МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по практическому заданию №2

по дисциплине «Машинное обучение»

Студент гр. 6304 _____ Ковынев М.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Задание

Задание 1

Рассмотрим данные

i	x_i
x_1	(4, 2.9)
x_2	(2.5, 1)
x_3	(3.5, 4)
x_4	(2, 2.1)

Есть ядро (функция сходства):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

Рассчитайте ядерную матрицу

Задание 2

Рассмотрим данные в виде матрицы D:

$\boldsymbol{X_1}$	$\boldsymbol{X_2}$
8	-20
0	-1
10	-19
10	-20
2	0

1. Рассчитайте среднее $\boldsymbol{\mu}$ и ковариационную матрицу $\boldsymbol{\Sigma}$ для матрицы D
2. Рассчитайте собственные числа для матрицы $\boldsymbol{\Sigma}$
3. Какой “внутренний” размер данного набора данных?
4. Рассчитай первый главный компонент

5. Если **μ** и **Σ** сверху характеризуют нормальное распределение, из которого были сгенерированы точки, нарисуйте ориентацию / протяженность 2-мерной функции нормальной плотности.

Задание 3

Для данных и ядра из первого задания найдите первую главную компоненту при нелинейном преобразовании для заданного ядра

Ход работы

1. Рассчитайте ядерную матрицу

```
import numpy as np
from matplotlib import cm
from scipy import stats
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
from sklearn.decomposition import KernelPCA

X = np.array([[4, 2.9], [2.5, 1], [3.5, 4], [2, 2.1]])

func = lambda xi, xj: np.linalg.norm(xi - xj) ** 2
matrix = [[func(xi, xj) for xi in X] for xj in X]

for row in matrix:
    print([round(x, 3) for x in row])
```

```
[0.0, 5.86, 1.46, 4.64]
[5.86, 0.0, 10.0, 1.46]
[1.46, 10.0, 0.0, 5.86]
[4.64, 1.46, 5.86, 0.0]
```

2. Рассчитайте среднее **μ** и ковариационную матрицу **Σ** для матрицы D

```
D = np.array([[8, -20], [0, -1], [10, -19], [10, -20], [2, 0]])
print('Mean:', np.mean(D, axis=0))

cov_m = np.cov(D, rowvar=False)
print('Cov:\n', cov_m)
```

```
Mean: [  6. -12.]
Cov:
 [[ 22.  -47.5]
 [-47.5 110.5]]
```

3. Рассчитайте собственные числа для матрицы **Σ**

```
cov_m = np.cov(D, rowvar=False)
eigvals = np.linalg.eig(cov_m)[0]
eigvecs =  np.linalg.eig(cov_m)[1]
print('Eig vals:',eigvals)
print('Eig vecs:\n',eigvecs)
```

```
Eig vals: [  1.332 131.168]
Eig vecs:
 [[-0.917  0.399]
 [-0.399 -0.917]]
```

4. Какой “внутренний” размер данного набора данных?

```
print('Size:', len(D), 'x',len(D[0]))
```

```
Size: 5 x 2
```

5. Рассчитай первый главный компонент

```
D_centered = D - np.mean(D, axis=0)
projection = -eigvecs[:, np.argmax(eigvals)]
print('First pc:\n', np.dot(D_centered, projection))
```

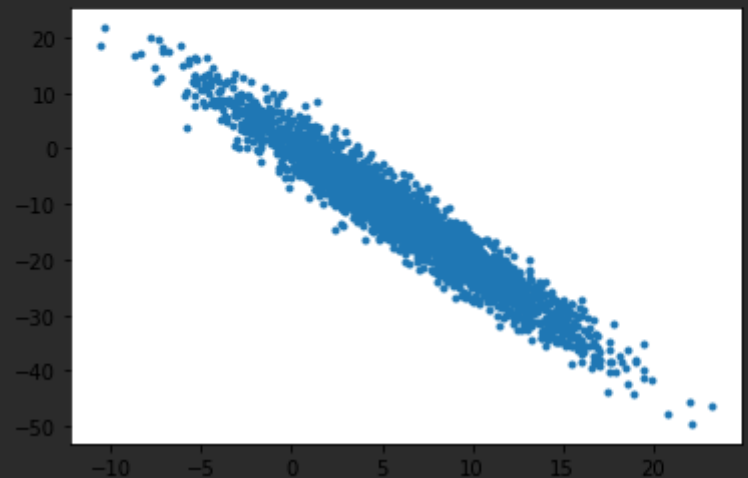
```
First pc:
 [-8.134 12.48  -8.015 -8.932 12.599]
```

6. Если **μ** и **Σ** сверху характеризуют нормальное распределение, из которого были сгенерированы точки, нарисуйте ориентацию / протяженность 2-мерной функции нормальной плотности.

```
mean = np.mean(D, axis=0)
cov_m = np.cov(D, rowvar=False)

rv = np.random.multivariate_normal(mean, cov_m, 3000)
plt.plot(rv[:,0], rv[:,1], '.')
```

```
[<matplotlib.lines.Line2D at 0x1f60fdce348>]
```



7. Для данных и ядра из первого задания найдите первую главную компоненту при нелинейном преобразовании для заданного ядра

```
alpha, beta = 2, 5
tranform_m = np.array(matrix) * alpha + np.ones((len(X), len(X))) * beta
print(KernelPCA(1, 'precomputed').fit_transform(tranform_m@matrix@tranform_m))
```

```
[[ 10.254]
 [-10.254]
 [-10.254]
 [ 10.254]]
```