

Лабораторная работа №7 “Метод главных компонент”

Долматович Алина, 858641

Набор данных ex7data1.mat представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит две переменные X1 и X2 - координаты точек, для которых необходимо выделить главные компоненты.

Набор данных ex7faces.mat представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит 5000 изображений 32x32 в оттенках серого. Каждый пиксель представляет собой значение яркости (вещественное число). Каждое изображение сохранено в виде вектора из 1024 элементов. В результате загрузки набора данных должна быть получена матрица 5000x1024.

```
In [1]: 1 from scipy.io import loadmat
        2 import matplotlib.pyplot as pyplot
        3 import numpy as np
        4 from scipy import misc
        5 from mpl_toolkits.mplot3d import Axes3D
        6 import math
```

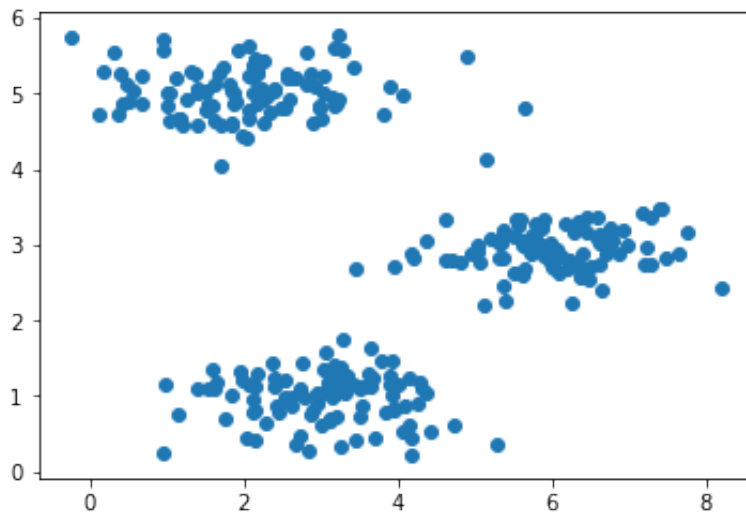
Загрузите данные ex7data1.mat из файла.

```
In [2]: 1 data = loadmat('ex7data1.mat')
        2
        3 x = data['X']
        4 print(x.shape)
```

(300, 2)

Постройте график загруженного набора данных.

```
In [3]: 1 pyplot.scatter(x[:,0], x[:, 1])  
2 pyplot.show()
```



Реализуйте функцию вычисления матрицы ковариации данных.

```
In [4]: 1 def getCovMatrix(x):  
2     return np.cov(x.T)
```

Вычислите координаты собственных векторов для набора данных с помощью сингулярного разложения матрицы ковариации (разрешается использовать библиотечные реализации матричных разложений).

```
In [5]: 1 def getEigenvectors(x):  
2     covMatrix = getCovMatrix(x)  
3     eigenvalues, eigenvectors = np.linalg.eig(covMatrix)  
4     return eigenvectors  
5  
6 eigenvectors = getEigenvectors(x)
```

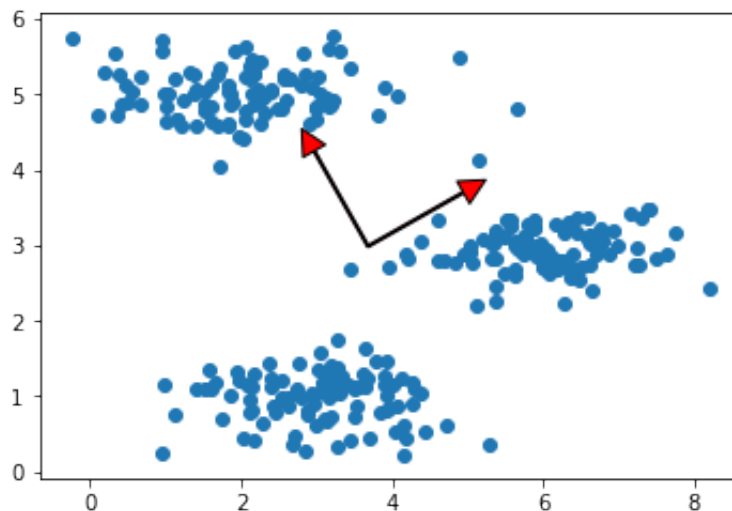
Постройте на графике из пункта 2 собственные векторы матрицы ковариации.

In [6]:

```

1 data = x
2 mu = data.mean(axis=0)
3 data = data - mu
4
5 projected_data = np.dot(data, eigenvectors)
6 sigma = projected_data.std(axis=0).mean()
7
8 fig, ax = pyplot.subplots()
9 ax.scatter(x[:,0], x[:, 1])
10 for axis in eigenvectors:
11     start, end = mu, mu + sigma * axis
12     ax.annotate('', xy=end, xycoords='data',
13                xytext=start, textcoords='data',
14                arrowprops=dict(facecolor='red', width=1.0))
15 ax.set_aspect('equal')
16 pyplot.show()

```



Реализуйте функцию проекции из пространства большей размерности в пространство меньшей размерности с помощью метода главных компонент.

In [7]:

```

1 def dimensionalityReduction(x):
2     eigenvectors = getEigenvectors(x)
3     v = -eigenvectors[:,1]
4     return np.dot(v, x.T), v
5
6 xNew, v = dimensionalityReduction(x)

```

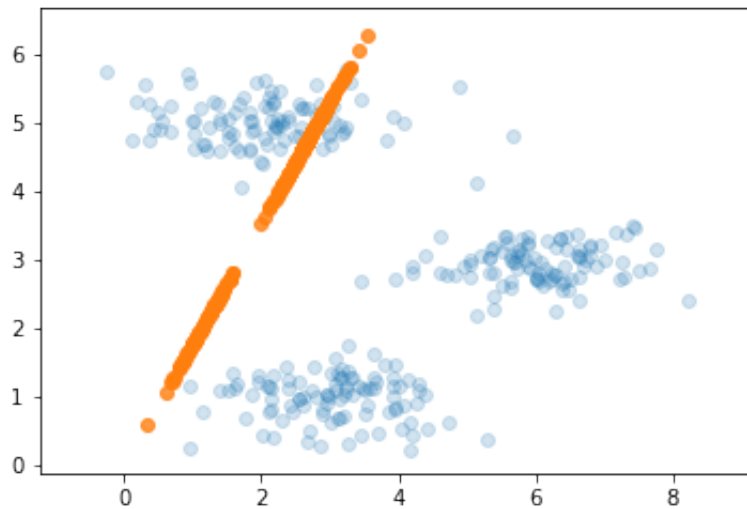
Реализуйте функцию вычисления обратного преобразования.

```
In [8]: 1 def restored(x):
2         xNew, v = dimensionalityReduction(x)
3         xRestored = []
4         for n in range(len(xNew)):
5             xRestored.append(np.dot(xNew[n], v))
6
7         return np.array(xRestored)
8
9 xRestored = restored(x)
10 print(xRestored)
```

```
[ 3.03721473  3.37334738]
[ 2.8086548   4.97453459]
[ 2.91257892  5.15859928]
[ 2.63731896  4.671074   ]
[ 2.6746425   4.7371794   ]
[ 2.7059882   4.79269718]
[ 2.71602545  4.8104746   ]
[ 2.77630483  4.91723811]
[ 2.57021063  4.55221542]
[ 2.8129783   4.98219214]
[ 2.66601398  4.72189704]
[ 2.46995835  4.37465411]
[ 2.66608703  4.72202643]
[ 2.66530379  4.7206392   ]
[ 2.658974    4.70942821]
[ 3.21239712  5.68962074]
[ 2.96897491  5.25848475]
[ 2.3564771   4.17366238]
[ 2.70098904  4.78384294]
[ 2.67433188  4.73662923]]
```

Постройте график исходных точек и их проекций на пространство меньшей размерности (с линиями проекций).

```
In [9]: 1 pyplot.scatter(x[:, 0], x[:, 1], alpha=0.2)
2 pyplot.scatter(xRestored[:, 0], xRestored[:, 1], alpha=0.8)
3 pyplot.axis('equal')
4 pyplot.show()
```



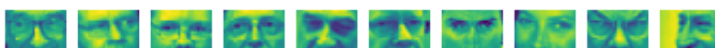
Загрузите данные ex7faces.mat из файла.

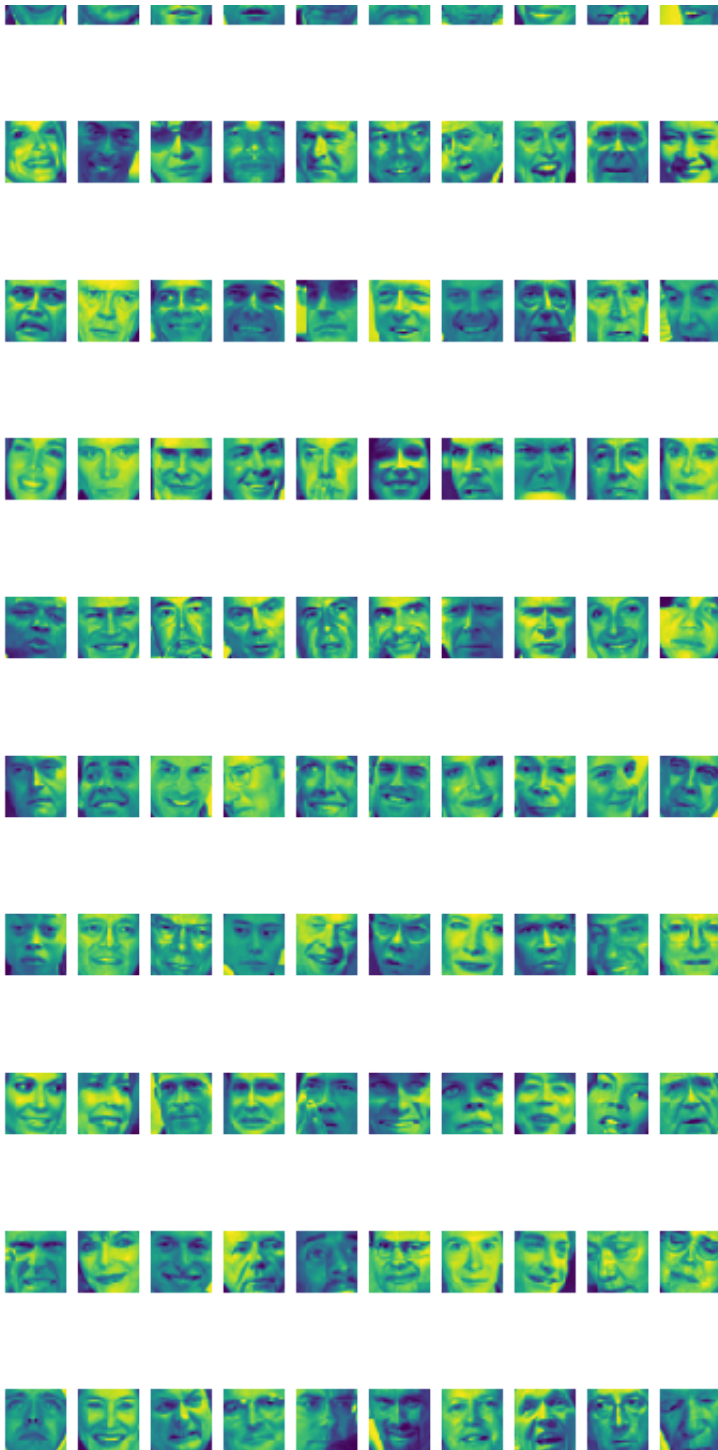
```
In [10]: 1 data = loadmat('ex7faces.mat')
2 x = data['X']
3
4 print(x.shape)
```

(5000, 1024)

Визуализируйте 100 случайных изображений из набора данных.

```
In [11]: 1 def visualization(x, count=100):
2     lines = int(math.ceil(count / 10.0))
3     for i in range(lines):
4         _, axis = pyplot.subplots(1, 10)
5         for j in range(10):
6             index = 10*i + j
7             if index < count:
8                 matrix = x[index].reshape(32, 32, order="F")
9                 axis[j].imshow(matrix)
10                axis[j].axis("off")
11        pyplot.show()
12
13 visualization(x)
```



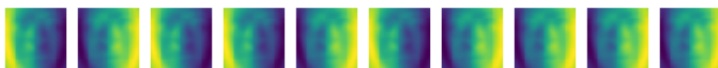
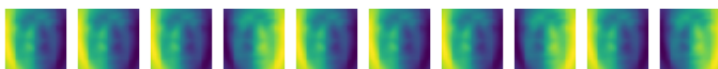
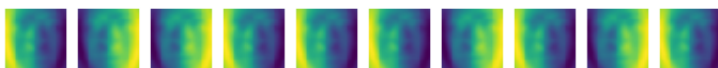
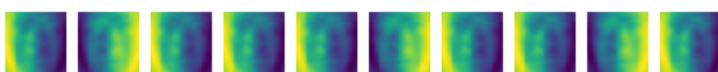


С помощью метода главных компонент вычислите собственные векторы.

```
In [12]: 1 eigenvectors = getEigenvectors(x)
          2 print(eigenvectors.shape)
          3
          (1024, 1024)
```

Визуализируйте 36 главных компонент с наибольшей дисперсией. Визуализируйте 100 главных компонент с наибольшей дисперсией. Как изменилось качество выбранных изображений?

```
In [13]: 1 def imagesVisualization(x, count):  
2     xRestored = restored(x)  
3     dispersions = np.array([np.var(value) for value in xRestored])  
4     indexes = dispersions.argsort()[-count:][::-1]  
5     values = np.array([xRestored[index] for index in indexes])  
6     visualization(values, count)  
7  
8 imagesVisualization(x, 50)
```



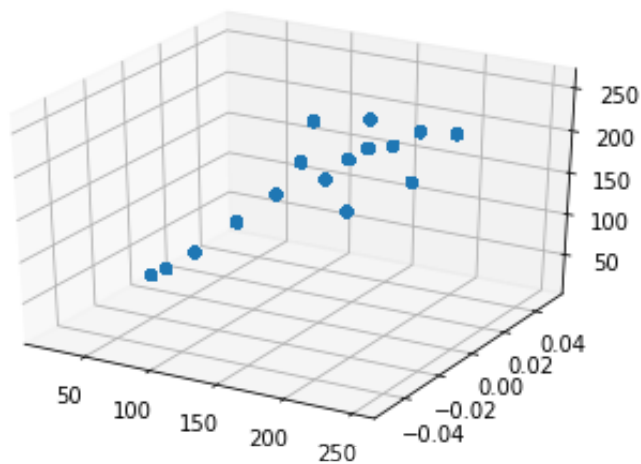
Используйте изображение, сжатое в лабораторной работе №6 (Кластеризация).

```
In [14]: 1 compressedImage = misc.imread('compressedImage.png')
2 print(compressedImage.shape)
3
4 compressedImage = compressedImage.reshape(16384, 3)
5 print(compressedImage)
6
```

```
(128, 128, 3)
[[221 186  0]
 [221 186  0]
 [221 186  0]
 ...,
 [ 58  55  0]
 [ 58  55  0]
 [ 58  55  0]]
```

С помощью метода главных компонент визуализируйте данное изображение в 3D и 2D.

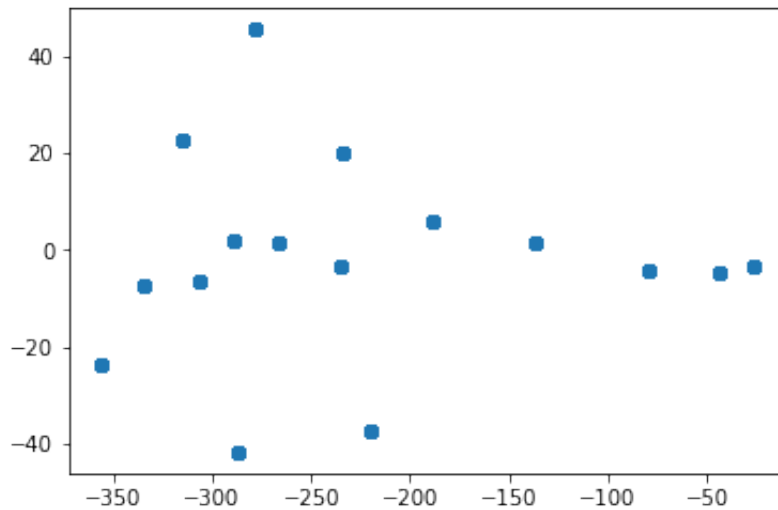
```
In [15]: 1 fig = pyplot.figure()
2 ax = fig.add_subplot(111, projection='3d')
3 ax.scatter(compressedImage[:, 1], compressedImage[:, 2], compressedImage[:, 0])
4
5 pyplot.show()
```



```
In [16]: 1 def dimensionalityReduction(x):
2     m = x.shape[1]
3     eigenvectors = getEigenvectors(x)
4     v = -eigenvectors[:, :m-1]
5     return np.dot(v.T, x.T), v
```

Соответствует ли 2D изображение какой-либо из проекций в 3D?


```
In [17]: 1 xNew, v = dimensionalityReduction(compressedImage)
          2 pyplot.scatter(xNew[0], xNew[1])
          3 pyplot.show()
```



Вывод

Метод главных компонент — один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации. Применяется во многих областях, в том числе, в эконометрике, биоинформатике, обработке изображений, для сжатия данных, в общественных науках.

Вычисление главных компонент может быть сведено к вычислению сингулярного разложения матрицы данных или к вычислению собственных векторов и собственных значений ковариационной матрицы исходных данных.