Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет информатики и радиоэлектроники”

Факультет компьютерных систем и сетей

Специальность “Обработка больших объемов информации”

Лабораторная работа №7

**Метод главных компонент**

Выполнил:

магистрант гр. 858641 Кальман В.А.

Проверил:

Стержанов М. В.

Минск 2019

**Данные.**

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.

**1. Загрузите данные ex7data1.mat из файла.**

ex7data1 = io.loadmat('Data/Lab 7/ex7data1.mat')

X = ex7data1['X']

**2. Постройте график загруженного набора данных.**

plt.figure(figsize=(16,10))

plt.scatter(X[:,0], X[:,1], color='b')

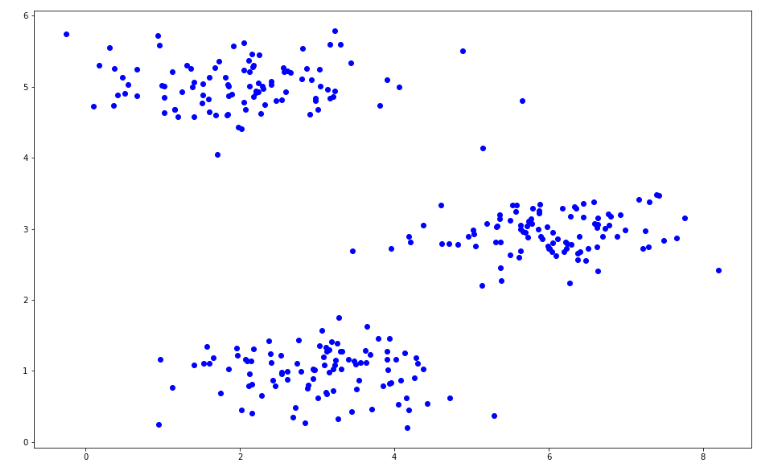


Рисунок 1 – визуализация исходных данных

**3. Реализуйте функцию вычисления матрицы ковариации данных.**

**def** get\_covariance\_matrix(A):

**return** (1.0/A.shape[0]) \* (A.T).dot(A)

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

**def** feature\_normalize(X):

mu = np.mean(X, axis=0)

X\_norm = X - mu

sigma = np.std(X\_norm, axis=0)

X\_norm = X\_norm/sigma

**return** X\_norm, mu, sigma

X\_norm, mu, sigma = feature\_normalize(X)

C = get\_covariance\_matrix(X\_norm)

U, S, Vh = linalg.svd(C)

print('собственные вектора: **\n**', U)

Результат выполнения:

[[17.26276267 20.82286988]

[20.82286988 26.05448259]]

**5. Постройте на графике из пункта 2 собственные векторы матрицы ковариации.**

**def** draw\_line(p1, p2, \*\*kwargs):

plt.plot(np.array([p1[0], p2[0]]), np.array([p1[1], p2[1]]), \*\*kwargs)

plt.figure(figsize=(16,10))

plt.scatter(X[:,0], X[:,1], color='b')

draw\_line(mu, mu + 1.5 \* S[0] \* U[:, 0], c='red', linewidth=2)

draw\_line(mu, mu + 1.5 \* S[1] \* U[:, 1], c='red', linewidth=2)

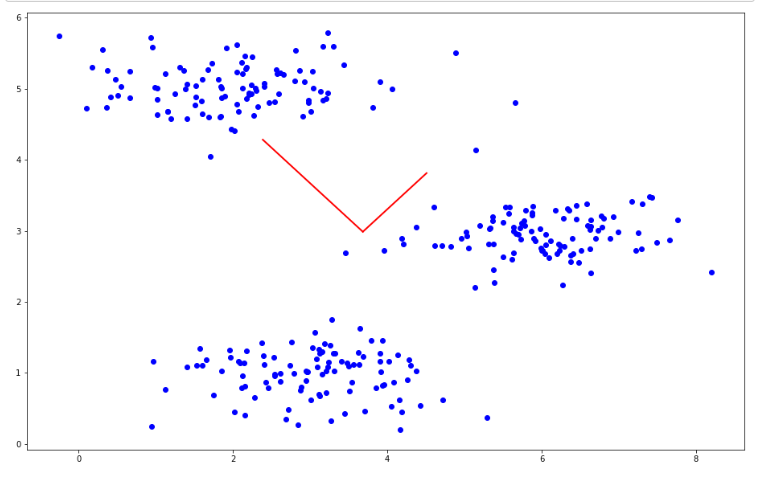


Рисунок 2 – собственные векторы матрицы ковариации

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

# Project the data onto K = 1 dimension

K = 1

Z = project\_data(X\_norm, U, K)

print('Projection of the first example: {:.6f}'.format(Z[0, 0]))

print('(this value should be about : 1.481274)')

Результат выполнения:

Projection of the first example: 1.496313

(this value should be about : 1.481274)

**7. Реализуйте функцию вычисления обратного преобразования.**

X\_rec = recover\_data(Z, U, K)

print('Approximation of the first example: [{:.6f} {:.6f}]'.format(X\_rec[0, 0], X\_rec[0, 1]))

print(' (this value should be about [-1.047419 -1.047419])')

Результат выполнения:

Approximation of the first example: [-1.058053 -1.058053]

(this value should be about [-1.047419 -1.047419])

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

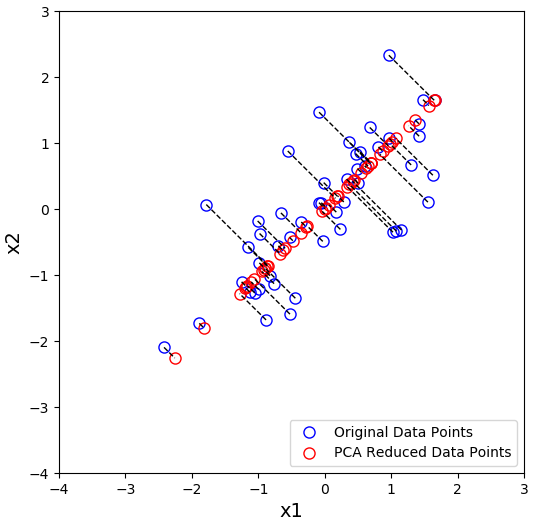


Рисунок 3 – график исходных точек и их проекции на пространство меньшей размерности

**9. Загрузите данные ex7faces.mat из файла.**

ex7faces = io.loadmat('Data/Lab 7/ex7faces.mat')

X = ex7faces['X']

**10. Визуализируйте 100 случайных изображений из набора данных.**

**def** show\_images(X, img\_size=32):

img\_in\_row = 10

img\_in\_col = int(100 / img\_in\_row)

f, axarr = plt.subplots(img\_in\_col, img\_in\_row,

figsize=(32,32))

row\_ind = -1

**for** i, ind **in** enumerate(np.random.randint(len(X), size=100)):

**if** i % img\_in\_row == 0:

row\_ind += 1

col\_ind = i - row\_ind \* img\_in\_row

axarr[row\_ind, col\_ind].imshow(X[ind].reshape(img\_size,img\_size), cmap='gray')

axarr[row\_ind, col\_ind].axis('off')

show\_images(X)



Рисунок 4 – визуализация 100 случайных изображений из набора данных

**11. С помощью метода главных компонент вычислите собственные векторы.**

means, stds, X\_norm = feature\_normalize(X)

U, S = pca(X\_norm)

**12. Визуализируйте 36 главных компонент с наибольшей дисперсией.**

Z\_36 = project\_data(X\_norm, U, 36)

show\_images(Z\_36, 6)



Рисунок 5 – 36 главных компонент с наибольшей дисперсией

**13.Как изменилось качество выбранных изображений?**

Eigen vectors с наибольшей дисперсией охватывают самые базовые черты. Так как большая часть деталей аппроксимируется. При уменьшении дисперсии количество деталей увеличивается.

**14. Визуализируйте 100 главных компонент с наибольшей дисперсией.**

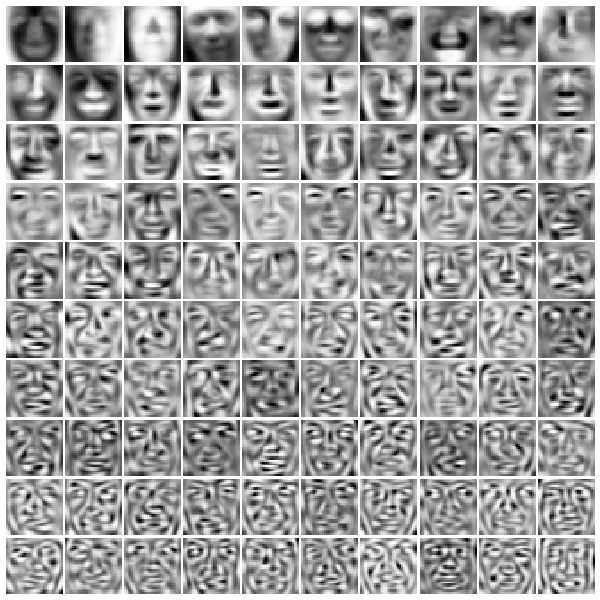


Рисунок 6 – 100 главных компонент с наибольшей дисперсией

**15. Как изменилось качество выбранных изображений?**

Изображения становятся сложнее (больше деталей).

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

car = cv2.imread('car\_kmeans.png')

car = cv2.cvtColor(car, cv2.COLOR\_BGR2RGB)

pixels = np.unique(car.reshape(-1, 3), axis=0)

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

fig = plt.figure(figsize=(10,6))

ax\_1 = fig.add\_subplot(111, projection='3d')

ax\_1.scatter(pixels[:, 0], pixels[:, 2], pixels[:, 1])

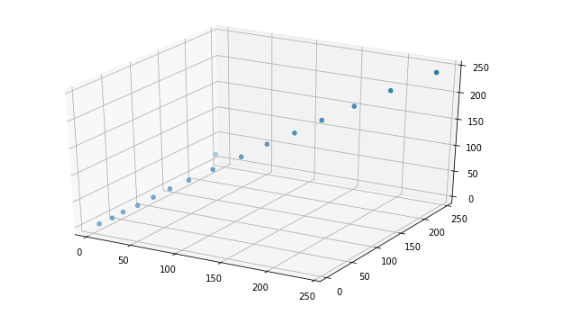


Рисунок 7 – визуализация пикселей изображения и их кластеров в 3D

X\_norm, mu, sigma = feature\_normalize(pixels)

C = get\_covariance\_matrix(X\_norm)

U, S, Vh = linalg.svd(C)

pixels2d = project\_data(X\_norm, U, 2)

fig = plt.figure(figsize=(16,10))

plt.scatter(pixels2d[:, 0], pixels2d[:, 1])

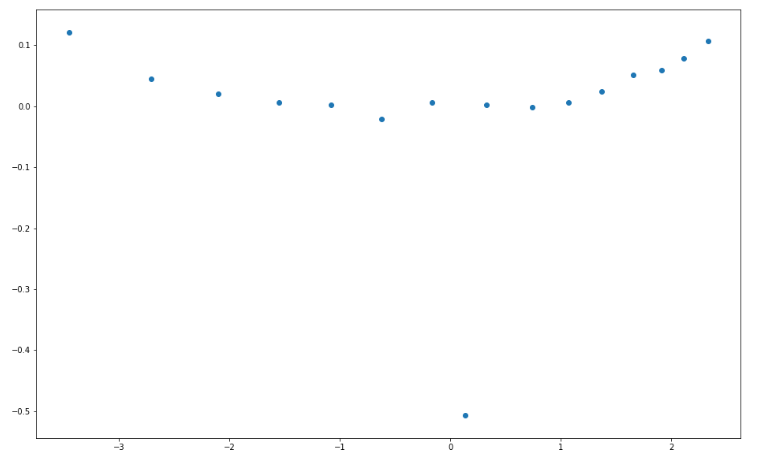


Рисунок 8 – визуализация пикселей изображения и их кластеров в 2D, с помощью PCA

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

2D изображение соответсвует “лучшей” проекции 3D изображения на двумерную плоскость.

**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.