شناسایی آماری الگو

تمرین های سری پنج

فرهاد دلیرانی
۹۶۱۳۱۱۲۵

dalirani@aut.ac.ir

dalirani.1373@gmail.com

تمام‌کدها با پایتون 3.6 نوشته شده‌اند.

همچنین از پکیج‌های زیر استفاده کرده ام:

- numpy
- sklearn
- scipy.io
- matplotlib

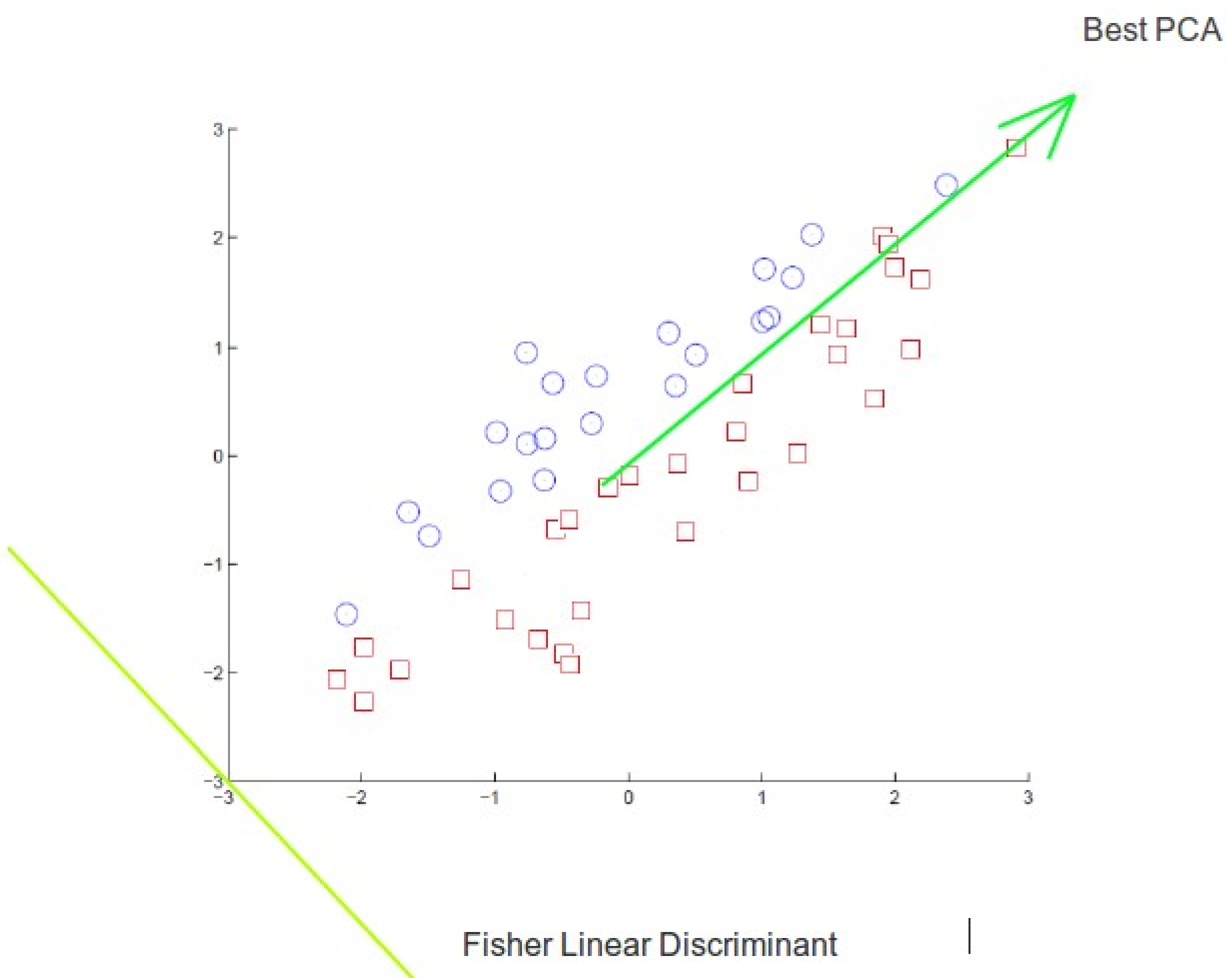
البته برای راحتی در نصب پایتون 3.6 و پکیج های مربوط به دیتاساینس که numpy و matplotlib هم جزیی از آن پکیج‌ها هستند از Anaconda 5.0.0 استفاده کرده‌ام که همه‌ی موارد گفته شده را بدون دردسر و سختی نصب می کند. تنها کافی است آن را از https://www.anaconda.com/download دانلود کنید و Installer باقی کار را انجام می‌دهد. البته به صورت مستقل هم، می‌توان آن‌ها را نصب کرد.

زبان برنامه نویسی: پایتون 3.6
پکیج‌ها: پکیج‌های گفته شده را برای راحتی در نصب با Anaconda نصب کردم.
ورژن Anaconda من: Anaconda 5.0.0 For Linux Installer که البته همین ورژن برای سایر سیستم عامل‌ها هم موجود است.
محیط برنامه نویسی: pyCharm Community Edition

سوال ۱)

PCA یک تبدیل خطی به مختصات جدید متعامد است که محور اصلی معادل با بزرگترین Eigen Value است که بیشتر مقدار را دارد.

Fisher Linear Discriminant Analysis برای داده‌ها با دو کلاس برابر با خطی می‌شود که موجب بیشترین separateness در نگاشت داده‌ها به فضای یک بعدی می‌شود.

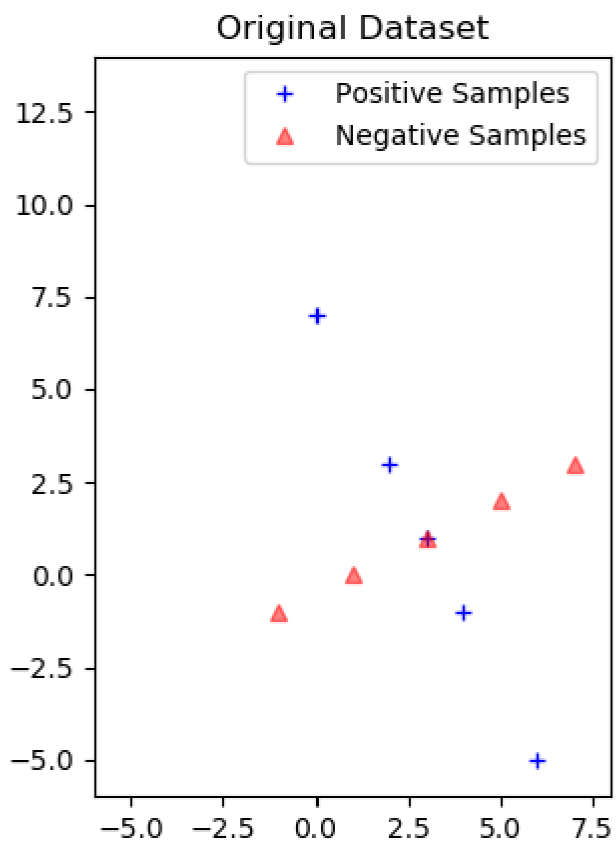


سوال ۲)

این سوال را به دو روش حل کرده‌ام. روش اول محاسبات همان‌طور که سوال خواسته است به صورت دستی انجام شده است و در روش دوم برنامه‌ای نوشته‌ام که همه چیز را محاسبه می‌کند.

روش اول: در این روش محاسبات با صورت دستی انجام شده است.

بخش a) در تصویر زیر داده‌ها رسم شده‌اند.



بخش b, c, d) در تصویر زیر بخش‌های b(کم کردن میانگین از داده‌ها), c(پیدا کردن ماتریس کواریانس و eigen values ها), d(دقت استفاده از eigen vector بزرگترین eigen value) رامشاهده می‌کنید.

Problem 2 :

$$X = \begin{bmatrix} 3 & 2 & 4 & 0 & 6 & 3 & 1 & 5 & -1 & 7 \\ 1 & 3 & -1 & 7 & -5 & 1 & 0 & 2 & -1 & 3 \end{bmatrix}$$

b)

$$\text{mean} = \frac{1}{10}\begin{bmatrix} 3+2+4+0+6+3+1+5-1+7 \\ 1+3-1+7-5+1+0+2-1+3 \end{bmatrix}$$

$$= \frac{1}{10}\begin{bmatrix} 30 \\ 10 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$$\rightarrow B = X - \text{mean} = \begin{bmatrix} 0 & -1 & 1 & -3 & +3 & 0 & -2 & 2 & -4 & 4 \\ 0 & 2 & -2 & +6 & -6 & 0 & -1 & 1 & -2 & 2 \end{bmatrix}$$

c)

$$\text{Cov} = \frac{1}{10}\begin{bmatrix} 0 & -1 & 1 & -3 & 3 & 0 & -2 & 2 & -4 & 4 \\ 0 & 2 & -2 & 6 & -6 & 0 & -1 & 1 & -2 & 2 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ -1 & 2 \\ 1 & -2 \\ -3 & 6 \\ 3 & 6 \\ 0 & 0 \\ -2 & 1 \\ 2 & 1 \\ -4 & -2 \\ 4 & 2 \end{bmatrix}$$

$$= \frac{1}{10}\begin{bmatrix} 60 & -20 \\ -20 & 90 \end{bmatrix} = \begin{bmatrix} 6 & -2 \\ -2 & 9 \end{bmatrix}$$

$$\det\left(\begin{bmatrix} 6-\lambda & -2 \\ -2 & 9-\lambda \end{bmatrix}\right) = 0 \rightarrow$$

$$(6-\lambda)(9-\lambda) = 4 \rightarrow \lambda^2 - 15\lambda + 54 - 4 = 0$$

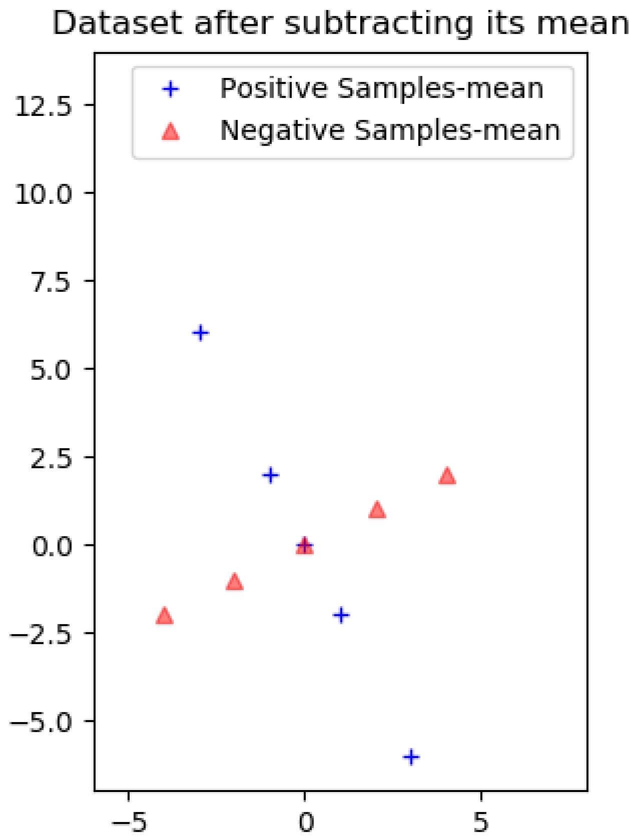
$$\rightarrow \lambda^2 - 15\lambda + 50 = 0 \rightarrow (\lambda-5)(\lambda-10) = 0$$

$$\rightarrow \boxed{\lambda = 5, 10}$$

d)

$$\frac{10}{10+5} * 100 = \frac{1000}{15} = 66.6\,\%$$

بخش دیگر از قسمت b) در قسمت b خواسته شده است نمونه‌ها را بعد از کم کردم میانگین رسم کنیم که در تصویر بالا موجود نیست. در شکل زیر پاسخ آن قسمت را مشاهده می‌کنید:



بخش e) پیدا کردن `Eigen vectors` و رسم آن‌ها: در تصویر زیر نحوه‌ی محاسبه‌ی `eigen vector` ها را مشاهده می‌کنید:

E)

$$\begin{bmatrix} 6 & -2 \\ -2 & 9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = 10 \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$\begin{bmatrix} 6-10 & -2 \\ -2 & 9-10 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -4 & -2 \\ -2 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{array}{cc|c} -4 & -2 & 0 \\ -2 & -1 & 0 \end{array} \longrightarrow \begin{array}{cc|c} -4 & -2 & 0 \\ 0 & 0 & 0 \end{array} \longrightarrow \begin{bmatrix} -\frac{1}{2}x_2 \\ x_2 \end{bmatrix} \longrightarrow \begin{bmatrix} +\frac{1}{2} \\ -1 \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} 1 \\ -2 \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{bmatrix} \Rightarrow \text{eign vector for } \lambda=10 = \boxed{\begin{bmatrix} 0.447213 \\ -0.894427 \end{bmatrix}}$$

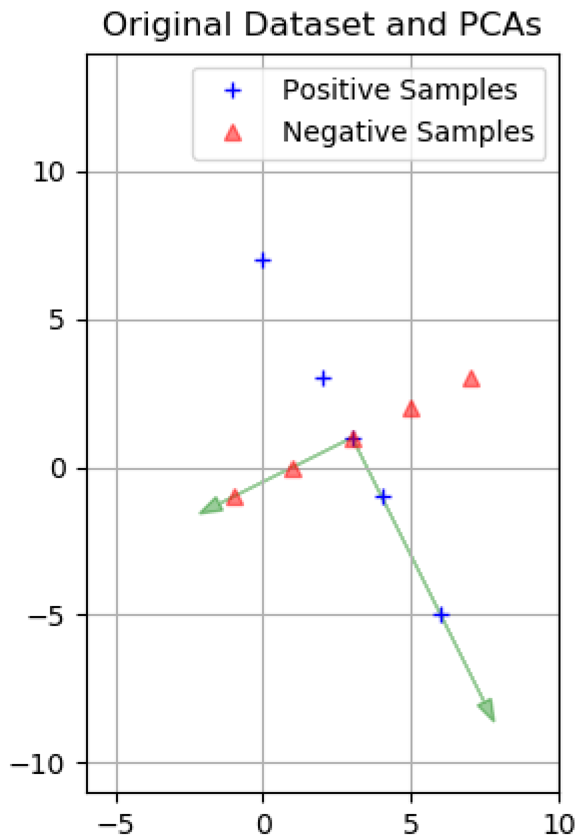
$$\begin{bmatrix} 6 & -2 \\ -2 & 9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = 5 \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{array}{cc|c} 1 & -2 & 0 \\ -2 & 4 & 0 \end{array} \longrightarrow \begin{array}{cc|c} 1 & -2 & 0 \\ 0 & 0 & 0 \end{array} \longrightarrow \begin{bmatrix} 2x_2 \\ x_2 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} \Rightarrow \text{eigen vector for } \lambda=5 = \boxed{\begin{bmatrix} 0.894427 \\ 0.447213 \end{bmatrix}}$$

بعد از به دست آوردن eigen vector ها در تصویر زیر آن‌ها را رسم کرده‌ام:



بخش f) در این بخش از سوال خواسته شده است که نمونه‌ها را بر روی PCAها نگاشت کنیم. برای این کار اگر ماتریس نمونه‌های ورودی برابر x باشد و ماتریس نمونه‌های نگاشت شده بر `eigen vector` اول برابر y باشد و `eigen vector` اول برابر با u باشد رابطه‌ی زیر بر قرار است:

```
y = transpose(u) * x
```

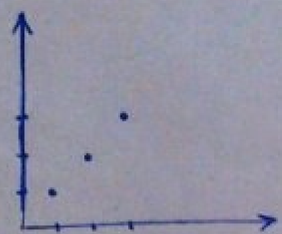
برای `eigen vector` دوم نیز به همین ترتیب. با استفاده از رابطه‌ی بالا مختصات نقاط جدید بر روی دو `eigen vector` را به دست می‌آوریم

```
Projected points on pca with eigenvalue:  5.0
 [[-3.13049517 -3.13049517 -3.13049517 -3.13049517 -3.13049517 -3.13049517
  -0.89442719 -5.36656315  1.34164079 -7.60263112]]

Projected points on pca with eigenvalue:  10.0
 [[ 0.4472136  -1.78885438  2.68328157 -6.26099034  7.15541753  0.4472136
   0.4472136   0.4472136   0.4472136   0.4472136 ]]
```

در روش دوم همه‌ی کارها را با استفاده از نوشتن کد انجام داده‌ام که کدها در `problem2.py` موجود است.

سوال ۳)

problem 3)



$$\text{observation Matrix} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

$$\text{Mean} = \frac{1}{3}\begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$\text{newObservationMatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} - \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{covariance} = \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix}$$

eigen values:

$$\det\begin{bmatrix} \frac{2}{3}-\lambda & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3}-\lambda \end{bmatrix} = 0 \rightarrow \left(\frac{2}{3}-\lambda\right)\left(\frac{2}{3}-\lambda\right) - \frac{4}{9} = 0$$

$$\rightarrow \left(\lambda - \frac{2}{3}\right)^2 - \frac{4}{9} = 0 \rightarrow \lambda^2 - \frac{4}{3}\lambda + \frac{4}{9} - \frac{4}{9} = 0 \rightarrow$$

$$\rightarrow \lambda\left(\lambda - \frac{4}{3}\right) = 0 \rightarrow \left.\begin{cases} \lambda = 0 \\ \lambda = \frac{4}{3} \end{cases}\right\} \underline{\text{eigen values}}$$

$$\rightarrow \begin{bmatrix} \frac{2}{3}-\frac{4}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3}-\frac{4}{3} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} -\frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{2}{3} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\rightarrow \text{eigen vector} = \begin{bmatrix} x_2 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \underline{\text{eigen vector}}$$

a) $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ آن داده‌ها یک PCA بیشتر ندارند و برابر با

است

b) $Y = P^T X$

$$Y = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}^T \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{4}{\sqrt{2}} & \frac{6}{\sqrt{2}} \end{bmatrix}$$

$$\text{mean} = \frac{1}{3} \times \frac{12}{\sqrt{2}} = \frac{4}{\sqrt{2}}$$

$$\rightarrow \text{variance} = \frac{1}{3} \left(\left(\frac{2}{\sqrt{2}} - \frac{4}{\sqrt{2}}\right)^2 + \left(\frac{4}{\sqrt{2}} - \frac{4}{\sqrt{2}}\right)^2 + \left(\frac{6}{\sqrt{2}} - \frac{4}{\sqrt{2}}\right)^2 \right)$$

$$= \frac{1}{3} \left(\left(-\frac{2}{\sqrt{2}}\right)^2 + (\emptyset)^2 + \left(\frac{2}{\sqrt{2}}\right)^2 \right) =$$

$$= \frac{1}{3} \left(2 + 0 + 2 \right) = \frac{4}{3}$$

c) $X = PY \rightarrow X = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{4}{\sqrt{2}} & \frac{6}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$

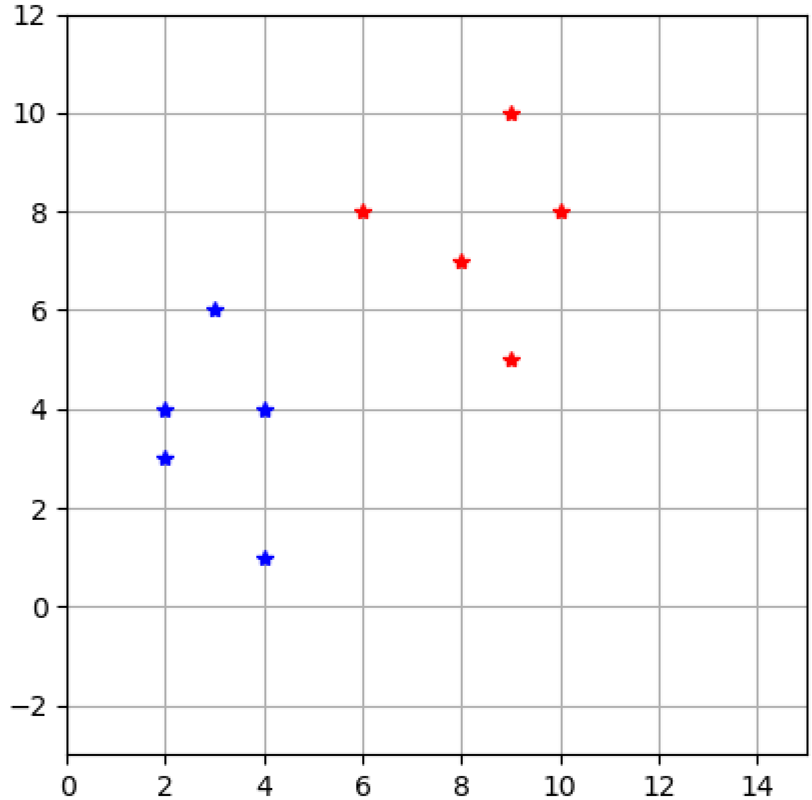
$$E_{construction} = 1 - accuracy_{construction} = 1 - \frac{\frac{4}{3}}{\frac{4}{3}} = \emptyset$$

$$\text{Covariance Matrix} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix} \rightarrow \text{Variance} = tr(\text{covariance}) = \frac{4}{3}$$

سوال 4)

کدهای این بخش از سوال در فایل `problem4.py` موجود است.

بخش a) در شکل زیر داده‌های رسم شده را مشاهده می‌کنید:



بخش B) در این بخش LDA را محاسبه می‌کنیم:

میانگین دو کلاس را اینگونه به دست می‌آوریم:

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$$

که برابر می‌شود با:

```
Mean Class 1:
 [[ 3. ]
 [ 3.6]]

Mean Class 2:
 [[ 8.4]
 [ 7.6]]
```

برای محاسبه‌ی Scatter تو کلاس اینگونه عمل می‌کنیم:

$$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

که برای کلاس یک و دو برابر می‌شود با

```
Scatter Class 1:
 [[  4.    -2. ]
 [ -2.   13.2]]

Scatter Class 2:
 [[  9.2  -0.2]
 [ -0.2  13.2]]
```

و within class scatter را اینگونه محاسبه می‌کنیم:

$$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

که برابر می‌شود با

```
Scatter within(Sw):
 [[ 13.2  -2.2]
 [ -2.2  26.4]]
```

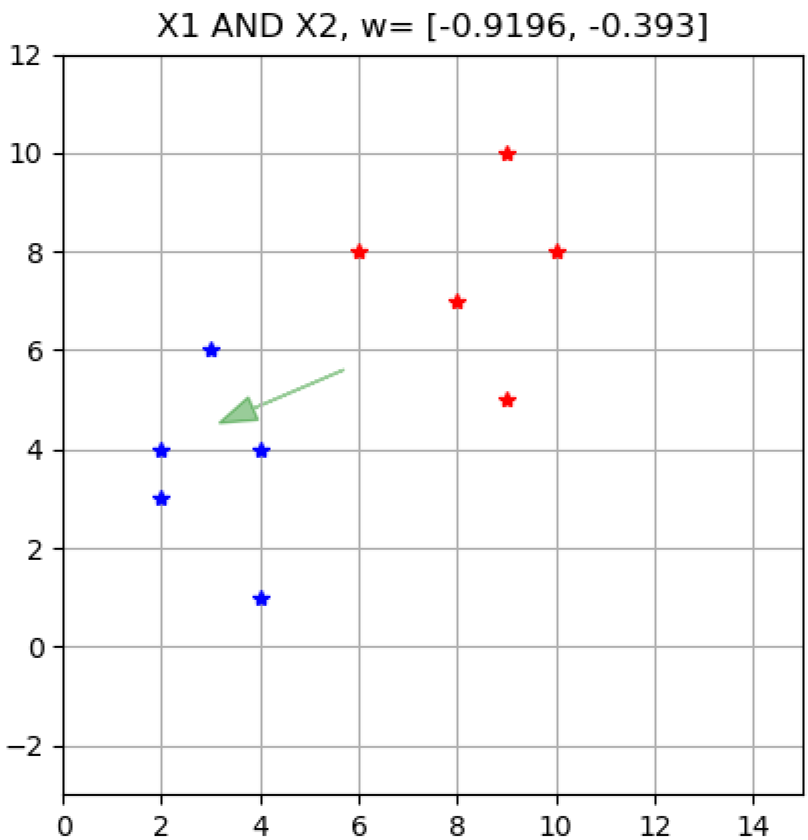
برای محاسبه‌ی بردار w به صورت زیر عمل می‌کنیم:

$$\boxed{\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2).}$$

که برابر می‌شود با

```
W:
 [[-0.44046095]
 [-0.18822023]]

Normalized W:
 [[-0.91955932]
 [-0.39295122]]
```

در تصویر زیر بردار w را مشاهده می‌کنید:



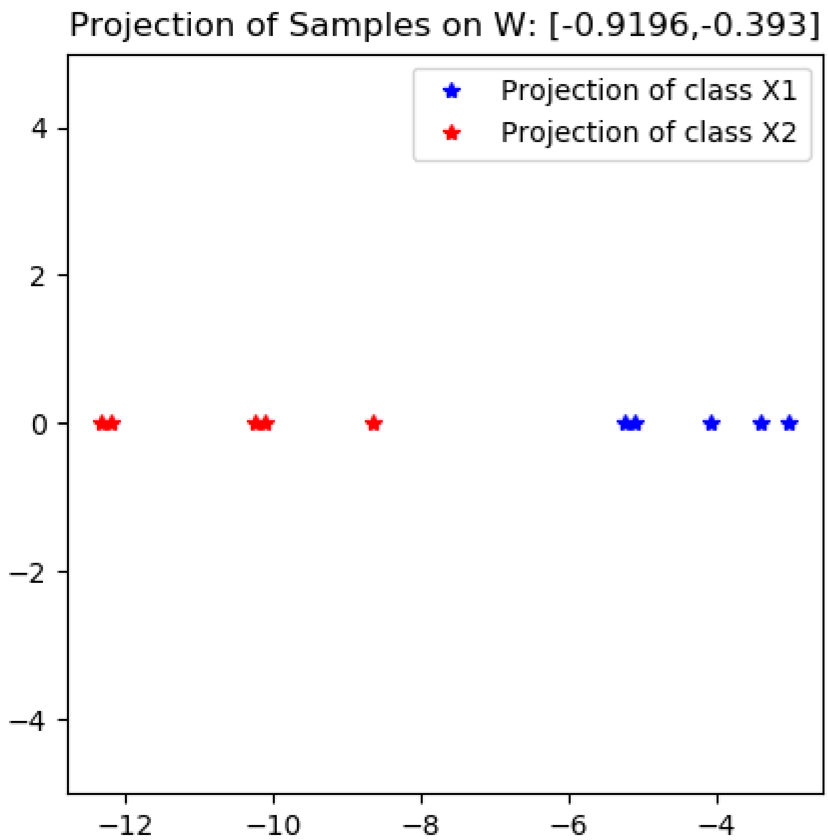
بخش C)برای نگاشت داده‌ها به W از رابطه‌ی زیر استفاده می‌کنیم:

$$y = \mathbf{w}^t\mathbf{x}$$

که در شکل زیر مقادیر عددی نقطه‌های پروجکت شده را می‌بینید:

```
Projection of Class1:
 [[ 4.07118849  3.41092352  3.0179723   5.11638527  5.25004215]]

Projection of Class2:
 [[ 12.20554606   8.66096567  10.24078996  10.10713308  12.33920294]]
```

در تصویر زیر نقاط پروجک شده را مشاهده می‌کنید:



بخش D) همان‌طور که در شکل بالا مشاهده می‌کنید در فضای جدید یک بعدی که داده‌ها به آن نگاشت شده‌اند، داده‌های هر دو کلاس کاملا از هم جدا پذیر هستند و جدا پذیری داده‌ها بر روی این خط ماکسیموم است زیر LDA جواب بهینه را برای جدا پذیری می‌دهد زیرا فاصله‌ی میانگین دو کلاس نسبت به مجموع scatter ها راحداکثر می‌کند.

سوال 5)

کدهای کامنتگذاری شده‌ی این بخش در فایل problem5.py قرار دارند.

برای حل این سوال تابع‌های زیر را نوشته‌ام:

```python
def subtract_mean(observationMatrix):
```

این تابع یک ماتریس observation می‌گیرد و میانگین نمونه‌ها را حساب می‌کند و ماتریس observation جدید را که میانگین از آن‌ها کم شده است را همراه میانگین باز می‌گرداند.

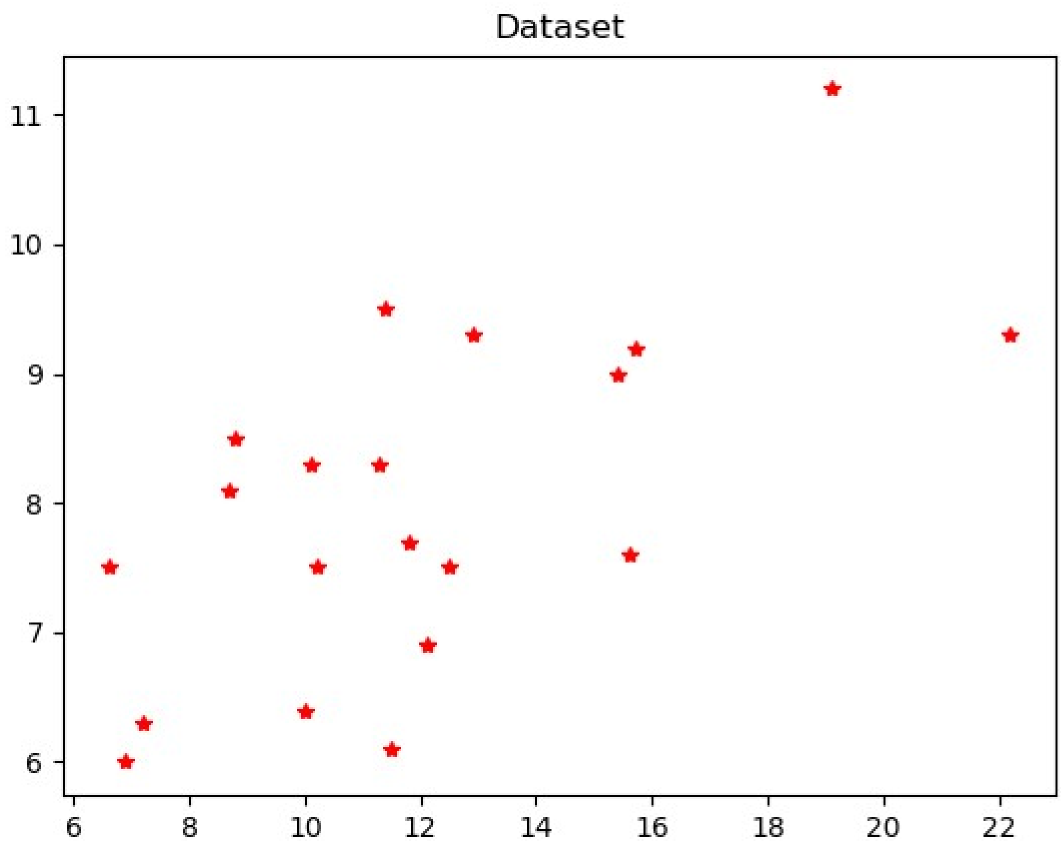
```python
def calculate_covariace(matrixX):
```

این تابع یک ماتریس می‌گیرد و بر اساس رابطه‌ی x*transpose(x)/(n-1) ماتریس کواریانس را محاسبه می‌کند.

```python
def eigen_value_vector(matrixX):
```

این تابع با گرفتم یک ماتریس eigen vector و eigen value های‌آن را حساب می‌کند. در این تابع از تابع‌های پایتون برای محاسبه‌ی eigen value و vector استفاده کرده‌ام.

در تصویر زیر دیتاست را مشاهده می‌کنید:



بخش A)

با استفاده از تابع calculate_covariace که در بالا آن را معرفی کردم، بدون اینکه میانگین را از ماتریس ورودی کم کنم ماتریس کواریانس را حساب کردم که برابر ماتریس زیر می‌شود:

```
Covariance Matrix Without Subtracting:
 [[ 167.63473684  104.63947368]
 [ 104.63947368   69.32736842]]
```

بخش B)

با استفاده از تابع eigen_value_vector ، مقدار eigen value و eigen vector ها را محاسبه می‌کنیم که نتایج آن را در تصویر زیر مشاهده می‌کنید:

```
Eigen values Without Subtracting:
 [ 234.09032615    2.87177911]
Eigen vectors Without Subtracting:
 [[ 0.84414773 -0.53611064]
 [ 0.53611064  0.84414773]]
```

بخش C)

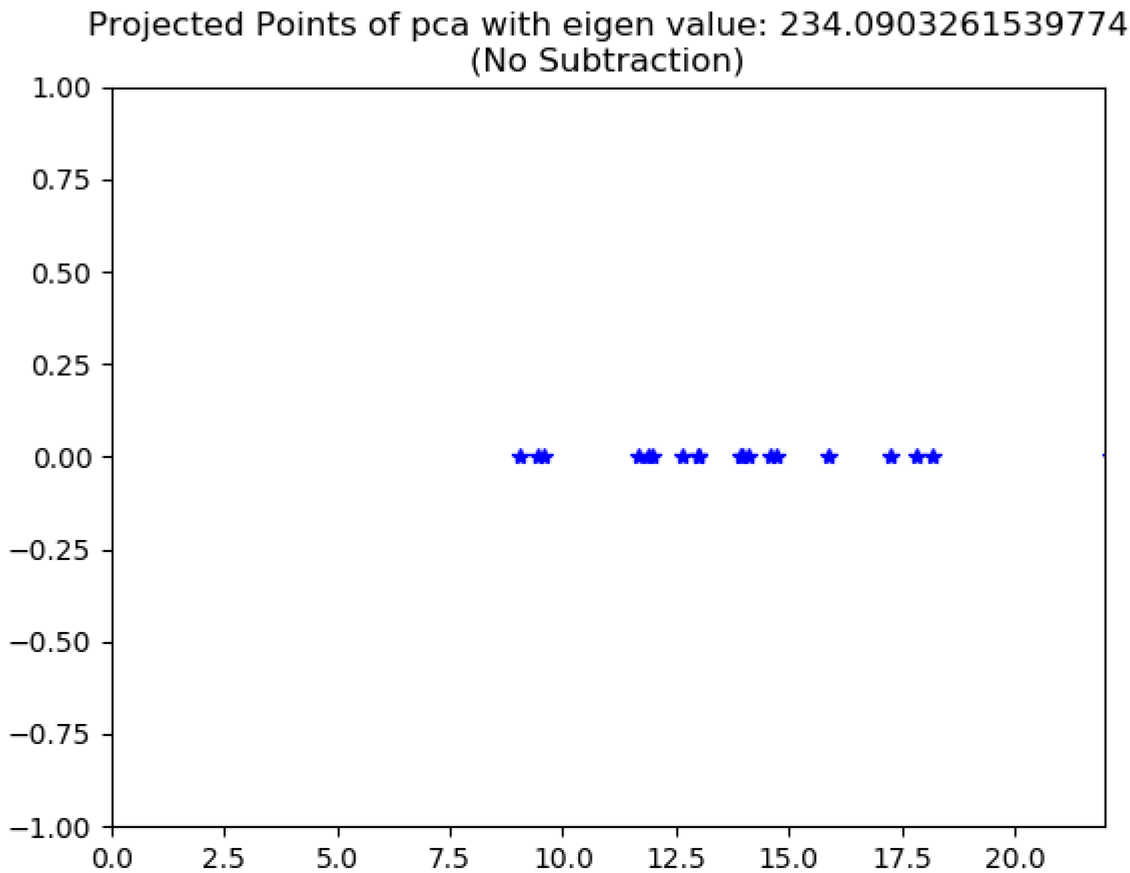
برای نگاشت نقاط به eigen vector u به صورت زیر عمل می‌کنیم:

*y = transpose(u) * x*

که نگاشت نقطه‌ها بر PCA که بیشترین مقدار واریانس را دارد به صورت زیر می‌شود:

```
Projected points on pca with eigenvalue:  234.0903261539774
 [[ 13.91335089   9.59220477  12.97797375  11.68658138  17.82487073
    9.04128314  11.98544041  12.97561033  22.12766072  17.24314539
   23.72590847  12.63113659  18.18533717  11.87258535   9.45536064
   14.08899508  13.9885876   14.57267636  14.71633514  15.8753346 ]]
```

در تصویر زیر نگاشت نقطه‌ها بر روی PCA را مشاهده می‌کنید:



بخش D)

با استفاده از تابع subtract_mean که آن را در بالا معرفی کردم میانگین ماتریس observation و ماتریس observation که میانگینش از آن کم شده است را محاسبه می‌کنیم. با استفاده از تابع calculate_covariace که در بالا آن را معرفی کردم، ماتریس کواریانس را حساب کردم که برابر ماتریس زیر می‌شود:

```
Covariance Matrix With Subtracting:
 [[ 16.05578947   3.46052632]
 [  3.46052632   1.79042105]]
```

با استفاده از تابع eigen_value_vector ، مقدار eigen value و eigen vector ها را محاسبه می‌کنیم که نتایج آن را در تصویر زیر مشاهده می‌کنید:

```
Eigen values With Subtracting:
 [ 16.85093135   0.99527918]
Eigen vectors With Subtracting:
 [[ 0.9746031  -0.22393927]
 [ 0.22393927  0.9746031 ]]
```

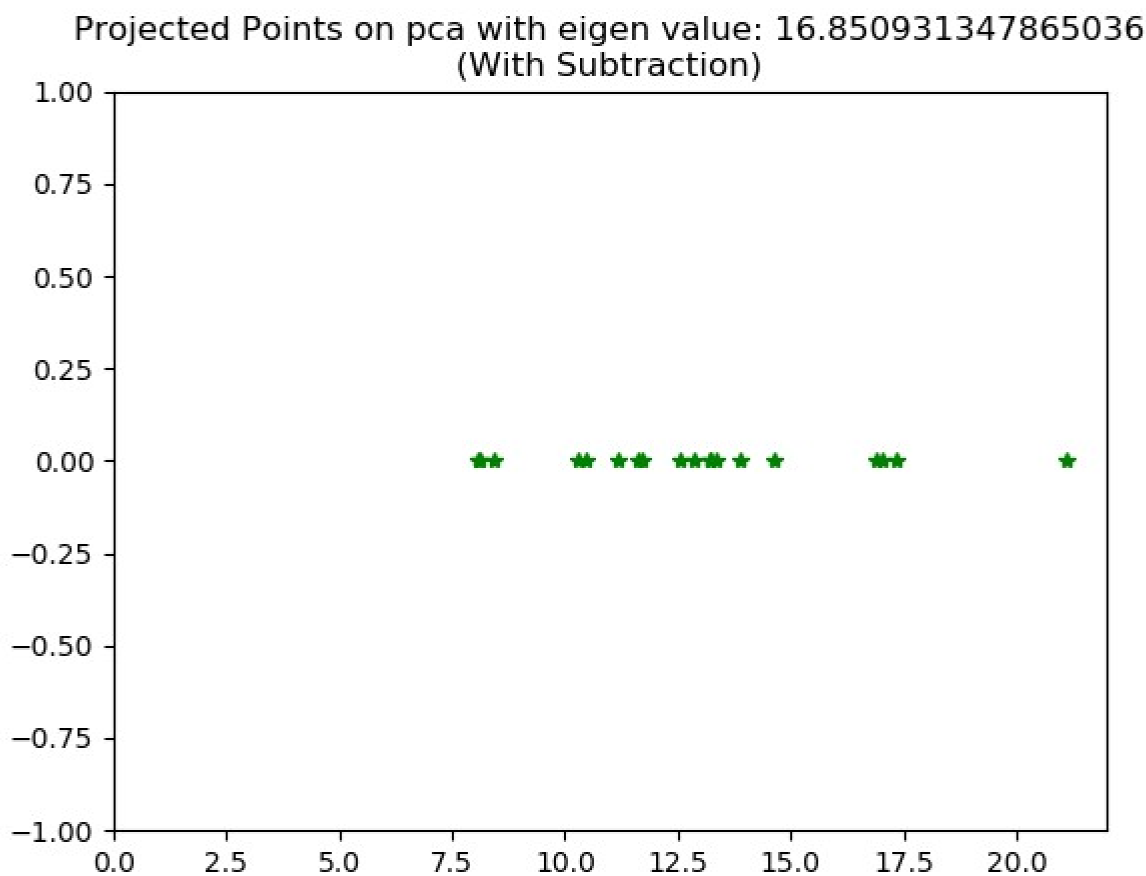
برای نگاشت نقاط به eigen vector u به صورت زیر عمل می‌کنیم:

*y = transpose(u) * x*

که نگاشت نقطه‌ها بر PCA که بیشترین مقدار واریانس را دارد به صورت زیر می‌شود:

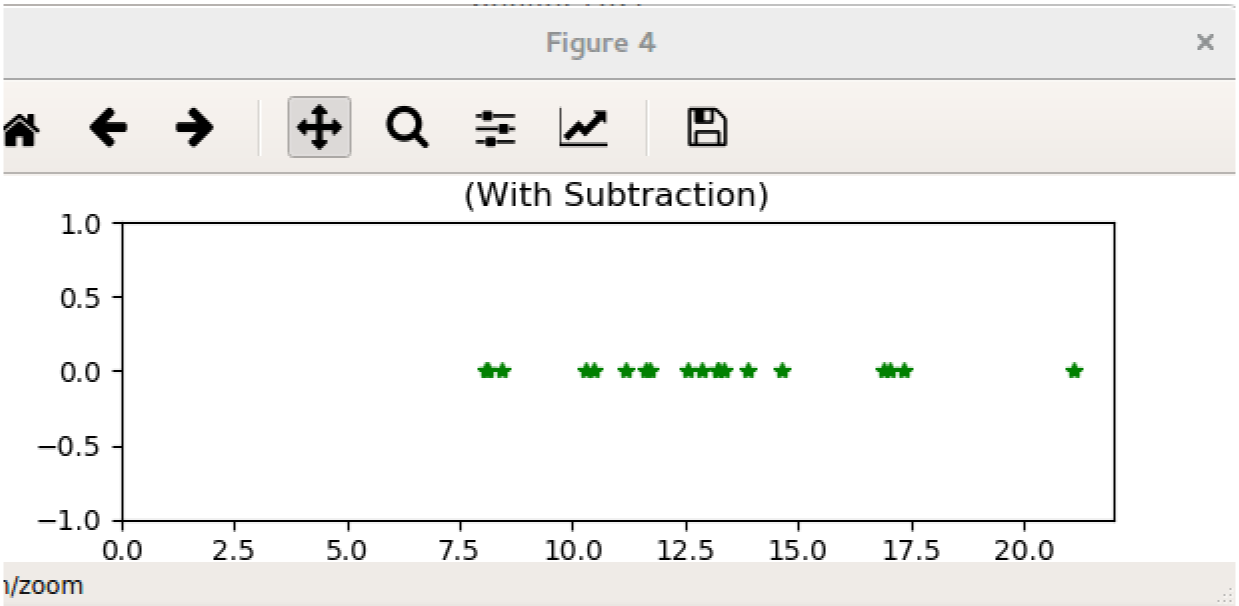
```
Projected points on pca with eigenvalue:  16.850931347865036
 [[ 13.33787848   8.11192499  12.5739652   10.29295506  17.02434117
    8.06839701  10.47999108  11.70218725  21.12303904  16.90574682
   23.71882404  11.62049615  17.36150996  11.17924233   8.42795972
   13.22464896  12.87171097  13.86208328  13.23789841  14.6550152 ]]
```

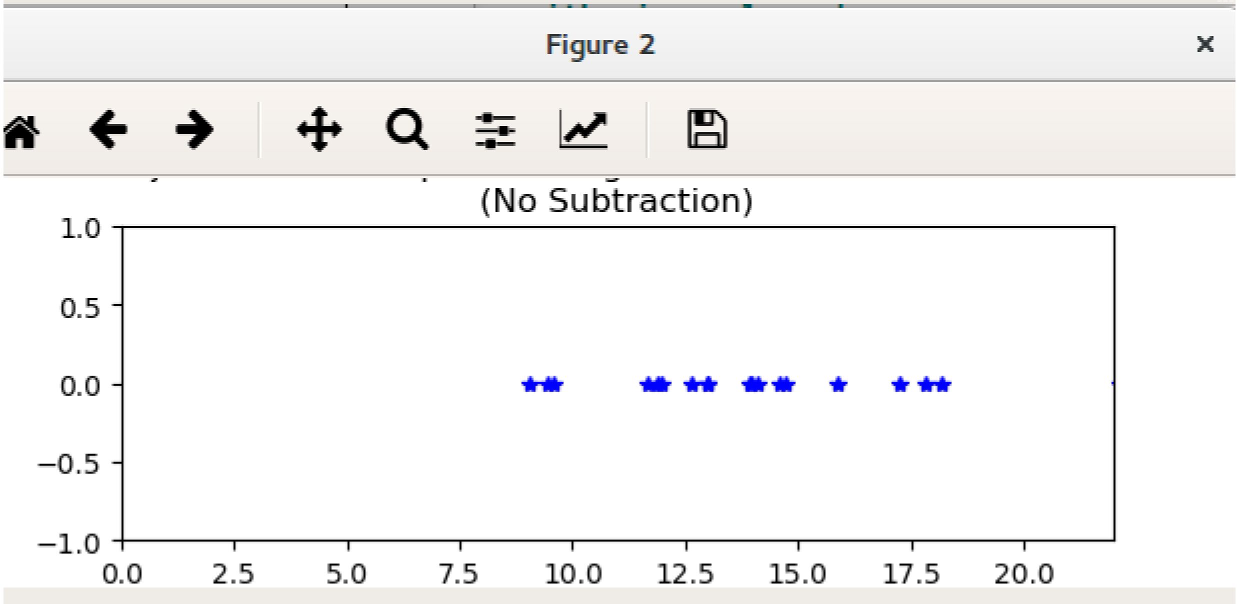
در تصویر زیر نگاشت نقطه‌ها بر روی PCA را مشاهده می‌کنید:



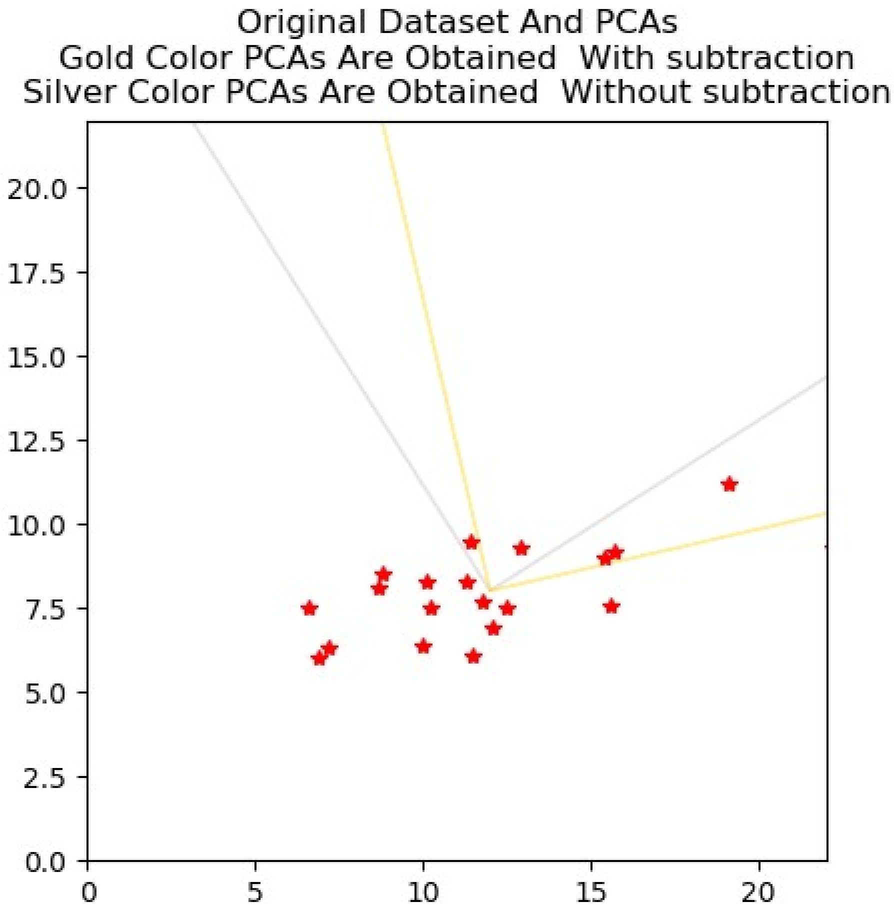
بخش E)

اگر به دو تصویر پروجکت نقاط بر روی PCA با بزرگترین واریانس در حالتی که میانگین را ک م نکرده‌ای م و حالتی که میانگین را کم کرده‌ایم توجه کنیم متوجه می‌شویم آن دو یکسان نیستند.





به همین دلیل به این موضوع به ذهنم رسید که PCA ها در هر دو حالت یکسان نیستند برای تحقیق این موضوع PCA ها را برای هر دو حالت رسم کردم که در تصویر آن را مشاهده می‌کنید:



رنگ طلایی CPA ها در حالتی است که میانگین را کم کرده‌ایم و رنگ نقره‌ای حالتی است ک ه می انگین را کم نکرده‌ایم. همین‌طور که مشاهده می‌شود آن‌ها با هم یکسان نیستند و حالتی کـه میـانگین را کـم نکرده ایم اشتباه است.

سوال 6)

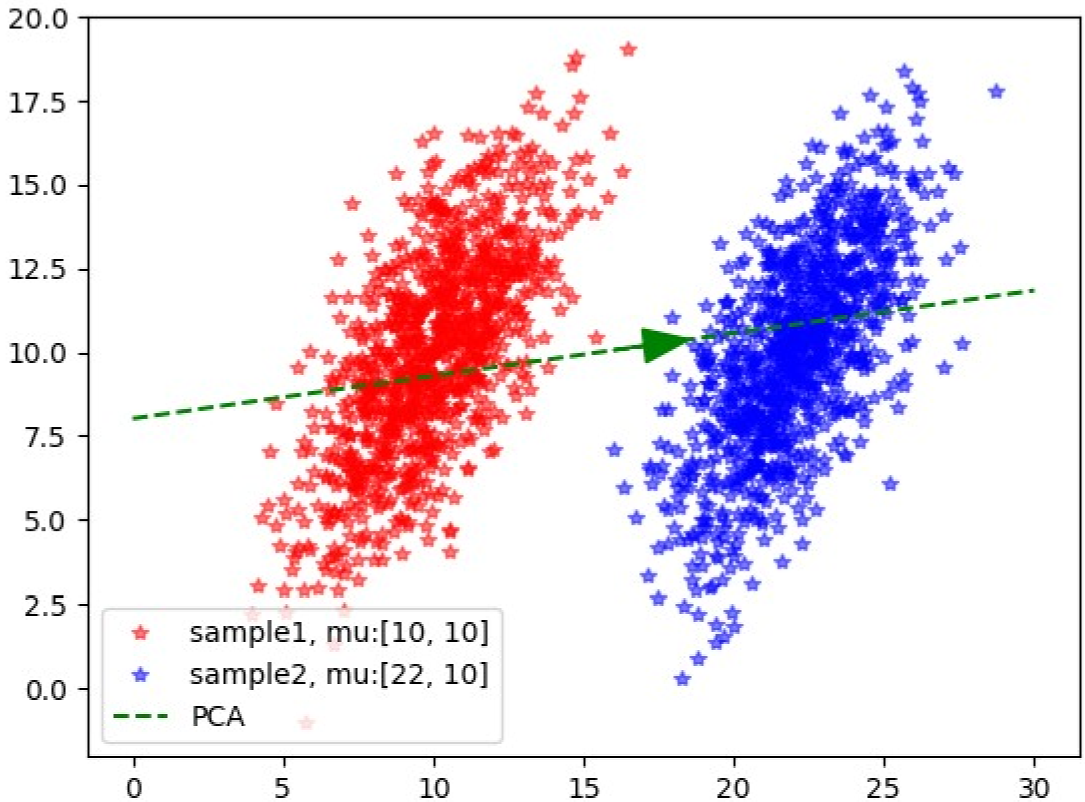
کدهای این بخش در فایل problem6.py موجود است.

بخش a)

ابتدا میانگین داده‌ها و ماتریس observation که میانگین از آن کم شده است را با استفاده از تابع subtract_mean که در سوال‌های قبل آن را معرفی کردم به دست می‌آوریم و سپس با استفاده از تابع calculate_covariace که آن را نیز در سوال‌های قبل شرح دادم، ماتریس کواریانس را به دست می‌آوریم و با استفاده از تابع eigen_value_vector مقدار eigen vectorها و eigen valueها را به دست می‌آوریم که مقدار آن‌ها برابر عداد درون شکل زیر می‌شود:

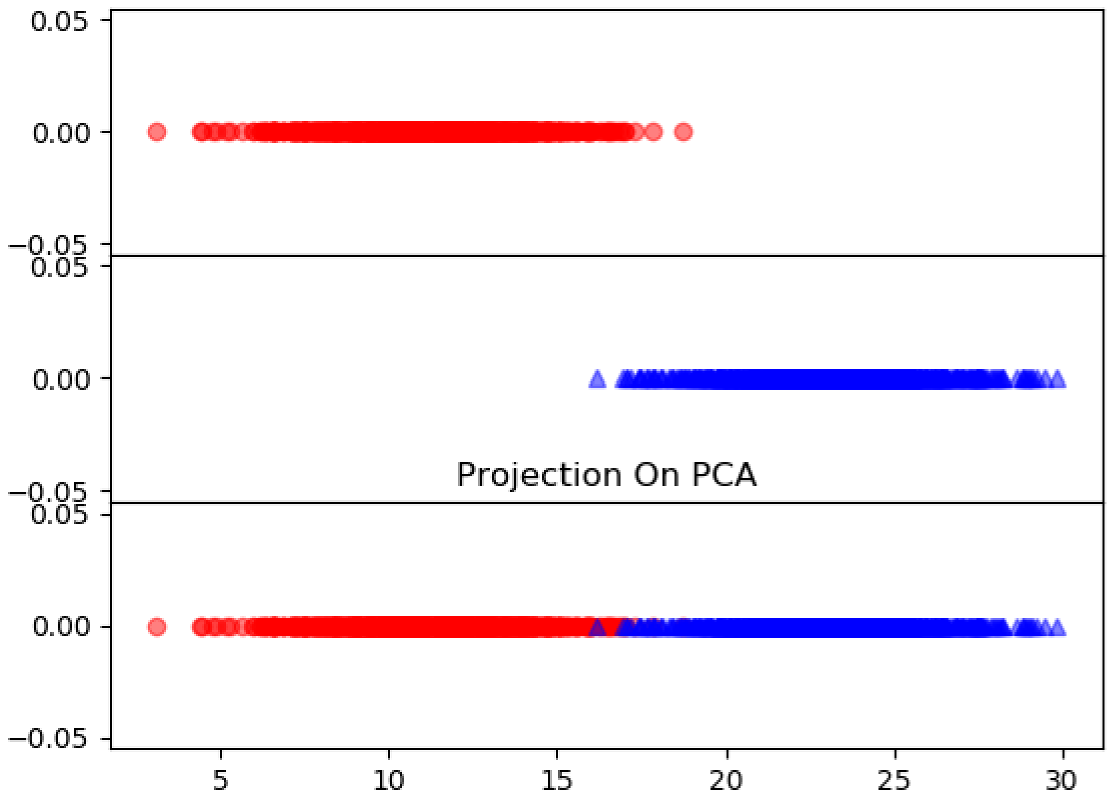
```
Mean Class 1:
 [[ 10.15629987]
 [ 10.18079947]]

Mean Class 2:
 [[ 21.93952692]
 [  9.86188861]]

Mean of all elements:
 [[ 21.93952692]
 [  9.86188861]]


Covariance For PCA:
 [[ 38.74202055   2.95230091]
 [  2.95230091   8.78950728]]
Eigen Values for covariance:
 [ 39.03024371   8.50128412]
Eigen Vectors for covariance:
 [[ 0.99526832 -0.09716468]
 [ 0.09716468  0.99526832]]
Eigen Vector with highest Eigen Value:
 [[ 0.99526832]
 [ 0.09716468]]
```

در تصویر زیر دیتا ست و بردار PCA معادل با بزرگ‌ترین eigen value را مشاهده می‌کنید:



بخش b)

با استفاده از رابطه‌ی y=transpose(u)*x که x سمپل‌ها است و u برابر با eigen vector معادل بزرگ‌ترین eigen value است نقطه‌ها را به فضای eigen vector نگاشت می‌کنیم. بعد از آن داده‌های پروجکت شده را رسم کرده‌ام که در شکل زیر آن را مشاهده می‌کنید. از آن جایی که داده‌ها هم پوشانی دارند ابتدا داده‌های کلاس‌های مختلف را جدا رسم کرده‌ام و سپس آن‌ها را کنار هم رسم کرده‌ام:



بخش C)

همین طور که در شکل قبل قابل مشاهده است، داده‌ها در فضای نگاشت شده هم پوشانی دارند، در حالی که امکان جداپذیر کردن آن‌ها با نگاشت‌های دیگر هست.

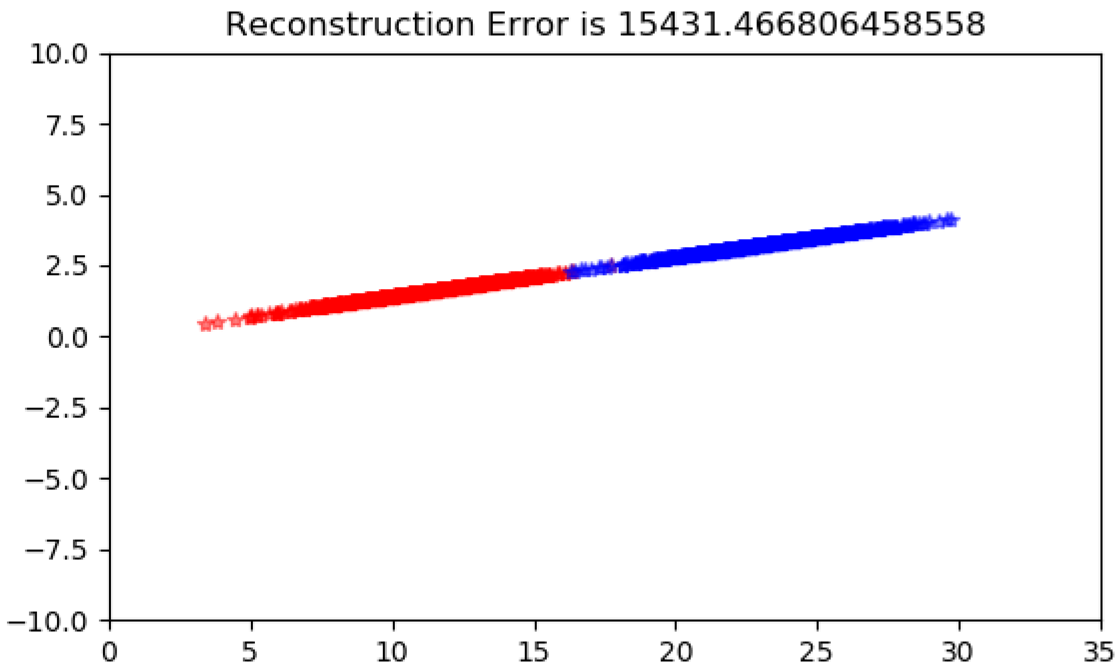
بخش D)

برای بازیابی نقطه‌ها از فضای نگاشت شده به فضای اصلی از رابطه‌ی x=u*y که x سمپل‌ها بازیابی شده از فضای نگشت به فضای اولیه است، u برابر با eigen vector معادل بزرگ ترین eigen value است و y برابر با داده‌ها در فضای نگاشت شده است استفاده می‌کنیم.

برای محاسبه‌ی خطای بازساخت داده‌ها فاصله‌ی هر نقطه‌ی بازیابی شده تا خود نقطه در آن فضا را محاسبه می‌کنیم که مجموع آن فاصله‌ها برابر است با میزان خطای بازیابی. که از آنجایی که داده‌ها را تصادفی ایجاد می‌کنیم میزان خطا در هم بار اجرا کمی تغییر می‌کند به عنوان نمونه در یکی از اجرا ها میزان خطا برابر عدد زیر شد:

```
Reconstruction Error is 15431.466806458558
```

در تصویر زیر داده‌های بازیابی شده را مشاهده می‌کنید:



بخش E)

در این بخش باید LDA را محاسبه کنیم، در کد طبق رابطه‌های زیر LDA به دست آمده است:

میانگین دو کلاس را اینگونه به دست می‌آوریم:

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$$

که برابر می‌شود با:

```
Mean Class 1:
 [[ 10.15629987]
 [ 10.18079947]]

Mean Class 2:
 [[ 21.93952692]
 [  9.86188861]]

Mean of all elements:
 [[ 21.93952692]
 [  9.86188861]]
```

برای محاسبه‌ی Scatter دو کلاس اینگونه عمل می‌کنیم:

$$\mathbf{S}_i = \sum_{\mathbf{x}\in\mathcal{D}_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t$$

که برای کلاس یک و دو برابر می‌شود با

```
Scatter Class 1:
 [[ 4069.41943493  4011.98720707]
 [ 4011.98720707  8622.60378712]]

Scatter Class 2:
 [[ 4170.15746012  3899.63422421]
 [ 3899.63422421  8399.13370491]]
```

و within class scatter را اینگونه محاسبه می‌کنیم:

$$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

که برابر می‌شود با

```
Scatter within(Sw):
 [[  8239.57689506   7911.62143128]
 [  7911.62143128  17021.73749204]]
```

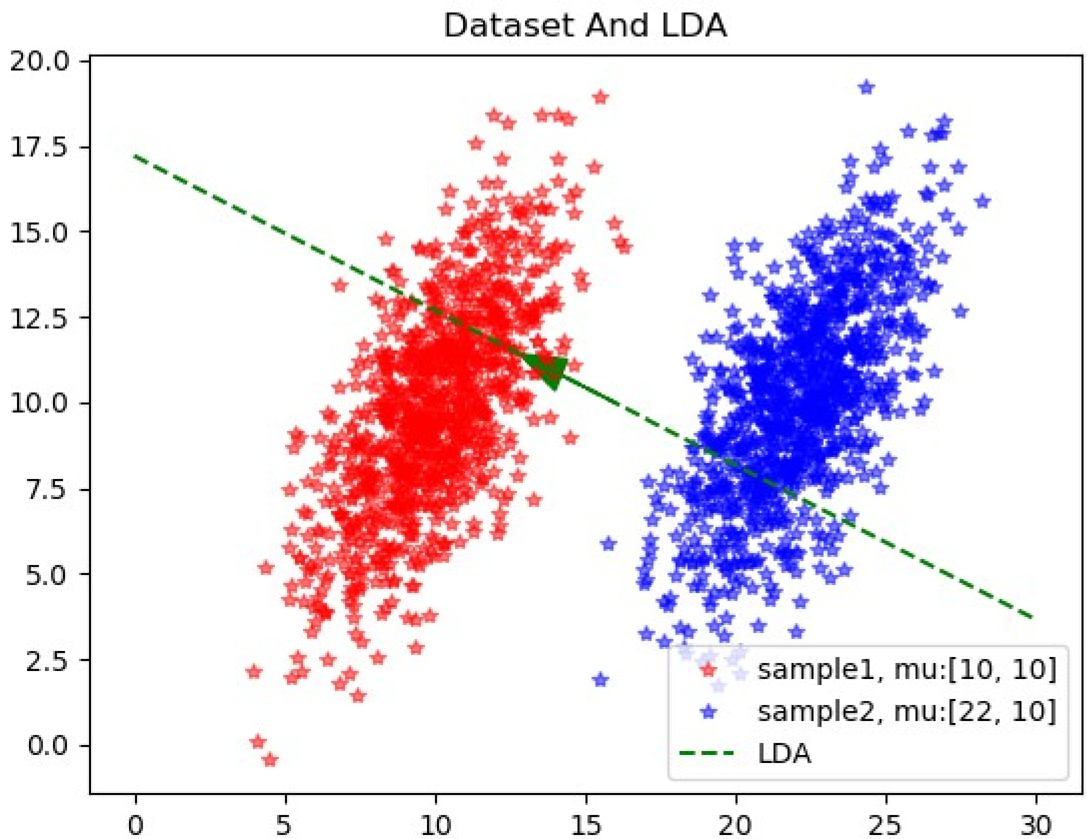
برای محاسبه‌ی بردار w به صورت زیر عمل می‌کنیم:

$$\boxed{\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2).}$$

که برابر می‌شود با

```
W:
 [[-0.00262455]
 [ 0.0012049 ]]
```

در تصویر زیر LDA و داده‌های رسم شده را مشاهده می‌کنید:



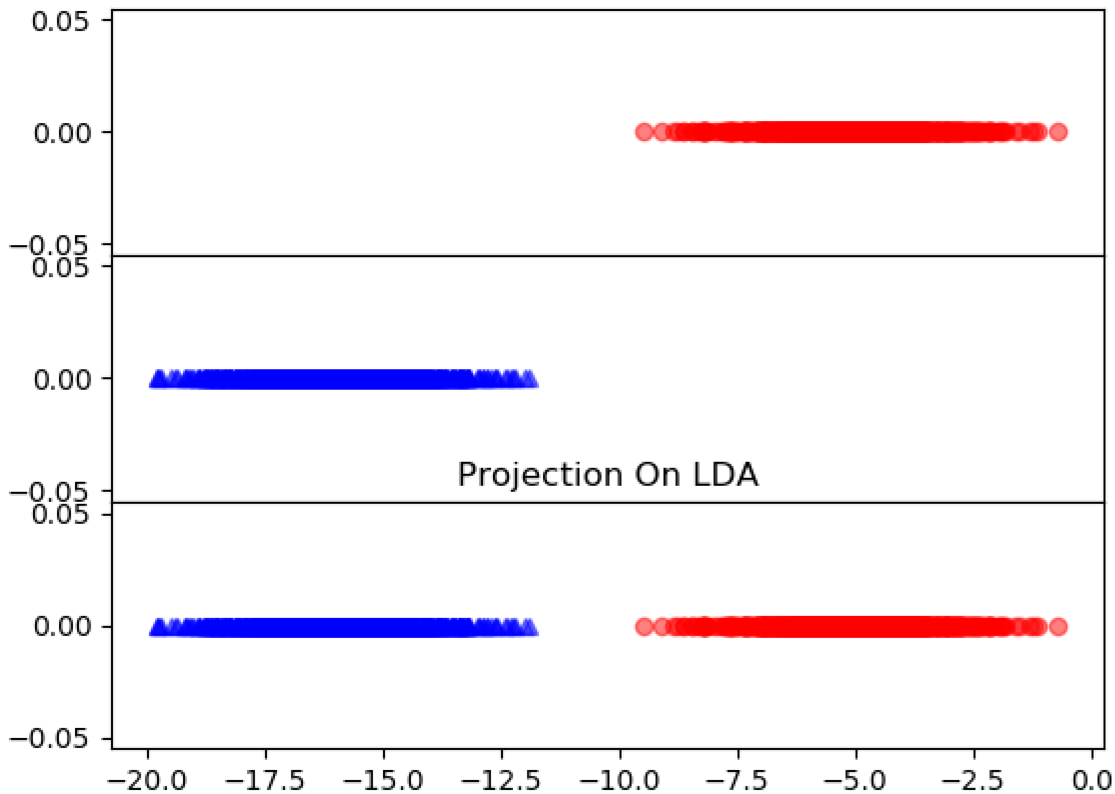
بخش f)

برای نگاشت داده‌ها به W از رابطه‌ی زیر استفاده می‌کنیم:

$$y = \mathbf{w}^t\mathbf{x}$$

در تصویر زیر نقاط پروجک شده را بر LDA را مشاهده می‌کنید که ابتدا داده‌های هر کلاس را جدا رسم

کرده‌ام و سپس آن‌ها را کنار هم رسم کرده‌ام:



بخش G)

همین طور که مشاهده می‌شود LDA با ماکزیمم کردن فاصله‌ی میانگین‌های دو کلاس نسبت به مجموع مربع‌های Scatter آن‌ها، خطی را پیدا کرده است که نگاشت داده‌های دو کلاس بر روی آن کاملا جدا از هم اند.

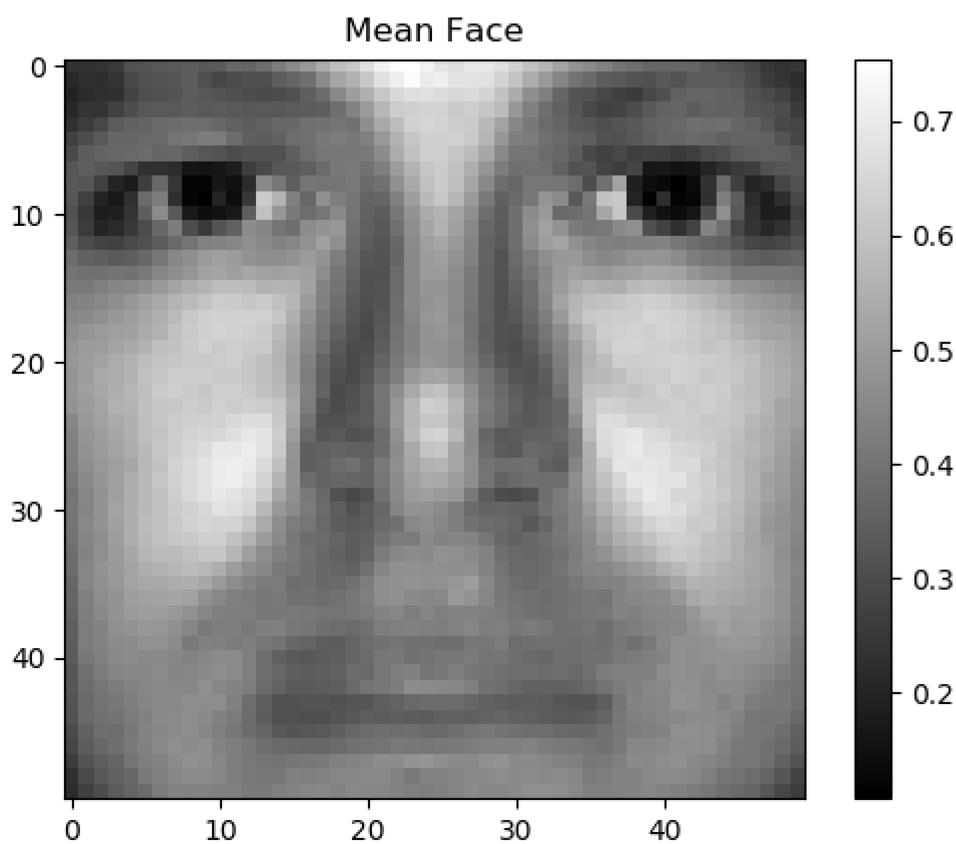
سوال ۷)

کدهای این سوال در `problem7.py` قرار دارد.

بخش a) برای حل این سوال از تابع‌هایی که قبلا نوشته‌ام و در سوال‌های قبل آن‌ها را توضیح داده‌ام استفاده کرده‌ام:

```
subtract_mean
calculate_covariace
eigen_value_vector
```

ابتدا با استفاده از `scipy.io.loadmat` دیتاست را خوانده‌ام. سپس مجموعه‌ی `Train` را تقسیم بر 255 کرده‌ام تا اعداد در بازه‌ی 0 تا 1 قرار بگیرند. سپس با استفاده از `subtract_mean` میانگین صورت‌ها را به دست آورده‌ام و آن را همه‌ی صورت‌ها کم کرده‌ام. در تصویر زیر میانگین صورت‌ها را مشاهده می‌کنید:



از آنجایی که ماتریس `A*Transpose(A)` بسیار بزرگ می‌شود و محاسبه‌ی `eigen value` ها و `eigen vector` ها دشوار می‌شود اینگونه 70 `eigen vector` و `eigen value` اول و بزرگ‌تر را به دست می‌آوریم:

Step 6: compute the eigenvectors u_i of AA^T

The matrix AA^T is very large --> not practical !!

Step 6.1: consider the matrix $A^T A$ (MxM matrix)

Step 6.2: compute the eigenvectors v_i of $A^T A$

$$A^T A v_i = \mu_i v_i$$

What is the relationship between us_i and v_i?

$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow$

$CAv_i = \mu_i A v_i$ or $Cu_i = \mu_i u_i$ where $u_i = Av_i$

Thus, AA^T and $A^T A$ have the same eigenvalues and their eigenvectors are related as follows: $u_i = Av_i$!!

Step 6.3: compute the M best eigenvectors of AA^T: $u_i = Av_i$ (i.e.,

(**important:** normalize u_i such that $\|u_i\| = 1$)

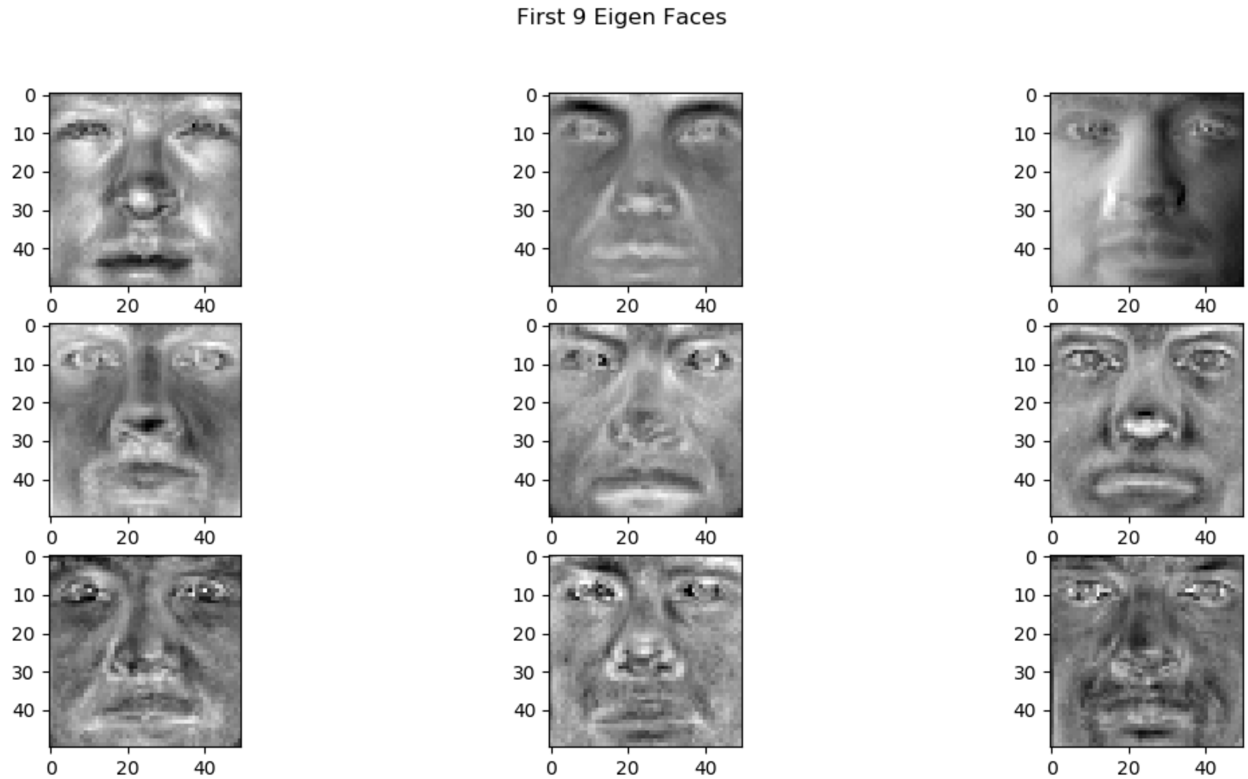
بعد از به دست آوردن eigen value ها و eigen vector ها آن‌ها را بر اساس eigen vector بزرگ به کوچک سورت می‌کنم.

در زیر مقادیر eigen value ها را مشاهده می‌کنید:

```
Number of Eigen Values of faces:
70
Eigen Values of faces:
[  5.73989720e-01   1.70853969e-01   1.16811388e-01   8.41921056e-02
   6.09786536e-02   5.68555579e-02   3.98767493e-02   3.30215061e-02
   2.58698727e-02   2.37555812e-02   1.86943424e-02   6.68788955e-03
   6.16037386e-03   4.75045386e-03   4.30806328e-03   3.94393974e-03
   3.72057066e-03   3.05020619e-03   2.86089760e-03   2.59895220e-03
   2.51250263e-03   2.36429846e-03   2.19658078e-03   2.04648687e-03
   2.00515884e-03   1.84583271e-03   1.79515810e-03   1.71026464e-03
   1.63857680e-03   1.54821251e-03   1.34307354e-03   1.25383675e-03
   1.07451402e-03   1.02523593e-03   9.55463783e-04   8.69886852e-04
   8.33609508e-04   7.86172139e-04   7.26652735e-04   6.99573761e-04
   6.21371377e-04   5.61746739e-04   5.14126148e-04   4.50403318e-04
   3.94875388e-04   3.89212872e-04   3.53439225e-04   3.45487746e-04
   2.89200264e-04   2.76140198e-04   2.67469705e-04   2.62832073e-04
   2.23784422e-04   1.97807274e-04   1.91185360e-04   1.82304714e-04
   1.66680173e-04   1.54472485e-04   1.48269082e-04   1.37638336e-04
   1.27725307e-04   1.14551614e-04   1.09573926e-04   1.08218773e-04
   1.01183224e-04   9.25962975e-05   8.08977100e-05   7.09664641e-05
   6.52030098e-05  -9.88080381e-19]
```

همان‌طور که مشاهده می‌شود در تصویر بالا 70 eigen value وجود دارد.

بعد از آن ۹ eigen vector اول معادل بزرگ‌ترین eigen value ها را رسم کرده‌ام. که در تصویر زیر آن‌ها را مشاهده می‌کنید:



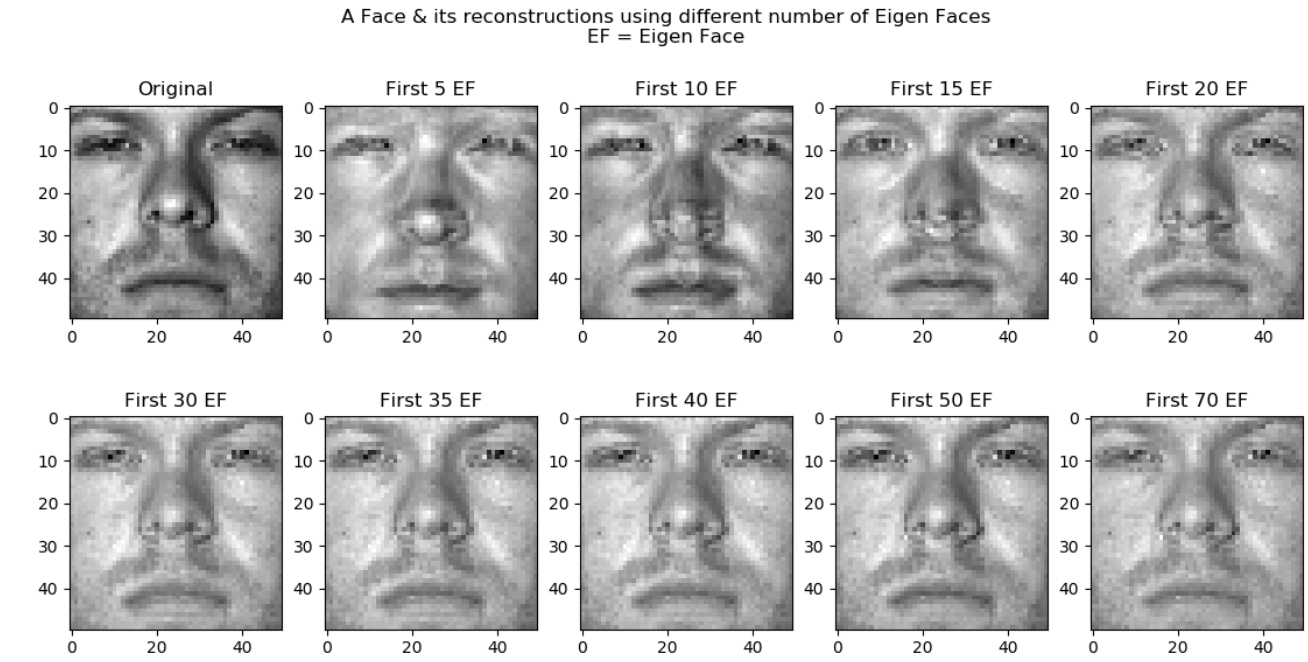
بعد از آن طبق خواسته‌ی سوال یک تصویر را انتخاب کرده‌ام و با انتخاب تعداد متفاوتی از eigen vector ها ابتدا تصویر انتخاب شده را با استفاده از

```
projected-face = transpose(first_m_eigenfaces) * face
```

به فضای آن m آیگن وکتور انتخاب شده برده‌ام و سپس با استفاده از

```
reconstructedface = first_m_eigenfaces * projected-face
```

تصویر را دوباره بازسازی کرده‌ام. در تصویر زیر یک تصویر انتخاب شده را می‌بینید که برای بازسازی آن ابتدا از 5 آیگن وکتور بزرگ‌تر استفاده کرده‌ام و سپس 10 آیگن وکتور اول و بعد 15، 20، 30، 35، 40، 50، 70 :



هر چه از تعداد eigen vector بیشتری استفاده شده است تصویر بازسازی شده به تصویر اصلی نزدیک‌تر شده است.

بخش b)

در این بخش از knn پکیج sklearn برای classification استفاده کرده‌ام. به ازای مقادیر [1, 3, 5, 7, 9] برای k و مقادیر [5, 7, 10, 15, 25, 35, 45, 60] برای m، الگوریتم KNN را با داده‌های train آموزش داده‌ام و بعد مقدار برچسب‌ها را برای داده‌های تست به دست آورده‌ام. قبل از این کارها داده‌های تست و آموزش را به فضای m آیگن وکتور اول برده‌ام:

```
# Project train face on first m PCA
projectedFaceTrain = np.transpose(mEigenFace) * trainFaces
# Project test face on first m PCA
projectedFaceTest = np.transpose(mEigenFace) * trainFaces2
```

در تصویر زیر مقدار خطا برای k و m های مختلف را مشاهده می‌کنید:

```
k=1 , First M EigenVector=5 , Error= 30.0%
k=1 , First M EigenVector=7 , Error= 7.14%
k=1 , First M EigenVector=10    , Error= 0.0%
k=1 , First M EigenVector=15    , Error= 0.0%
k=1 , First M EigenVector=25    , Error= 0.0%
k=1 , First M EigenVector=35    , Error= 0.0%
k=1 , First M EigenVector=45    , Error= 0.0%
k=1 , First M EigenVector=60    , Error= 0.0%
k=3 , First M EigenVector=5 , Error= 30.0%
k=3 , First M EigenVector=7 , Error= 8.57%
k=3 , First M EigenVector=10    , Error= 4.29%
k=3 , First M EigenVector=15    , Error= 0.0%
k=3 , First M EigenVector=25    , Error= 0.0%
k=3 , First M EigenVector=35    , Error= 0.0%
k=3 , First M EigenVector=45    , Error= 0.0%
k=3 , First M EigenVector=60    , Error= 0.0%
k=5 , First M EigenVector=5 , Error= 25.71%
k=5 , First M EigenVector=7 , Error= 10.0%
k=5 , First M EigenVector=10    , Error= 1.43%
k=5 , First M EigenVector=15    , Error= 0.0%
k=5 , First M EigenVector=25    , Error= 0.0%
k=5 , First M EigenVector=35    , Error= 0.0%
k=5 , First M EigenVector=45    , Error= 0.0%
k=5 , First M EigenVector=60    , Error= 0.0%
k=7 , First M EigenVector=5 , Error= 31.43%
k=7 , First M EigenVector=7 , Error= 11.43%
k=7 , First M EigenVector=10    , Error= 5.71%
k=7 , First M EigenVector=15    , Error= 2.86%
k=7 , First M EigenVector=25    , Error= 2.86%
k=7 , First M EigenVector=35    , Error= 2.86%
k=7 , First M EigenVector=45    , Error= 2.86%
```

```
k=7 , First M EigenVector=60    , Error= 2.86%
k=9 , First M EigenVector=5 , Error= 34.29%
k=9 , First M EigenVector=7 , Error= 14.29%
k=9 , First M EigenVector=10    , Error= 8.57%
k=9 , First M EigenVector=15    , Error= 1.43%
k=9 , First M EigenVector=25    , Error= 1.43%
k=9 , First M EigenVector=35    , Error= 1.43%
k=9 , First M EigenVector=45    , Error= 1.43%
k=9 , First M EigenVector=60    , Error= 1.43%
```

از آنجایی که چند مورده خطای مینیمم داریم k=5 و M=۱۵ را انتخاب کردم.

مانند قسمت قبل ابتدا داده‌ها را به فضای PCA نگاشت کرده‌ام و بعد KNN را آموزش داده‌ام و بعد از

تعیین برچسب خطا را محاسبه کردم. خطا برای داده‌های yale3 برابر با عدد زیر شد:

```
Error of Dataset Yale3:
k=5 , First M EigenVector=15     , Error= 24.29%
```