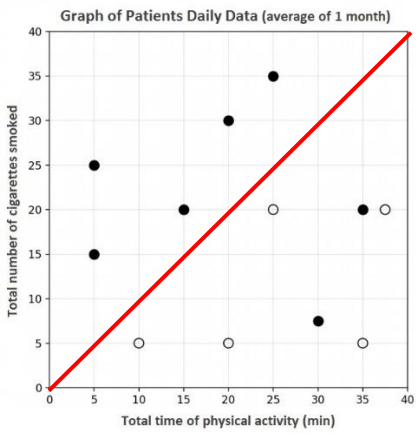
98131018

Homework 4

1.

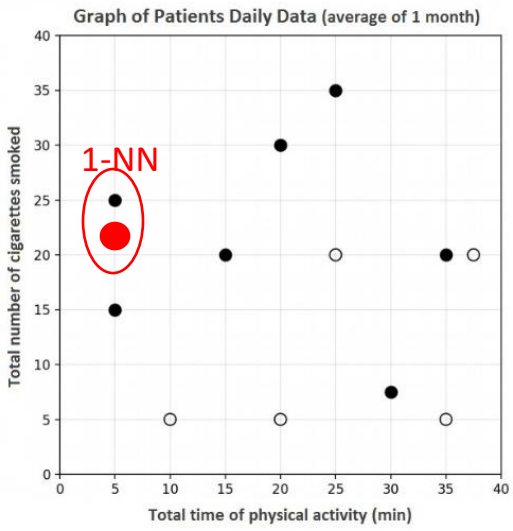
a.

As the values of first principal component vector are the same, the line is diagonal.



b.

1-NN classifier classifies the mentioned data as **'suffering from lung cancer'** according to the graph below:



c.

This time the classifier would classify the patient as **'not suffering from lung cancer'**. Considering the labels are projected onto the red line, distance of (5,21) to point (20,5) is lower than (5,25) therefore (20,5) will be considered as closest neighbor of (5,21).

d.

No, they are not. Since, PCA method does not consider labels and it is mainly designed for accurate data representation but not for data classification. To put it simply, PCA doesn't take any assumptions related to data's label, therefore, data points are not separated arranged based on their labels after implementing this method, so, it is not wondering to see the 1NN method leads to different outputs.

e and f.

gene1:

mean = 1.3 variance = 9.21

gene2:

mean = 6 variance = 0.8

gene3:

mean = 7.6 variance = 0.84

g.

$$\tilde{X} = X - \bar{X}$$

We have: $\bar{X} = \begin{matrix} 1.3 \\ 6 \\ 7.6 \end{matrix}$

$$\tilde{X} = \begin{bmatrix} -1 & 6 & -1 & 2 & 0 & -1 & -1 & -2 & 5 & 6 \\ 6 & 6 & 5 & 6 & 8 & 5 & 6 & 5 & 6 & 7 \\ 7 & 9 & 7 & 7 & 8 & 9 & 7 & 8 & 8 & 6 \end{bmatrix}$$

h.

$$\text{gene-covariance}(\tilde{X}) = C = \begin{matrix} 10.23 & 1.11 & -0.09 \\ 1.11 & 0.88 & -0.22 \\ -0.09 & -0.22 & 0.93 \end{matrix}$$

i.

now we calculate eigenvalues of C:

eigenvalues of C.8

$$C = \begin{bmatrix} 10.23 & 1.11 & -0.09 \\ 1.11 & 0.88 & -0.22 \\ -0.09 & -0.22 & 0.93 \end{bmatrix} \qquad |C - \lambda I| = 0 \rightsquigarrow \lambda = \text{eigenvalues}$$

$$\begin{vmatrix} 10.23-\lambda & 1.11 & -0.09 \\ 1.11 & 0.88-\lambda & 0.22 \\ -0.09 & -0.22 & 0.93-\lambda \end{vmatrix} = 0 \Longrightarrow \lambda_1 = 10.365$$

$$\lambda_2 = 1.072$$

$$\lambda_3 = 0.617$$

Three eigenvalues

j.

According to the eigenvalues obtained in previous part, besides, knowing that eigenvalues are variances of the corresponding principal directions, the first principal component of C would be maximum value of eigenvalues which is 10.365.

Now calculating sum of all eigenvalues results in:

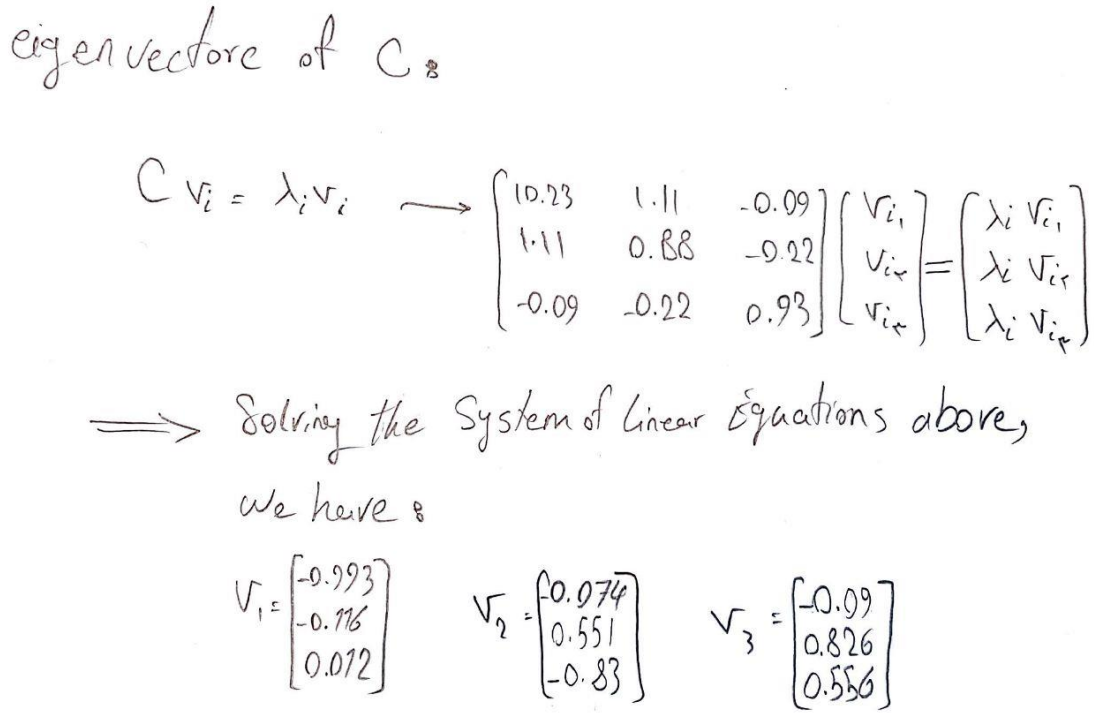
Total variance = 12.054

First Principal Component variance = 10.365

Therefore, the First Principal Component accounts for approximately **86 percent** of total variance of the data.

k.

Considering eigenvalues obtained in part 'i', the corresponding eigenvectors are as follows:



I.

Let ***e1***, ***e2***, …, ***ek*** be the columns of matrix ***E = [e1, e2, …, ek]*** where 'e_i' is eigenvectors.

The transformed data points (Y) would be obtained by following equation:

$$Y = E^T * \tilde{X}$$

We have:

$$E = \begin{matrix} -0.993 & -0.074 & -0.09 \\ -0.116 & 0.551 & 0.826 \\ 0.012 & -0.83 & 0.556 \end{matrix}$$

Using mentioned equation, Y approximately becomes:

$$Y = \begin{matrix} 2.27 & -4.65 & 2.39 & -0.70 & 1.06 & 2.41 & 2.27 & 3.39 & -3.66 & -4.80 \\ 0.66 & -1.51 & 0.11 & 0.44 & 0.86 & -1.54 & 0.66 & -0.63 & -0.60 & 1.53 \\ -0.12 & 0.35 & -0.95 & -0.39 & 1.99 & 0.15 & -0.12 & -0.30 & -0.11 & -0.48 \end{matrix}$$

2.

a.

Firstly, we compute within the class scatter matrix:

$$Sw = S1 + S2$$

Then, using following equation, we compute the Linear Discriminant:

$$V = S_w^{-1} (\mu_1 - \mu_2)$$

Therefore, 'V' becomes:

$$V = \begin{bmatrix} \mathbf{-1.932} \\ \mathbf{0.252} \end{bmatrix}$$

b.

Denoting transformed datapoint X by X' we have:

$$\mathbf{X' = \frac{V.X}{|V|}}$$

For X = $[1.27 \quad 4.45]^T$, X' becomes:

$$\mathbf{X' = -0.6838}$$

Now to classify X', we have to transform both means ($\bar{X}_1$ and $\bar{X}_2$) and assign the data point to the label of closer mean.

$$\mu_1' = V.\mu_1 / |v|$$

$$\mu_2' = V.\mu_2 / |v|$$

Therefore:

$$\mu_1' = -1.281$$

$\mu_2' = -4.417$

It is obvious that X' is closer to μ_1' hence, it is assigned to class 2.

c.

Some assumptions have been suggested for implementing discriminant analysis (or specifically for Linear Discriminant Analysis), although, discriminant analysis is relatively robust to slight violations of these assumptions. Some of the main assumptions are as follows:

One of the fundamental assumptions of the LDA method is that the variables are distributed normally. Since there is no information about the distribution of the classes, it seems acceptable to have such an assumption for our problem here!

Besides, it is assumed that the variables are continuous and independent. In our situation, the variables seem to be continuous but as it can be figured out from covariance matrixes, the variables are not independent but as it is mentioned above, LDA is robust to some slight violations. For example, we know that normally distributed classes may be separated linearly if they have the same covariance matrixes. Hence, as the covariance matrixes are slightly the same, the mentioned assumption may be ignored.

What's more, the covariance matrixes of classes should be the same which was discussed in previous paragraph.

Also, it should be noted that the means of classes should not be equal and within classes scatter matrix should be full rank

in order to be inversible. The recent mentioned criteria are not violated in our case.

To summarize, it is valid to consider two classes of our problem normally distributed with approximately same covariances. Therefore, it is logical to implement LDA in order to separate classes here.

d.

As it was noted in part 'c', the covariance matrixes should be the same or just slightly violated. Here, since S_2' remarkably differs from S_1 it is not reasonable to apply LDA. One solution to overcome this situation may be normalization of data in such a way that S_2' becomes close to S_1.

e.

In order to make two covariances as similar as possible, is can be suggested to :

1) divide first variable of S_2' by $\sqrt{40}$
2) divide second variable of S_2' by $\sqrt{20}$

Applying '1' and '2' results in:

$$S_2' = \begin{bmatrix} 41 & 40 \\ 40 & 41 \end{bmatrix} \rightarrow \begin{bmatrix} 1.02 & 1.44 \\ 1.44 & 2.04 \end{bmatrix}$$

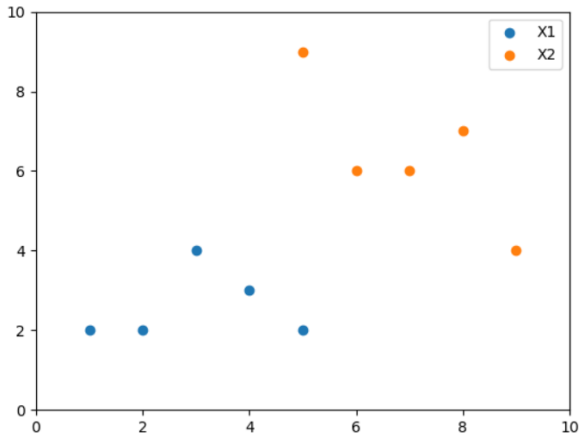
But it should be noted that mean also changes this way:

$$\bar{X}'_2 = \begin{bmatrix} 5.5 \\ 8.01 \end{bmatrix} \rightarrow \begin{bmatrix} 0.86 \\ 1.79 \end{bmatrix}$$

Now to classify X, exactly the same way of part 'b' should be done.

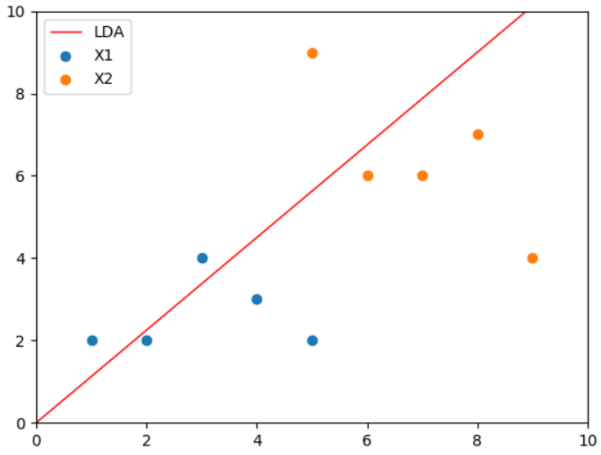
f.

(related script: **p2.f**):



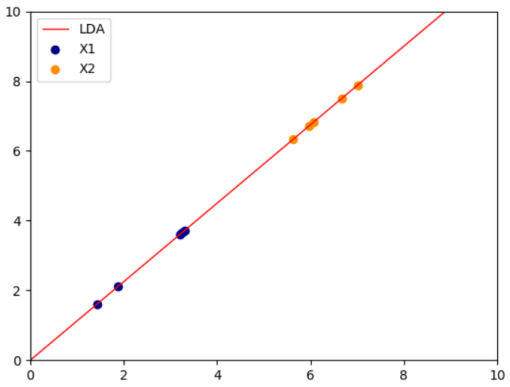
g.

(related script: **p2.f**):



h.

(related script: **p2.f**):



i.

Discriminability is obtained by equation below:

$$J(v) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

Therefore, for our case discriminability becomes:

```
projected X1 mean = -3.9363557450672744
projected X2 mean = -9.43396651138486
pojected X1 scatter = 5.755586206896551
projected X2 scatter = 2.2797241379310353
discriminability: 0.6841815101586531
```

Hence,

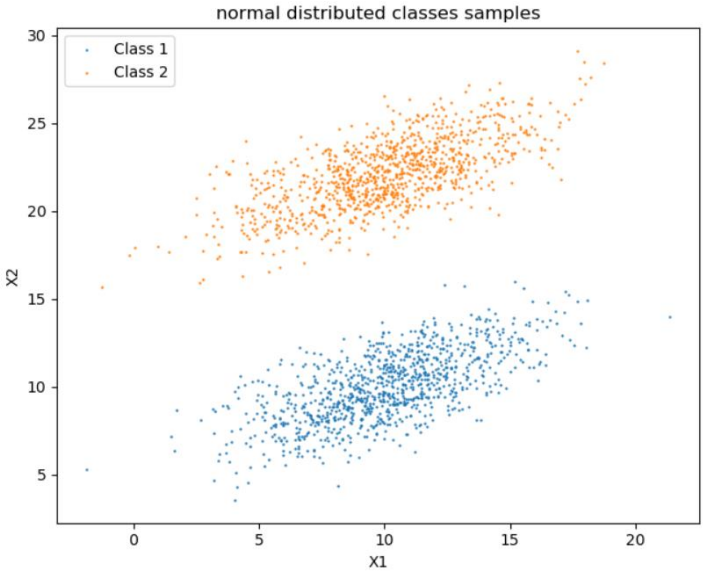
Discriminability = 0.68

3.

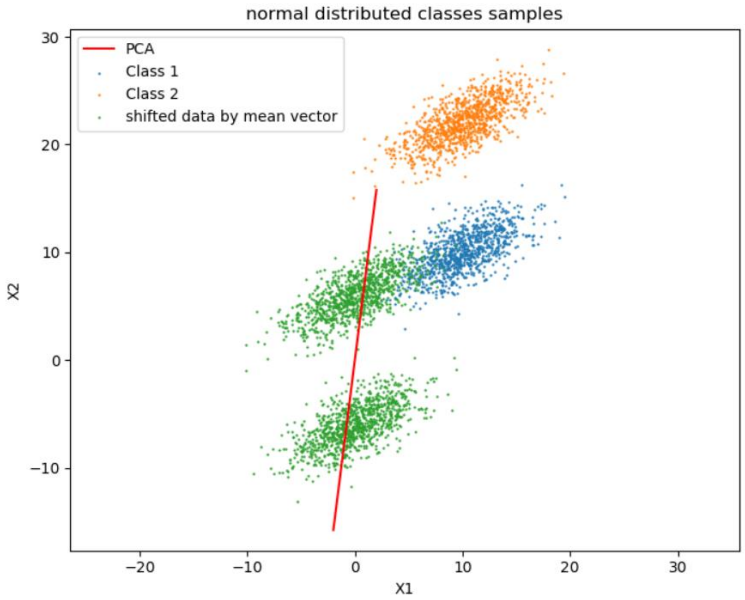
a.

Assuming each class follows a normal distribution,

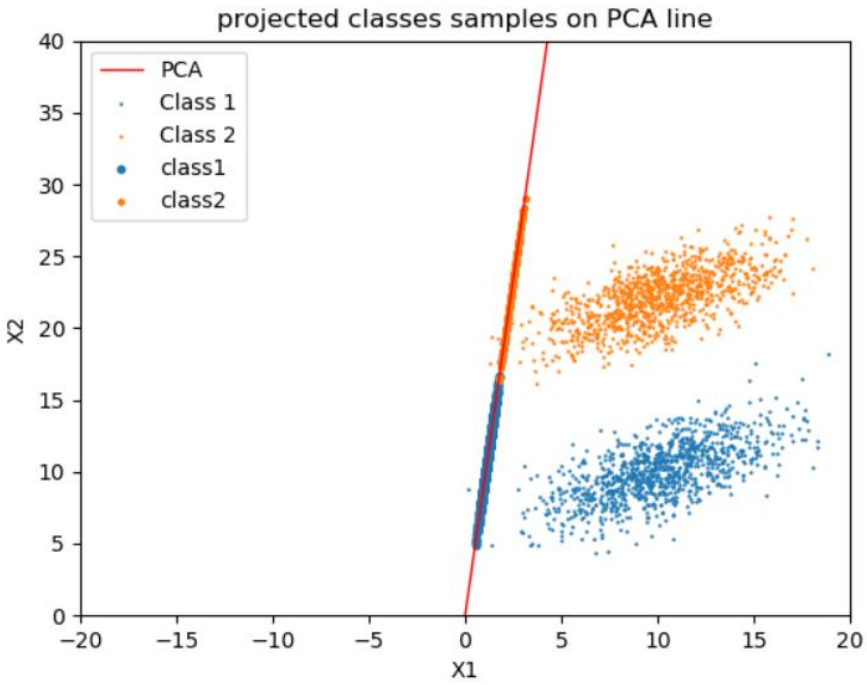
(related script: **p3.b**):



b. **PCA line (red)** (related script: **p3.b**):



c. (related script: **p3.c**):



d.

Despite PCA is not introduced in order to classify different classes but Dimensionality Reduction, it was expected that projected data points on PCA line become mixed and inseparable but, exclusively, in this case both classes became slightly distinguishable and separate with negligible overlap.

e.

Related script is **p3.e**:

Using function below:

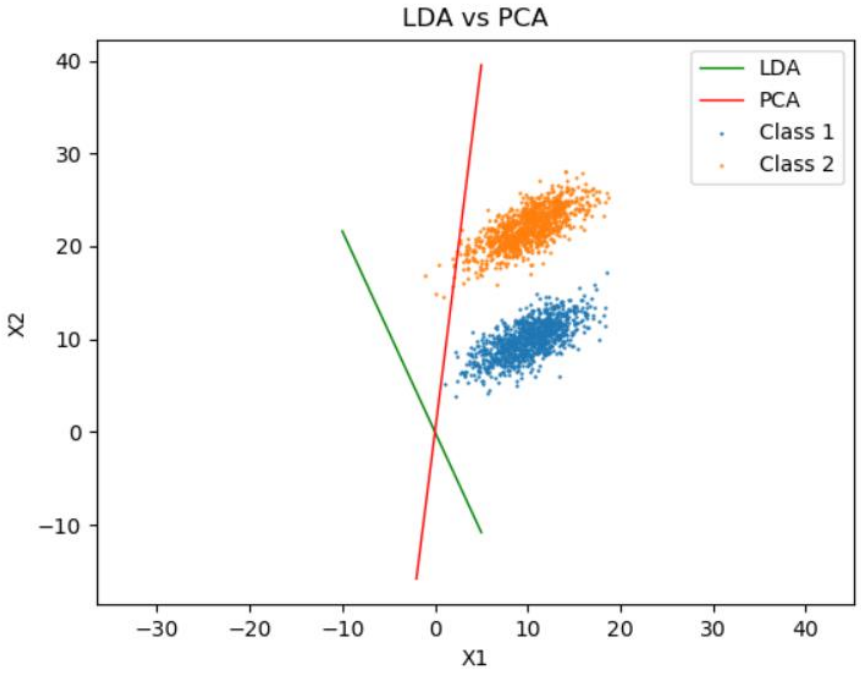
$$J(e_1,\ldots,e_k,\alpha_{11},\ldots\alpha_{nk}) = \sum_{j=1}^{n}\left\| x_j - \sum_{i=1}^{k}\alpha_{ji}e_i \right\|^2$$

The total error (reconstruction error) becomes:

```
======== RESTART: C:/Users/sherw/OneDrive
Reconstruction Error would be: 16480.55
>>>
```

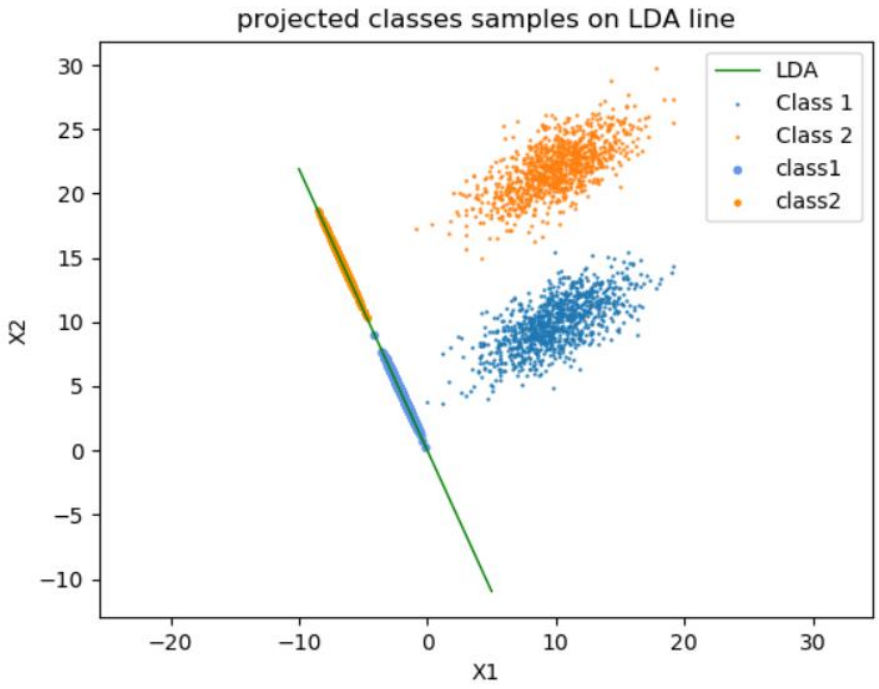
f.

LDA line (green) and PCA line (red) (related script: **p3.f**):



g.

Projected samples on LDA line (related script: **p3.g**):

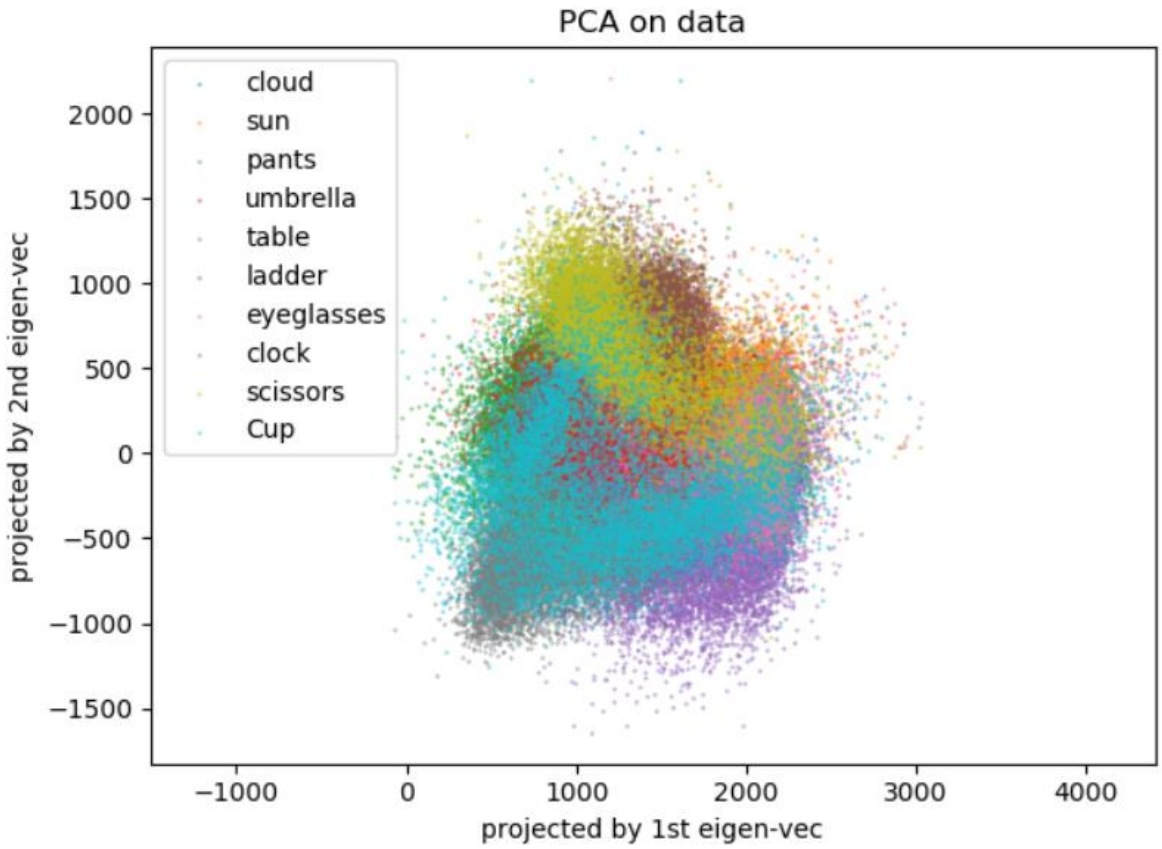


h.

As it can be figured out from figure above, both classes are approximately separated perfectly with the no overlap between classes' datapoints. Since LDA is a method of classification with dimensionality reduction with respect to class labels, as against PCA, it was expected to project data points in a separable way especially when each class is assumed to be normally distributed with unique mean value.

i.

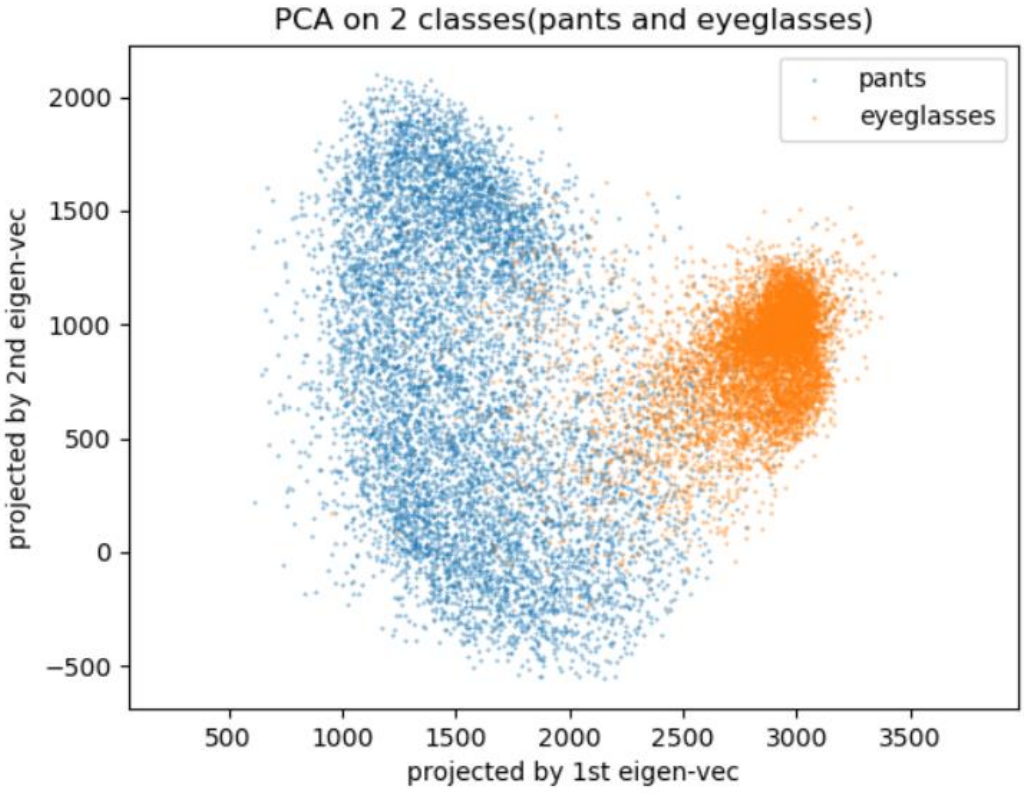
(Related script is: **p6.i**)

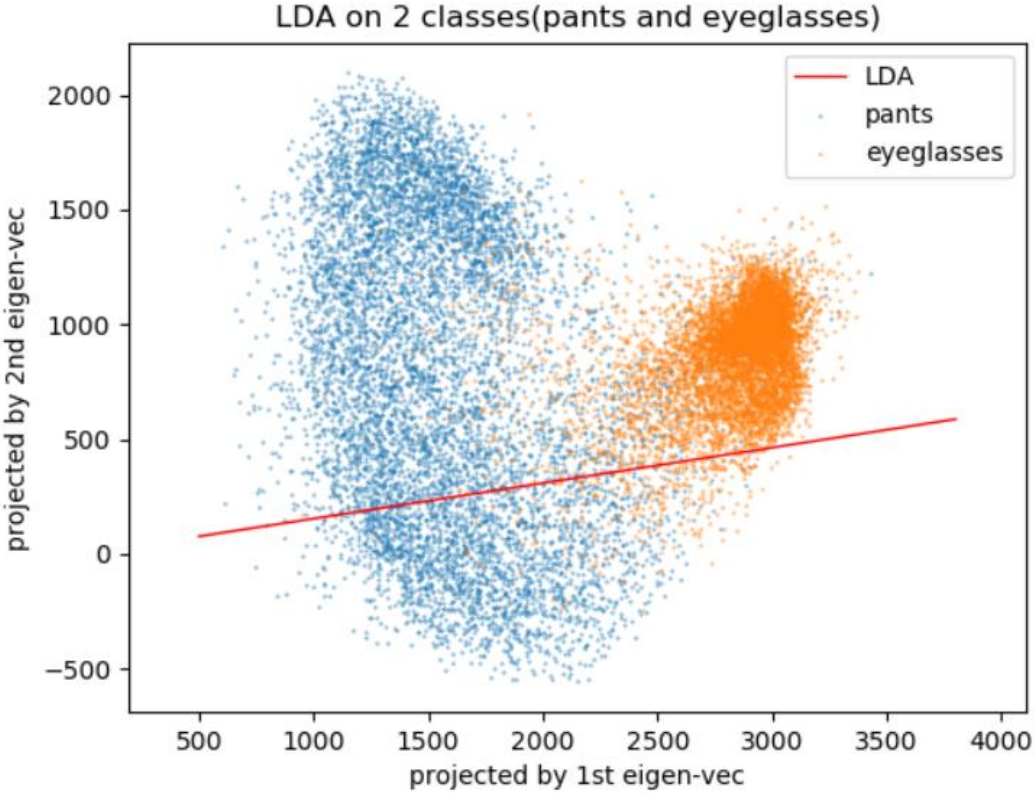


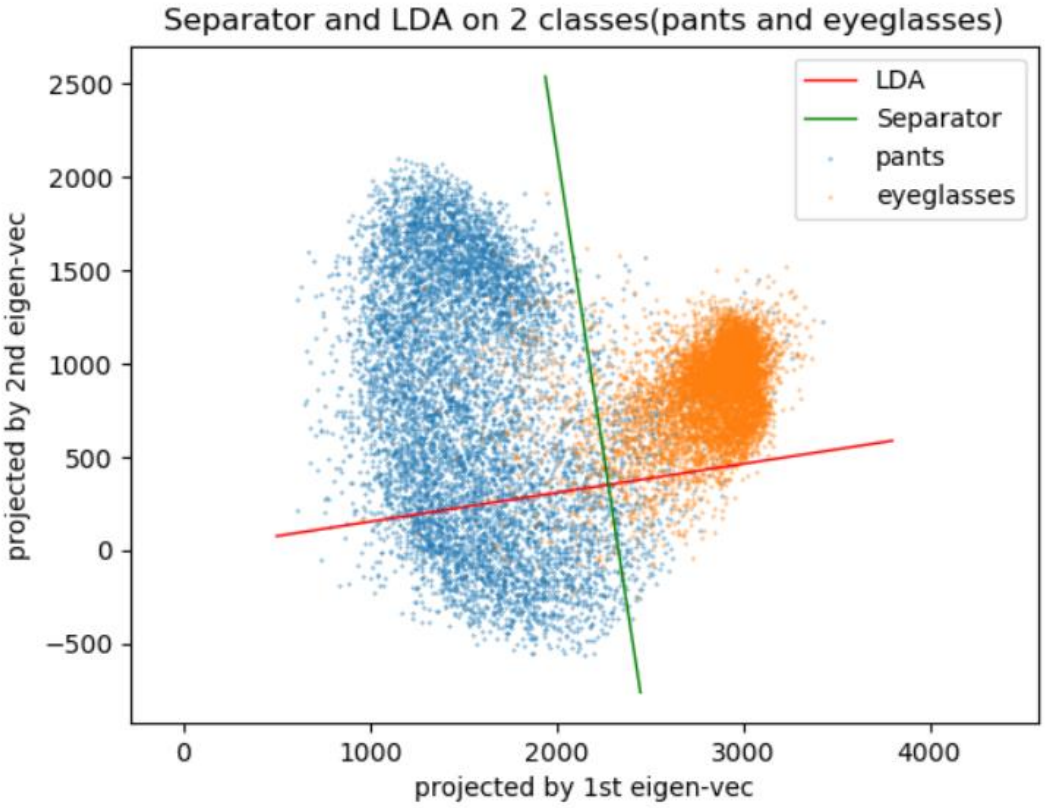
j & k.

(Related script is: **p3.j & p3.j.prime**)

Here we ignore all classes except classes 2 and 6 then apply PCA and LDA on results obtained by PCA:


PCA on 2 classes(pants and eyeglasses)
pants
eyeglasses
projected by 2nd eigen-vec
projected by 1st eigen-vec
2000
1500
1000
500
0
−500
500
1000
1500
2000
2500
3000
3500


LDA on 2 classes(pants and eyeglasses)
LDA
pants
eyeglasses
projected by 2nd eigen-vec
projected by 1st eigen-vec
2000
1500
1000
500
0
−500
500
1000
1500
2000
2500
3000
3500
4000



Confusion Matrix for Training Sets:

```
confusion matrix for training set would be:
 [[7513.  502.]
 [ 315. 7723.]]
accuracy for training set would be:
 94.91
```

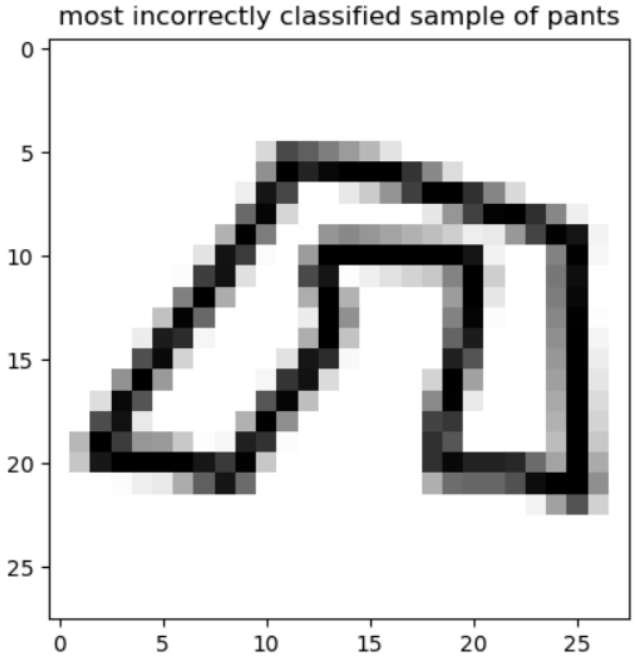
Confusion Matrix for Test Sets:

```
confusion matrix for test set would be:
 [[1854.  131.]
 [  98. 1864.]]
accuracy for test set would be:
 94.20
```

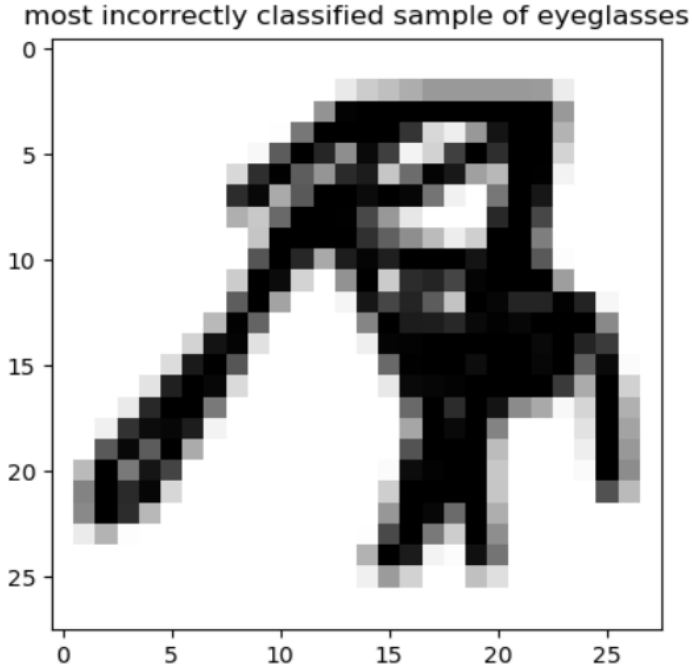
l.

(Related script is: **p3.l**)

The most incorrectly classified test sample of pants:



The most incorrectly classified test sample of eyeglasses:



m.

???

n.

???

4.

???

5.

a.

Line 41 of classify_grid(): v = v + **((alpha(j) .* SV(j , :)) * xy);**

b.

Line 47 of trainsvm(): H(i,j) = **(xi * xj').* (yi * yj');**

c.

Line 77 of trainsvm:

[x,fval] = quadprog**(H, f, [], [], Aeq, beq, lb, ub, x0, options);**

d.

SV: each row of SV matrix represents each support vector.

alpha: each number of alpha vector represents the coefficient of each support vector.

b: 'b' represents the offset.

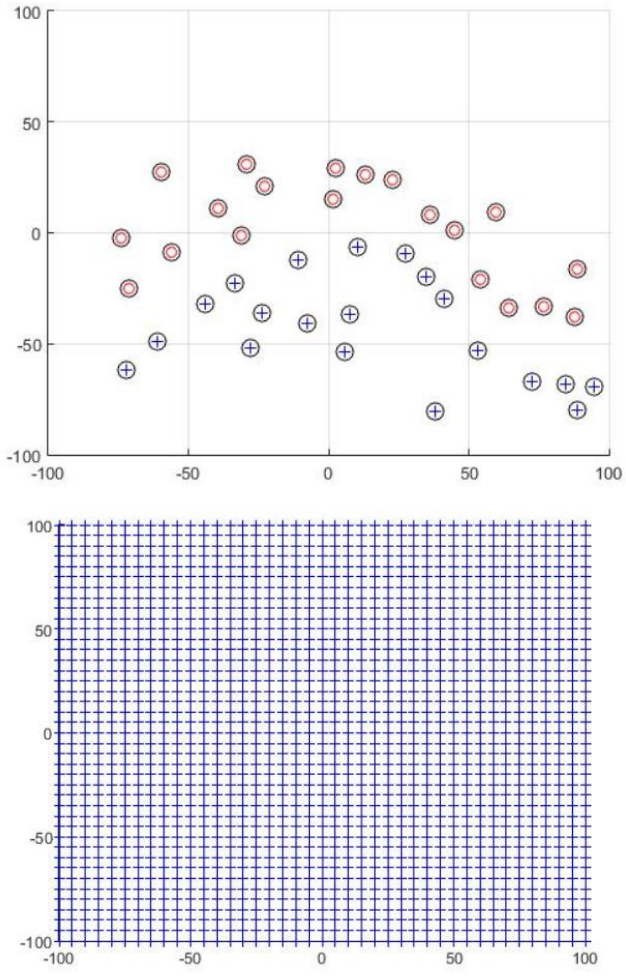
e.

The circled points are called support vectors which are the samples closest to the separating line and determine the separating line properties or in other words, the separating line is based on them.

f.

As it is mentioned in ‘trainsvm’ notepad, parameter ‘C’ is represents the cost of misclassifying training point in non-separable case. C = inf means separable case. In other words ‘inf’ probably represents ‘infinity’ with the meaning of that all datapoints should be separated in their right class.

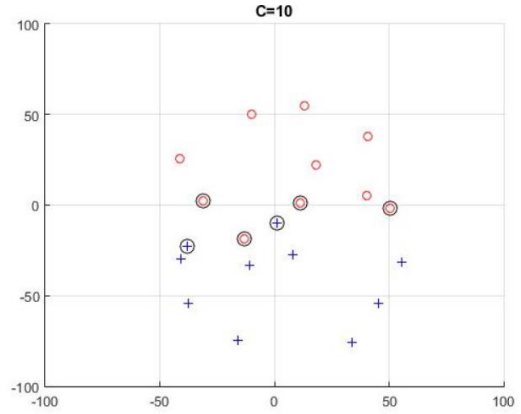
g.

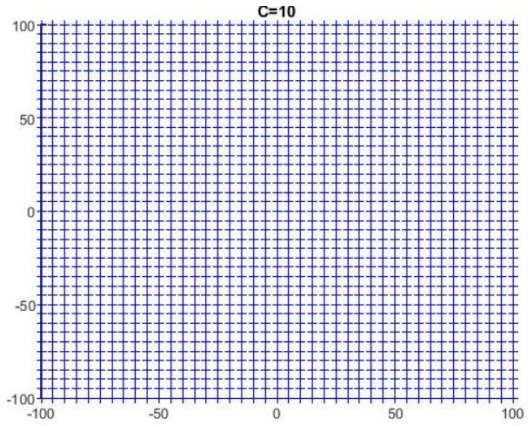


Seems implementing the quadratic kernel leads ignoring of red datapoints!

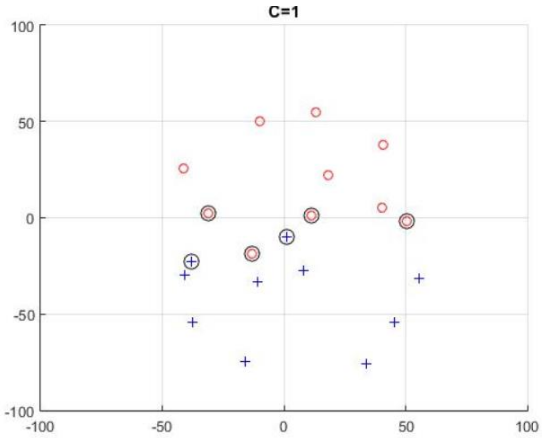
h.

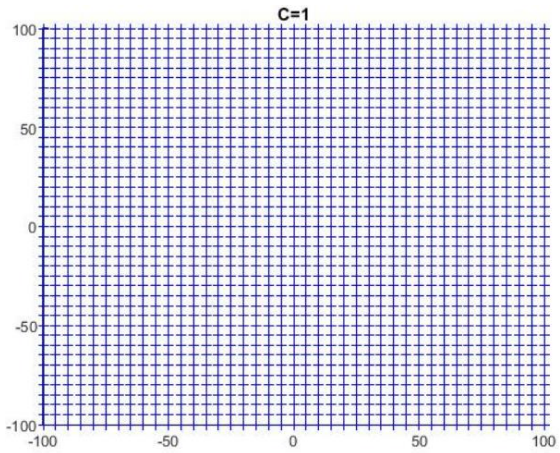
(C = 10)


C=10
100
50
0
-50
-100
-100
-50
0
50
100

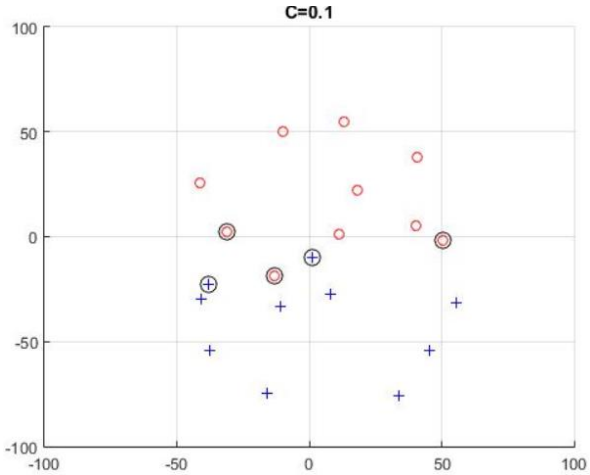

C=10
100
50
0
-50
-100
-100
-50
0
50
100

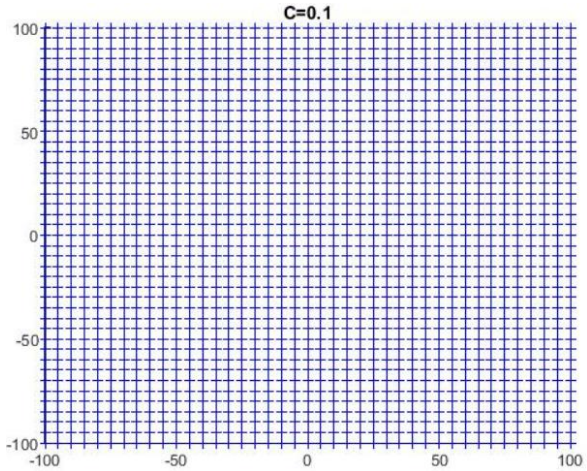
(c = 1)


C=1
100
50
0
-50
-100
-100
-50
0
50
100



(c = 0.1)





As the parameter ‘C’ increases the variance of the model tends to increase, hence, the training error decreases and simultaneously the margin increases and vice versa.

6. a.

(Related script is: **p6.a**) **(Solved by Python)**

```
patch extraced image matrix:
 [[ 3  3  3 ... 39 37  5]
 [ 3  3  3 ... 39 35  4]
 [ 3  3  4 ... 38 38  4]
 ...
 [ 5  7  7 ... 33 16  5]
 [ 6  7  7 ... 33  6 19]
 [ 6  7  7 ... 33  2 33]]
patch extraced image matrix shape:
 (64, 1350)
```

b. (Related script is: **p6.b**)

```
1st biggest eigen-value is:
327601.15508234716
corresponding eigen-vector is:
 [ 1.21470474e-01  1.73473707e-01 -1.97857292e-01 -4.34302750e-02
  6.70055789e-02  2.72696143e-01 -7.70942106e-02  2.20365301e-01
 -2.65718351e-01 -9.95971150e-02 -9.33477740e-02  3.61785507e-01
 -4.60718447e-02  1.94580802e-01  1.62961170e-01 -3.54577586e-01
  1.57810692e-01  1.48831563e-01 -2.36592863e-02  1.19570534e-01
  3.04814360e-01 -6.53660587e-03 -3.93008709e-02 -4.86440969e-02
 -6.17373402e-03 -1.49178032e-01  7.12543990e-02 -1.78302439e-01
 -5.99033999e-02 -1.14680527e-01 -2.70295754e-01 -7.09294952e-02
  1.18994382e-01 -1.44437863e-02  2.20179847e-02 -1.01787393e-01
  3.49875508e-02 -4.92339033e-02  8.29504280e-03 -1.29284066e-02
 -5.02296792e-03 -1.06887453e-01  5.71634550e-03  2.04938432e-02
 -3.72728343e-06  1.42092492e-02 -5.72518187e-03  2.87987221e-02
  1.77199402e-02  3.69338714e-02 -4.65175182e-02 -5.75438864e-03
  1.41029383e-03  5.64168644e-03 -2.53447298e-03  2.77942939e-02
  6.17545874e-02  1.98244066e-03  3.04468482e-02 -4.62164885e-02
 -1.18169617e-02 -2.13719427e-02  3.67555311e-02  4.09907244e-03]
2st biggest eigen-value is:
17556.248248324264
corresponding eigen-vector is:
 [ 0.12400416  0.14040671 -0.17597622  0.03910798 -0.07241145  0.21411328
  0.09246566  0.20004021 -0.0390126  -0.0193197  -0.21794224  0.11533818
 -0.06019659 -0.00459239 -0.08804577 -0.14616581  0.14719846 -0.16949958
  0.0450442   0.04912508 -0.4044589   0.00535826 -0.02160375  0.15487313
  0.03456892 -0.20372945 -0.07694805 -0.04107012  0.07059579  0.07089179
  0.26533025  0.34772974 -0.13837784  0.10816922  0.14071918  0.16719343
 -0.22928711  0.00888375 -0.01582016 -0.0071815   0.01990884  0.04818369
  0.060081   -0.05429138 -0.09300315  0.01649849  0.05681312 -0.11029378
 -0.07005266 -0.13589187  0.09703994 -0.038294    0.0047069  -0.03263127
  0.05978006 -0.02403855 -0.02910692  0.0316934  -0.08347474  0.02780222
  0.02067231  0.12603213 -0.09014985  0.06390433]
3st biggest eigen-value is:
3141.5701634757324
corresponding eigen-vector is:
 [ 0.1254903   0.08832029 -0.14119957  0.17777289 -0.11734737  0.10314953
 -0.01326631  0.13937874  0.19565377 -0.17622927 -0.19850112 -0.20629534
 -0.12320621 -0.08968153 -0.07242063 -0.07480381  0.13819184 -0.14337832
 -0.07621573  0.13722968  0.02850973  0.05495421  0.26931023  0.14323696
 -0.06288111  0.23079422 -0.06213034 -0.03564848 -0.07582579  0.05970958
 -0.03984275 -0.01087247  0.13808586 -0.20840549 -0.09514998  0.28523468
  0.14699963  0.06653235 -0.02826597 -0.05680078  0.07670204  0.13078384
 -0.10532159  0.01465645  0.16201168  0.05263771 -0.08616651  0.04771397
 -0.05966322  0.35194641 -0.02907885  0.10665256 -0.07238176  0.01007797
 -0.10300411  0.00353991 -0.0576137  -0.05756053  0.16477882 -0.06178213
 -0.04427379 -0.07871625  0.0119553  -0.05911812]
```

```
4st biggest eigen-value is:
2686.102330413219
corresponding eigen-vector is:
 [ 0.12687485  0.02122882 -0.12494589  0.25922991 -0.00967637  0.04361238
 -0.23751692  0.0757649   0.23101947 -0.06233558 -0.13482406  0.02615339
  0.33952675  0.00122948 -0.01921744  0.03786847 -0.04981528  0.04737179
 -0.22082435 -0.15704015 -0.06883051  0.316452   -0.1104407  -0.00603306
  0.04545536  0.11491     0.05259395  0.00693771 -0.22676481  0.02574803
 -0.18381089 -0.17579776 -0.25468776 -0.02356133 -0.06829686 -0.02100016
 -0.13825964 -0.04078655  0.16399212 -0.11518874 -0.10406279  0.09721079
 -0.0922509   0.02229737 -0.22488401  0.05600472  0.03936291 -0.07274341
 -0.03912088 -0.18581293 -0.02250902 -0.01193789  0.04933052 -0.01702115
  0.06540078  0.07602986  0.10295073 -0.13527997  0.01186809  0.02447785
 -0.01407167  0.07171362  0.0737872  -0.04413965]
5st biggest eigen-value is:
903.460858195169
corresponding eigen-vector is:
 [ 1.26984684e-01 -5.67387008e-02 -1.55483277e-01  1.90716120e-01
  1.75805297e-01 -4.39414753e-02 -6.09566659e-02  1.21503952e-01
  1.95319732e-01  1.37468740e-01  7.00433280e-03  2.57331808e-01
 -7.16925032e-02 -1.63745721e-01  1.03025097e-01  2.04142238e-01
 -3.22876727e-03 -7.87202487e-02 -8.14598069e-02 -1.90966061e-01
  3.85597919e-02 -1.94846778e-01  1.05163366e-01 -6.42573813e-02
 -3.25283191e-01 -7.77669282e-03  1.62296415e-01  1.88242054e-01
 -1.25332392e-01  5.43302717e-02 -3.67949179e-02  1.43839941e-01
  1.95889255e-01  2.37767135e-02  1.25715026e-01  2.07939524e-03
  1.51572665e-01 -3.46184054e-02  8.66573602e-02  1.62447545e-02
 -1.73024659e-01 -6.07721666e-02  3.32673563e-01 -1.48631191e-01
 -1.06082511e-01  7.20570185e-02 -4.00533494e-02 -4.50420076e-02
  1.52363822e-02  4.44019008e-04 -6.17980940e-02 -1.04388876e-02
 -1.15056310e-01  1.66808791e-03  1.31764718e-02  2.34940761e-02
 -7.19170733e-02  1.19930490e-01  2.41549319e-04 -8.82571296e-02
  3.60292744e-02 -3.17751447e-02 -1.67299608e-01 -4.49376434e-04]
6st biggest eigen-value is:
777.0156466410547
corresponding eigen-vector is:
 [ 0.12592634 -0.10688378 -0.19825336  0.08867915  0.17908039 -0.0847092
  0.08475463  0.21617471  0.14097689  0.01388676  0.15612101  0.09425403
 -0.13119641 -0.05266898  0.08099495  0.2529085  -0.07900976 -0.01942668
  0.20081015 -0.13630907  0.13665417  0.07855188  0.06184563  0.08782043
  0.24458754 -0.34628068 -0.14822579 -0.03953622 -0.12230511 -0.03347739
  0.16369991 -0.10648184 -0.02368698 -0.24704875  0.0474273   0.00481488
 -0.00059947 -0.08201992 -0.0642951  -0.07894963 -0.02804839 -0.11474603
 -0.23779175  0.06547427  0.1427214   0.00851408  0.16077682  0.09522392
 -0.19034117  0.03769931  0.09508342 -0.07485477  0.05729869 -0.02455214
  0.02026525 -0.12734805  0.05589126 -0.10031015 -0.11247778 -0.112636
 -0.00169466 -0.02418865  0.06743835  0.03989458]
```

.

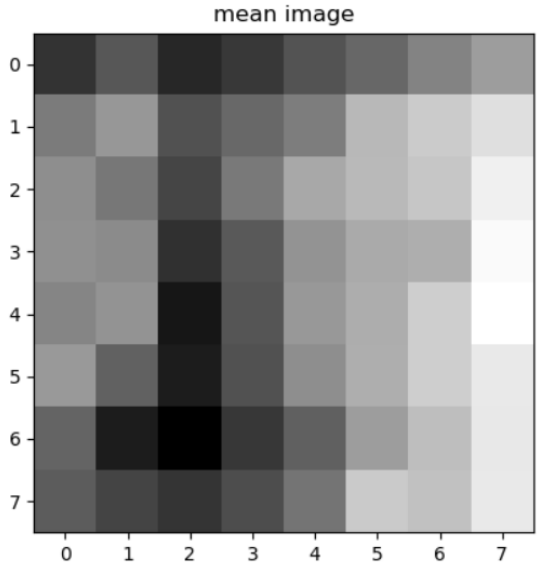
.

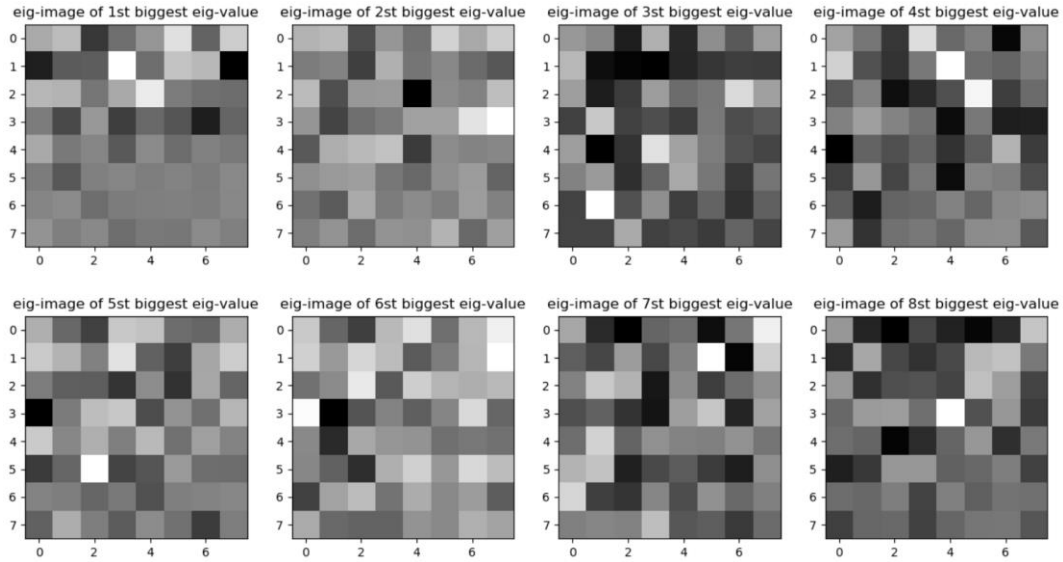
.

.

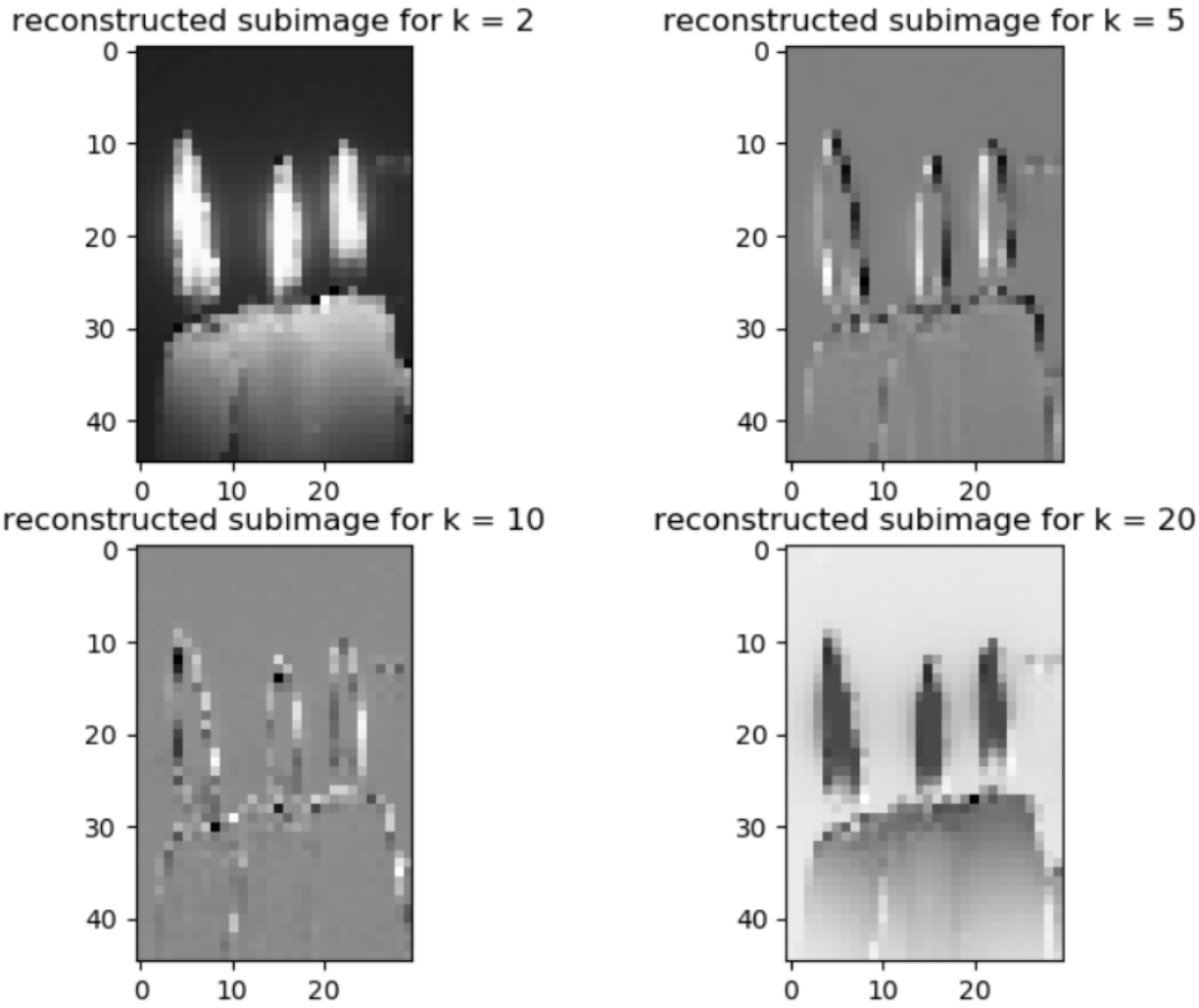
.

```
18st biggest eigen-value is:
51.149418701727164
corresponding eigen-vector is:
 [ 0.12468485  0.14894677 -0.09951806 -0.00951805 -0.06740253  0.15352631
  0.1692324  -0.01620113 -0.06640365  0.15505171  0.07449441 -0.07748098
  0.09035573 -0.04473454 -0.12159791  0.14813386 -0.14623763 -0.05645451
  0.19789954 -0.09998684 -0.17968534 -0.0652407  -0.13160942 -0.14718722
 -0.04654417  0.02117508  0.22189223 -0.20733545  0.02707669 -0.13262919
 -0.21277481 -0.26104594  0.22168033 -0.09357343  0.01101524  0.0221519
 -0.24238797 -0.0025303   0.04715285 -0.07896704  0.01405658  0.02552643
 -0.02688247 -0.06429239  0.13164878 -0.03589216  0.12733781 -0.03666775
 -0.05774139  0.09694314  0.20573064 -0.0663522  -0.09514322 -0.1504117
  0.01366655  0.06783979 -0.18665787  0.073811    0.19404527  0.10890259
 -0.12242931  0.14737723 -0.19577358 -0.06435526]
19st biggest eigen-value is:
46.97065679703919
corresponding eigen-vector is:
 [ 0.12619204  0.09645819 -0.04308156  0.12691493 -0.16063993  0.05532937
  0.07979332 -0.08207023  0.11800634 -0.00843781  0.00920736 -0.26594332
 -0.12348391 -0.07331306  0.05542973  0.00114717 -0.03075217  0.05500755
  0.0893272   0.10460736  0.32866881 -0.21507406 -0.00804772 -0.13284293
  0.06729524 -0.24319923  0.00857474 -0.04250392  0.20477444 -0.05621673
  0.0910263   0.15378351 -0.1115535   0.02023323 -0.06026206 -0.10557108
  0.00700719  0.04194707  0.21562627  0.01825086 -0.21194583  0.10102075
 -0.01322159 -0.20859599 -0.26269818  0.12021783  0.01965093 -0.07246863
 -0.05249108 -0.09968185  0.07872048  0.06328113  0.00719499  0.00081147
 -0.06972726  0.15591082 -0.08785613 -0.14312069  0.21850272 -0.00538056
 -0.1204635  -0.01285733  0.22363522 -0.1091149 ]
20st biggest eigen-value is:
39.88845953594597
corresponding eigen-vector is:
 [ 0.12826909  0.02586182  0.00263547  0.2078777  -0.06448919 -0.01231117
 -0.14314535 -0.20201326  0.05133214  0.01293909  0.08862442 -0.03750201
  0.19669921  0.11771099  0.08876467 -0.13068209  0.03184104  0.08187874
  0.04306147  0.07363723 -0.05792341  0.09080938 -0.16706917 -0.0717313
  0.02503202 -0.04661965 -0.08305575  0.07646901  0.01744423  0.1262749
  0.07455151  0.05294277  0.19675244  0.1680126   0.07335508  0.18997304
  0.12225216  0.12663482 -0.1536261  -0.05640839 -0.33874986 -0.17364952
 -0.24840431 -0.22510627  0.17329861  0.04572336  0.06111718  0.1253089
  0.15400281  0.07834915  0.16587298 -0.09591145 -0.06110621  0.09685817
  0.16783796  0.12839033  0.02891676  0.13206682 -0.00848576 -0.06090043
 -0.08312119 -0.05865974 -0.01610176  0.22944855]
```



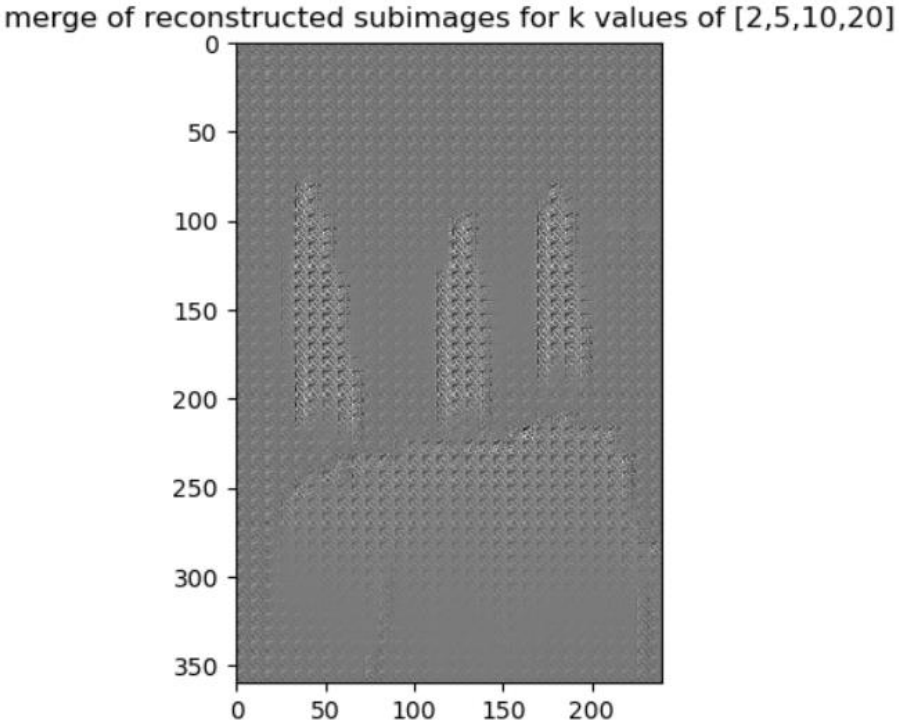


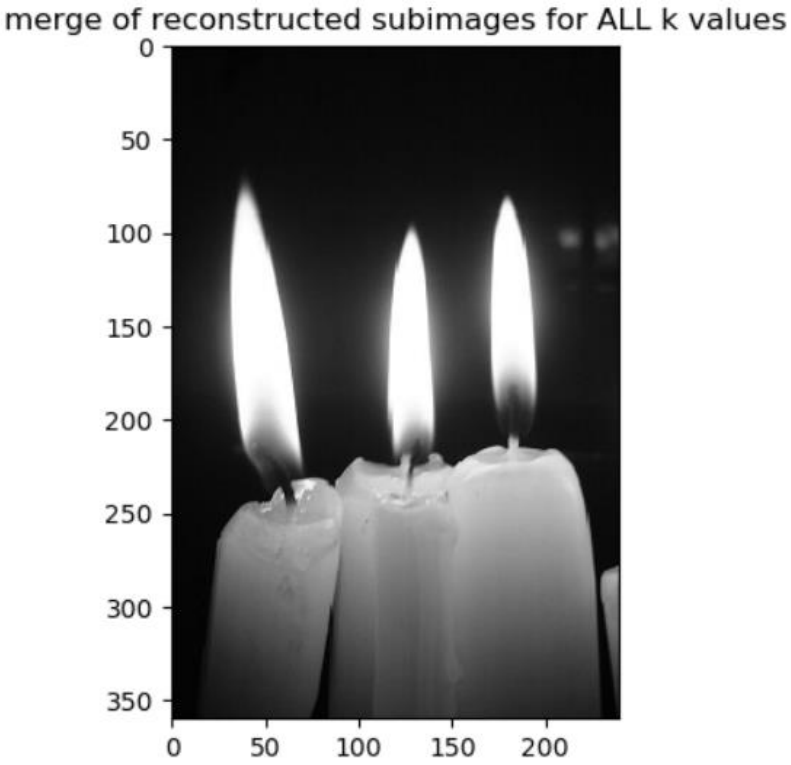
c & d. (Related script is: **p6.c**)



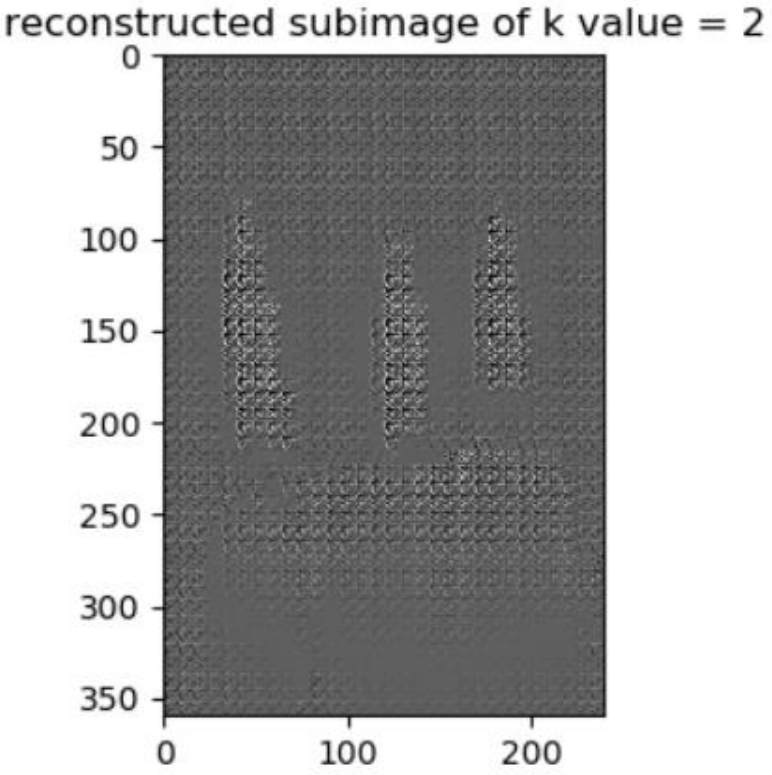
e.

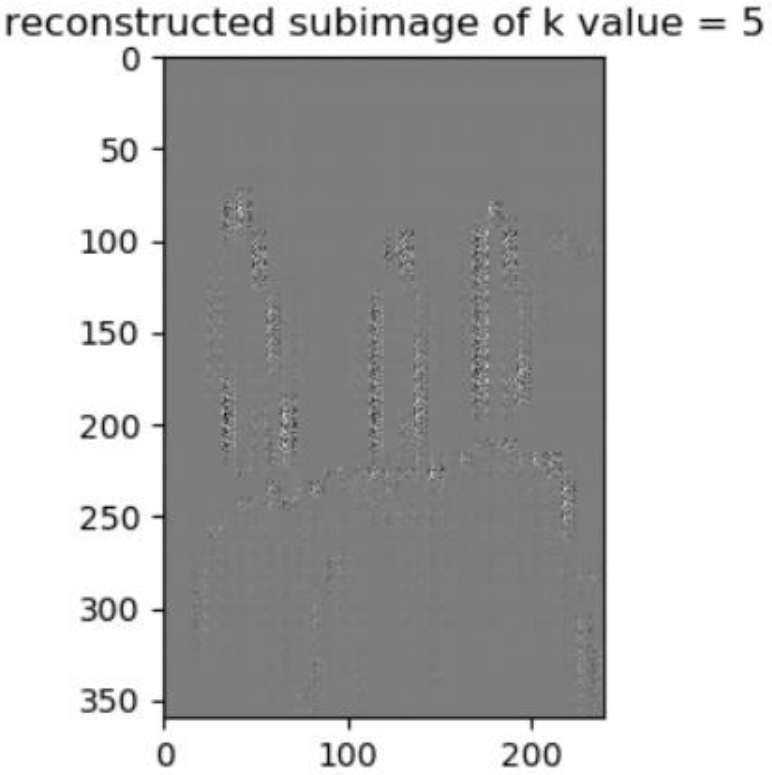
(Related script is: **p6.e and p6.eprime**)

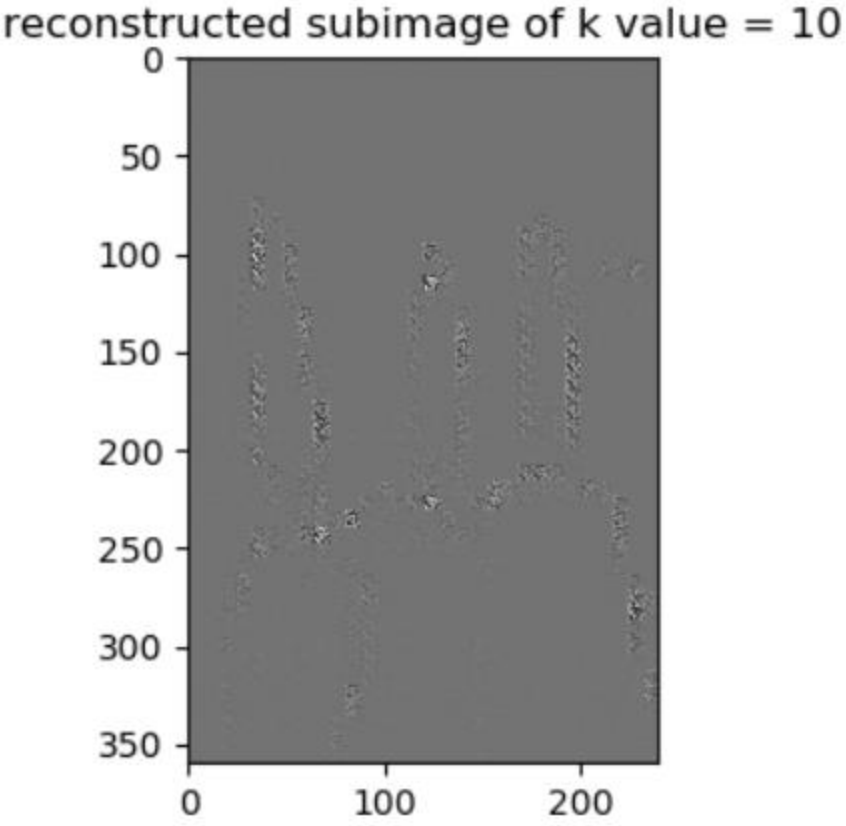


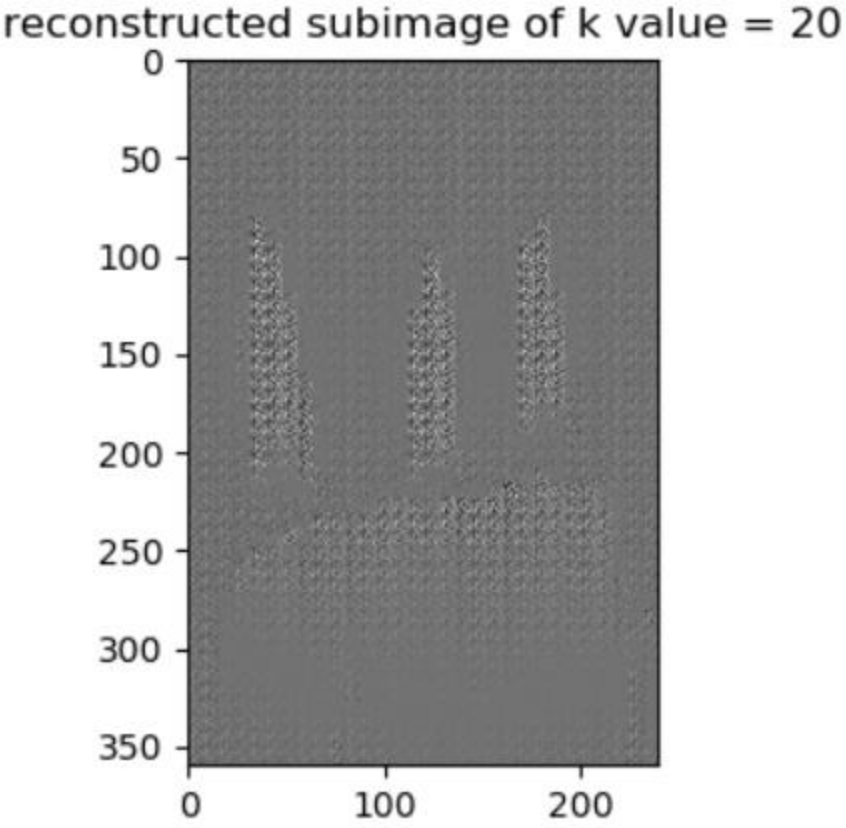


reconstructed subimage of single eigenvector(k value)






reconstructed subimage of k value = 10
0
50
100
150
200
250
300
350
0
100
200


reconstructed subimage of k value = 20
0
50
100
150
200
250
300
350
0
100
200

Different ‘K’ value represents different eigenvectors with descending order with respect to corresponding eigenvalues. Therefore, each K in fact shows one of the principle components of the data and as the number of ‘K’ increases, probably the significance of the corresponding eigenvector decreases.

f.

???

7.

a.

PCA in other words means eigenvectors of standardized data's covariance. Since total number of couple of eigenvalues and corresponding eigenvectors of a n-dimensional space is equal to 'n', the maximum number of principle components would be 'n' and any number greater than 'n' doesn't make any sense.

b.

From a point of view, the fact that PCA method's output is not affected by the order of Principle Components is knows as 'Label Switching'.

c.

Yes! SVD and PCA may produce same results. Both PCA and SVD are used to reduce high-dimensional data set into fewer dimensions while retaining important information with the help of the fact that both are eigenvalue methods.

The two methods lead to same results if the data is preprocessed to have zero mean before implementation (data is centered).

Since:

Assuming X is the **centered** data matrix:

Covariance Matrix = XX^T

for normalized eigenvectors (orthonormal):

$$XX^T = WDW^T \quad (I)$$

Applying SVD to the data matrix X:

$$X = U\Sigma V^T$$

$$XX^T=(U\Sigma V^T)(U\Sigma V^T)^T$$

$$XX^T=U\Sigma^2U^T \quad (II)$$

Now considering both (I) and (II) the correspondence of SVD and PCA is easily seen (the square roots of the eigenvalues of XX^T are the singular values of X).

d.

The curse of high dimensionality in data such as images stems from the fact that each pixel in an image plays a role as a feature which leads to like thousands of features for a single datapoint (image). This lastly consequents in lack of training data compared to dimensions (features) which leads to low rank covariance matrix, high number of eigenvectors, zero eigenvalues and so on. High dimensionality also leads to complexity increase such as covariance computation and so on. Another problem of high dimensionality and low training data is the problem of overfitting due to redundant dimensions.

To tackle mentioned issues, it is mainly suggested to patch extract the data (image) and see one single image as number of images extracted from the main image. This way implicitly, the number of training data is increased and features are decreased

e.

PCA may be used for noise reduction (but not noise elimination) in cases which noises play insignificant and negligible role in variation of variance, since, PCA selects principle components based on variance.

f.

???

g.

???

h.

???