Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

ЛабоРАТОРНАЯ РАБОТА №7

«Метод главных компонент»

Выполнил: Шпаковская Валерия

магистрант кафедры информатики

группа №858641

Проверил: доцент, кандидат технических наук Стержанов Максим Валерьевич

Минск 2019

ХОД РАБОТЫ

**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.

**Выполнение:**

1. Загрузите данные ex7data1.mat из файла:

file\_path = 'ex7data1.mat'

dataset = sio.loadmat(file\_path)

X = dataset["X"]

1. Постройте график загруженного набора данных:

# X[:, 0] - first column

plt.scatter(X[:, 0], X[:, 1], marker="o")

plt.show()

Результат выполнения:

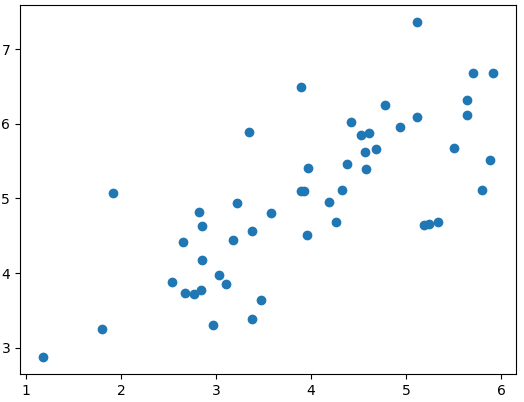


Рисунок 1 – визуализация исходных данных

3. Реализуйте функцию вычисления матрицы ковариации данных:

def compute\_covariance\_matrix(X):

return X.T.dot(X) / X.shape[0]

covariance\_matrix = compute\_covariance\_matrix(X)

print(covariance\_matrix)

Результат выполнения:

[[17.26276267 20.82286988]

[20.82286988 26.05448259]]

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

def feature\_normalize(X):

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

X\_norm = X - means

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

X\_norm = X\_norm / stds

return means, stds, X\_norm

def pca(X):

covariance\_matrix = compute\_covariance\_matrix(X)

U, S, V = svd(covariance\_matrix, full\_matrices=True, compute\_uv=True)

return U, S

# Feature normalize

# mu, sigma

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

# Run SVD

U, S = pca(X\_norm)

print(U, S)

Результат выполнения:

(array([[-0.70710678, -0.70710678],

[-0.70710678, 0.70710678]]), array([1.73553038, 0.26446962]))

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

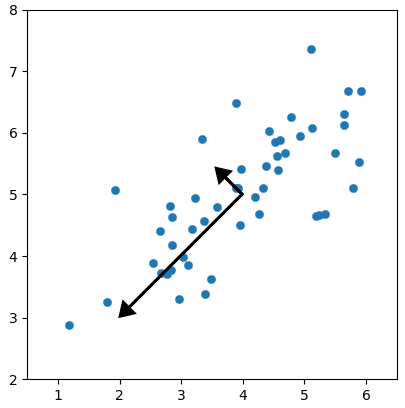


Рисунок 2 – собственные векторы матрицы ковариации

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):

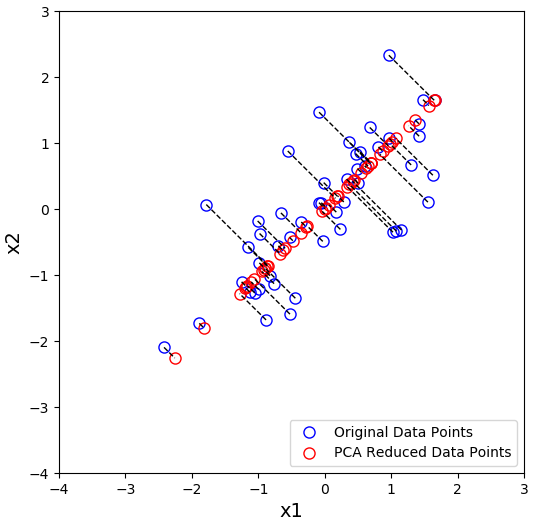


Рисунок 3 – график исходных точек и их проекции на пространство меньшей размерности

9. Загрузите данные ex7faces.mat из файла:

file\_path = 'ex7faces.mat'

dataset = sio.loadmat(file\_path)

X = dataset['X']

10. Визуализируйте 100 случайных изображений из набора данных:



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:



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:

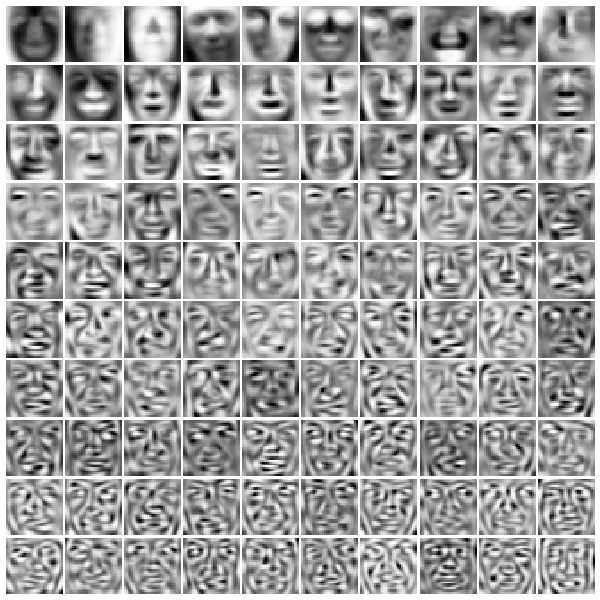


Рисунок 6 – 100 главных компонент с наибольшей дисперсией

15. Как изменилось качество выбранных изображений?

Изображения становятся сложнее (больше деталей).

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

A = img.imread('output.jpg')

X = A.reshape(-1, 3)

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

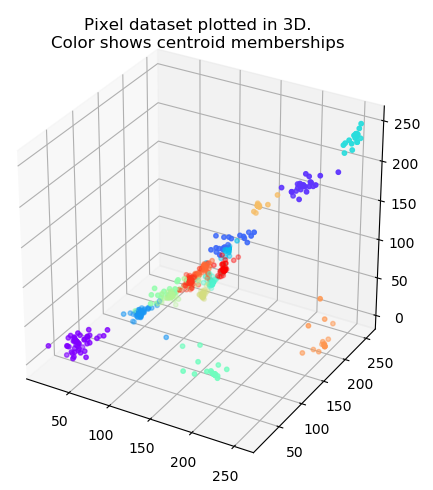


Рисунок 7 – визуализация пикселей изображения и их кластеров в 3D

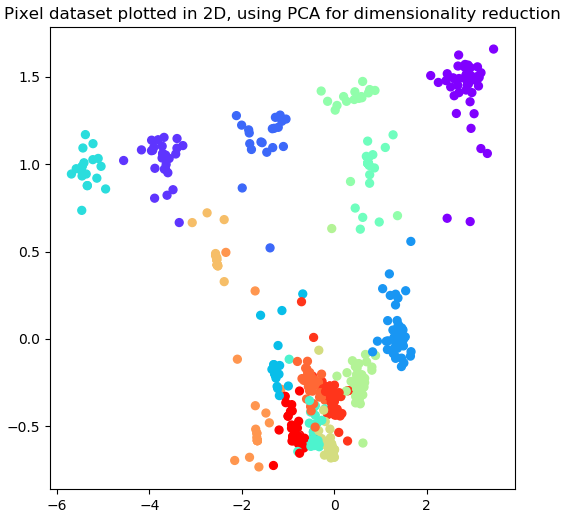


Рисунок 8 – визуализация пикселей изображения и их кластеров в 2D, с помощью PCA

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

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