



Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
«Московский государственный технический университет имени  
Н.Э. Баумана (национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»  
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

## ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ по дисциплине «Вычислительная математика»

Студент:	Магомедов Зайнутдин Арсланович
Группа:	РК6-54Б
Тип задания:	лабораторная работа
Тема:	Спектральное и сингулярное разложение

Студент

\_\_\_\_\_  
подпись, дата

Магомедов З.А.,  
\_\_\_\_\_  
Фамилия, И.О.

Преподаватель

\_\_\_\_\_  
подпись, дата

Соколов А.П.  
\_\_\_\_\_  
Фамилия, И.О.

Москва, 2021

# Содержание

<b>Спектральное и сингулярное разложения</b>	<b>3</b>
1    Задание . . . . .	3
2    Цель выполнения лабораторной работы . . . . .	5
3    Выполненные задачи . . . . .	5
Реализация функции метода главных компонент <i>pca()</i> . . . . .	5
Нахождение главных компонент заданного набора данных. Вывод на экран стандартных отклонений . . . . .	7
Демонстрация достаточности первых двух главных компонент для сепарации типов опухолей . . . . .	9
Построение лапласианов . . . . .	10
Доказательство положительной полуопределенности лапласиана . . . . .	13
Поиск спектра графа . . . . .	15
Поиск числа кластеров в третьем графе . . . . .	17
4    Заключение . . . . .	18

# Спектральное и сингулярное разложения

## 1 Задание

Спектральное разложение (разложение на собственные числа и вектора) и сингулярное разложение, то есть обобщение первого на прямоугольные матрицы, играют настолько важную роль в прикладной линейной алгебре, что тяжело придумать область, где одновременно используются матрицы и не используются указанные разложения в том или ином контексте. В базовой части лабораторной работы мы рассмотрим метод главных компонент (англ. Principal Component Analysis, PCA), без преувеличения самый популярный метод для понижения размерности данных, основой которого является сингулярное разложение. В продвинутой части мы рассмотрим куда менее очевидное применение разложений, а именно одну из классических задач спектральной теории графов – задачу разделения графа на сильно связанные компоненты (кластеризация на графе).

### Задача 1 (спектральное и сингулярное разложения)

Требуется (базовая часть):

1. Написать функцию **pca(A)**, принимающую на вход прямоугольную матрицу данных  $A$  и возвращающую список главных компонент и список соответствующих стандартных отклонений.
2. Скачать набор данных Breast Cancer Wisconsin Dataset: <https://archrk6.bmstu.ru/index.php/f/854843>  
- Указанный датасет хранит данные 569 пациентов с опухолью, которых обследовали на предмет наличия рака молочной железы. В каждом обследовании опухоль была проклассифицирована экспертами как доброкачественная (benign, 357 пациентов) или злокачественная (malignant, 212 пациентов) на основе детального исследования снимков и анализов. Дополнительно на основе снимков был автоматически выявлен и задокументирован ряд характеристик опухолей: радиус, площадь, фрактальная размерность и так далее (всего 30 характеристик). Постановку диагноза можно автоматизировать, если удастся создать алгоритм, классифицирующий опухоли исключительно на основе этих автоматически получаемых характеристик. Указанный файл является таблицей, где отдельная строка соответствует отдельному пациенту. Первый элемент в строке обозначает ID пациента, второй элемент – диагноз (M = malignant, B = benign), и оставшиеся 30 элемент соответствуют характеристикам опухоли(их детальное описание находится в файле <https://archrk6.bmstu.ru/index.php/f/854842>).
3. Найти главные компоненты указанного набора данных, используя функцию **pca(A)**.
4. Вывести на экран стандартные отклонения, соответствующие номерам главных компонент.

5. Продемонстрировать, что проекций на первые две главные компоненты достаточно для того, чтобы произвести сепарацию типов опухолей (доброкачественная и злокачественная) для подавляющего их большинства. Для этого необходимо вывести на экран проекции каждой из точек на экран, используя scatter plot.
6. Опциональное задание №1. Постройте классификатор в полученном пространстве пониженной размерности, используя любой из классических алгоритмов машинного обучения (например, логистическую регрессию или метод опорных векторов) и, при необходимости, кроссвалидацию.

Требуется (продвинутая часть):

1. Построить лапласианы (матрицы Кирхгофа)  $L$  для трех графов:
  - полный граф  $G_1$ , имеющий 10 узлов;
  - граф  $G_2$ , изображённый на рисунке 1;
  - граф  $G_3$ , матрица смежности которого хранится в файле: <https://archrk6.bmstu.ru/index.php/f/854844>, где лапласианом графа называется матрица  $L = D - A$ , где  $A$  – матрица смежности и  $D$  – матрица, на главной диагонали которой расположены степени вершин графа, а остальные элементы равны нулю.

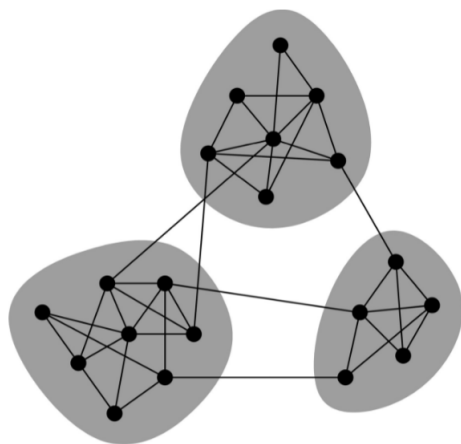


Рис. 1. Граф, содержащий три кластера

2. Доказать, что лапласиан неориентированного невзвешенного графа с  $n$  вершинами является положительно полуопределенной матрицей, имеющей  $n$  неотрицательных собственных чисел, одно из которых равно нулю.
3. Найти спектр каждого из указанных графов, т.е. найти собственные числа и вектора их лапласианов. Какие особенности спектра каждого из графов вы можете выделить? Какова их связь с количеством кластеров?
4. Найти количество кластеров в графе  $G_3$ , используя второй собственный вектор лапласиана. Для демонстрации кластеров выведите на графике исходную матрицу смежности и её отсортированную версию.

5. Опциональное задание №2. Реализуйте алгоритм DBSCAN и произведите с помощью него кластеризацию графа  $G_2$ .

## 2 Цель выполнения лабораторной работы

Познакомиться на примере с методом главных компонент, изучить кластеризацию на графе - одну из классических задач спектральной теории графов.

## 3 Выполненные задачи

1. Реализация функции метода главных компонент  $pca()$
2. Нахождение главных компонент заданного набора данных. Вывод на экран стандартных отклонений
3. Демонстрация достаточности первых двух главных компонент для сепарации типов опухолей
4. Построение лапласианов
5. Доказательство положительной полуопределенности лапласиана
6. Поиск спектра графа
7. Поиск числа кластеров в третьем графе

### Реализация функции метода главных компонент $pca()$

Метод главных компонент - популярный линейный метод, используемый для уменьшения размерности данных с минимальной потерей информации. Он применяется для наглядного представления данных, сокращения размерности описания, выделения значимой информации.

При использовании метода главных компонент предполагается, что задана некоторая матрица данных  $\mathbf{X} \in \mathbb{R}^{m \times n}$ , где строки соответствуют измерениям, столбцы - определенным показателям этих измерений (см. рис. 2)

$$\mathbf{X} = \begin{array}{c} \begin{array}{c} \xrightarrow{\text{Показания}} \\ \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \\ \downarrow \text{Измерения} \end{array} \end{array}$$

Рис. 2. Матрица данных

- Строки матрицы:  $\xi_i = [x_{i1}, \dots, x_{in}]$
- Столбцы матрицы:  $x_i = [x_{1i}, \dots, x_{mi}]$
- Вектор выборочных средних показаний измерений:

$$\bar{x} = \frac{1}{m} e^T X,$$

где  $e$  - единичный вектор.

Далее можно получить матрицу центрированных данных  $A$ . Центрирование означает линейный сдвиг выборки таким образом, чтобы средние значения признаков были равны 0. Получить ее можно следующим образом:

$$A = X - e\bar{x} = \left(E - \frac{1}{m}ee^T\right)X,$$

где  $ee^T$  - матрица единиц,  $E$  - единичная матрица.

- Строки матрицы:  $\alpha_i = [a_{i1}, \dots, a_{in}]$
- Столбцы матрицы:  $a_i = [a_{1i}, \dots, a_{mi}]$

Реализация получения матрицы центрированных данных  $A$  представлена ниже (см. Листинг 1):

Листинг1. Получение матрицы центрированных данных

---

```

1 def center(A):
2     m = len(A)
3     A_center = (np.eye(m) - 1./m * np.ones((m, m))) @ A
4     return A_center

```

---

Далее для нахождения главных компонент используется теорема о главных компонентах:

Главными компонентами матрицы центрированных данных  $A$  являются ее сингулярные вектора, при этом  $j$ -ая главная компонента соответствует  $j$ -му сингулярному вектору  $q_j$  и стандартному отклонению  $\sqrt{\nu}\sigma_j$ , где  $\sigma_j$  является  $j$ -ым сингулярным числом,  $A \in \mathbb{R}^{m \times n}$ ,  $\nu = \frac{1}{m-1}$  (по умолчанию)[1].

Таким образом, находятся главные компоненты и стандартные отклонения, соответствующие им. Стоит упомянуть о важных свойствах главных компонент:

1. Формируют ортонормальный базис, состоящий из векторов, ассоциированных с наибольшими выборочными дисперсиями
2. Первая главная компонента является направлением, вдоль которого выборочная дисперсия максимальна.
3. Вторая главная компонента является направлением, ортогональным первой главной компоненте, вдоль которого выборочная дисперсия максимальна

4. и так далее...

5. Формируют ортонормальный базис, в котором межкоординатные коэффициенты корреляции равны нулю[1].

В данной лабораторной работе была написана функция *pca()*, программно реализующая метод главных компонент. Код представлен ниже(см. Листинг 2)

Листинг 2. Реализация функции *pca()*

---

```
1 def pca(A):
2     K = A.transpose() @ A
3     self_numbers, self_vectors = np.linalg.eig(K)
4     nu = 1/(len(A) - 1)
5
6     sort_num = np.argsort(self_numbers)
7     i = np.flip(sort_num)
8     self_numbers = self_numbers[i]
9     self_vectors = self_vectors[:, i]
10
11     standard_dev = []
12     for i in range(len(self_numbers)):
13         standard_dev.append(np.sqrt(nu)*np.sqrt(self_numbers[i]))
14     return self_vectors.T, standard_dev
```

---

Для получения сингулярных чисел и векторов матрицы **A** необходимо найти собственные числа и вектора матрицы Грама  $\mathbf{K} = \mathbf{A}^T \mathbf{A}$ , которые будут являться сингулярными числами и векторами матрицы **A**. Для нахождения главных компонент производится сортировка сингулярных векторов по убыванию. Далее в цикле находим стандартные отклонения по формуле  $\sqrt{\nu}\sigma = \sqrt{\nu}\lambda$ . При этом по упомянутой выше теореме главными компонентами являются сингулярные вектора матрицы **A**, т.е. собственные вектора матрицы **K**. Таким образом, функция возвращает главные компоненты и стандартные отклонения.

### Нахождение главных компонент заданного набора данных. Вывод на экран стандартных отклонений

По условию в заданном файле находится матрица данных. Для нахождения главных компонент необходимо прочитать матрицу из файла, найти матрицу центрированных данных, а затем по указанному выше алгоритму получить главные компоненты заданного набора данных.

Функция получения набора данных из файла представлена ниже(см. Листинг 3):

Листинг 3. Получение матрицы из файла

---

```
1 def get_a():
2     with open("wdbc.data") as file:
3         for line in file:
4             data.append([x for x in line.split(',')])
```

---

```

5
6     diagnosis = []
7     for row in data:
8         if row[1] == 'M':
9             diagnosis.append("red")
10        else:
11            diagnosis.append("blue")
12
13    for row in data:
14        row.pop(1)
15        row.pop(0)
16
17    for row in data:
18        for i in range(0, len(row)):
19            row[i] = float(row[i])
20
21    A = np.array(data)
22
23    return A, diagnosis

```

Изначально чтением из файла была получена матрица данных. Однако ее изначальный формат не подходит. В матрице 32 столбца, первый из которых соответствует ID пациента, а второй соответствует диагнозу в виде буквы (М - malignant, В - benign). Первый столбец не интересует, а второй записывается в отдельный список в виде цветов (для злокачественных опухолей (М) красным цветом, для доброкачественных (В) - синим цветом). Остальные 30 столбцов соответствуют информации о пациентах, эта информация записывается в формате матрицы в переменную **A**.

Полученная матрица **A** передается в функцию *center()* (см. Листинг 1). Там получается матрица центрированных данных. Далее применяется алгоритм РСА для получения главных компонент и стандартных отклонений (см. Листинг 2).

Далее необходимо произвести вывод на экран стандартных отклонений, соответствующих номерам главных компонент. Для этого была написана функция. Код представлен ниже (см. Листинг 4):

Листинг 4. Вывод отклонений

```

1 def plot_dev(standard_dev):
2     fig, ax = plt.subplots()
3     pc_numbers = [i for i in range(0, len(standard_dev))]
4     ax.plot(pc_numbers, standard_dev, 'o--')
5     ax.grid()
6     plt.show()

```

Ниже представлен график стандартных отклонений (см. рис. 3):



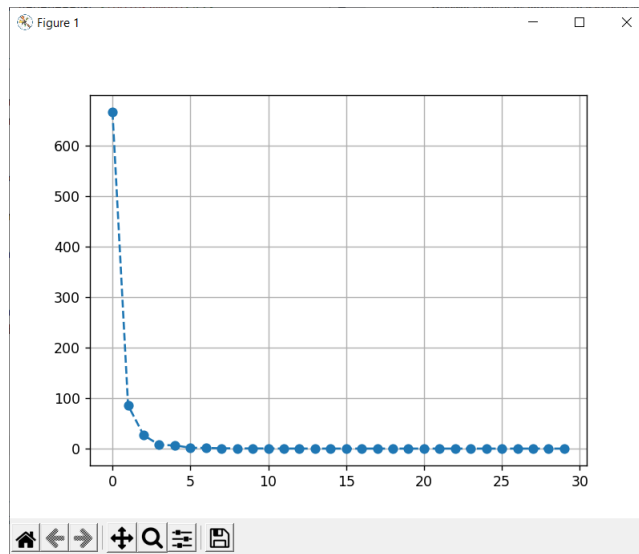


Рис. 3. Стандартные отклонения

### Демонстрация достаточности первых двух главных компонент для сепарации типов опухолей

По графику(рис. 3) видно, что основное отклонение сосредоточено на первой и второй главных компонентах. По сути этих двух компонент достаточно, что произвести сепарацию типов опухолей. Покажем это на графике. Для этого была написана функция *graph()*. Код представлен ниже(см. Листинг 5):

Листинг 5. Функция вывода данных и главных компонент на экран

---

```

1 def graph(A, pc, diagnosis):
2     fig, ax = plt.subplots()
3     x = [np.mean(A[:, 0]), np.mean(A[:, 1])]
4     sigma = [st.stdev(A[:, 0]), st.stdev(A[:, 1])]
5     _A = (A - x) / (sigma)
6
7     ax.scatter(_A[:, 0], _A[:, 1], c=diagnosis, s=3)
8     ax.plot([0], [0], 'go', markersize=10)
9
10    _max = np.max(np.abs(_A))
11    for item in pc:
12        ax.plot([0, _max/1.5 * item[0]], [0, _max/1.5 * item[1]], linewidth=3)
13    ax.grid()
14    plt.show()

```

---

Изначально производится масштабирование исходных данных для удобства восприятия. Сначала находятся  $\bar{x}$  как среднее арифметическое от всех. Затем находятся стандартные отклонения через функция *stdev()* библиотеки *statistics*. Далее производится само масштабирование, а затем вывод исходных данных по цветам(синим

цветом выводятся пациенты с доброкачественной опухолью, а красным - со злокачественной опухолью). После этого выводятся главные компоненты. График представлен ниже(рис. 4):

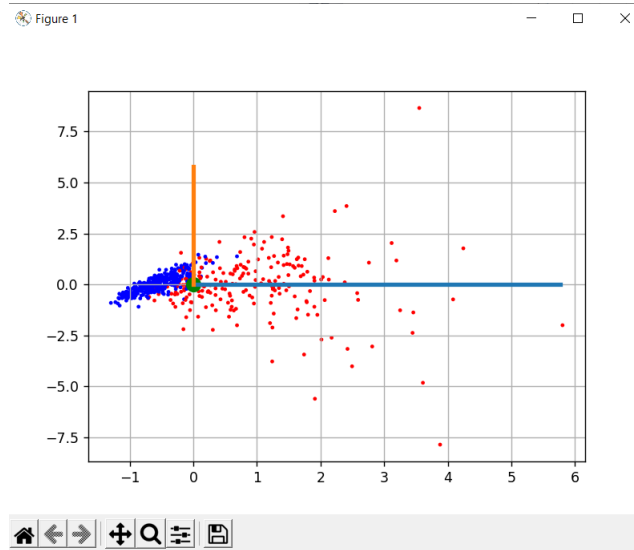


Рис. 4. Набор данных. Главные компоненты

График наглядно показывает, что первых двух главных компонент достаточно для сепарации типов опухолей для подавляющего их большинства.

## Построение лапласианов

Граф - это геометрическая фигура, которая состоит из точек и линий, которые их соединяют. Точки называют вершинами графа, а линии — ребрами.

Для любого графа существует матрица смежности - квадратная матрица, в которой каждый элемент принимает одно из двух значений: 0 или 1(в зависимости от существования связи между вершинами). Число строк матрицы смежности равно числу столбцов и соответствует количеству вершин графа.

Согласно условию необходимо построить лапласианы(матрицы Кирхгофа) для трех графов:

- Полный граф  $G_1$  имеющий 10 узлов
- граф  $G_2$ , изображенный на рисунке 1
- граф  $G_3$ , матрица смежности которого хранится в файле: <https://archrk6.bmstu.ru/index.php/f/854844>

Лапласианом графа называется матрица  $L = D - A$ , где  $A$  – матрица смежности и  $D$  – матрица, на главной диагонали которой расположены степени вершин графа, а остальные элементы равны нулю.

Для полного графа  $G_1$ , имеющего 10 узлов, матрица смежности будет заполнена единицами за исключением диагонали, на которой будут нули. Для построения лапласиана была написана функция *laplacian\_G1()*, реализация которой представлена ниже(см. Листинг 6):

Листинг 6. Реализация функции *laplacian\_G1()*

---

```

1 def laplacian_G1():
2     adj_matrix = np.ones((10, 10))
3
4     for i in range(0, 10):
5         adj_matrix[i][i] = 0
6
7     D = np.zeros((10, 10))
8     for i in range(0, 10):
9         D[i][i] = 9
10
11    L = D - adj_matrix
12    return L

```

---

Изначально создается матрица единиц размерностью  $10 \times 10$ . Затем диагональные элементы зануляются, и получается матрица смежности. Затем формируется матрица  $D$ , состоящая только из нулей. Поскольку речь идет о полном графе, то степень всех его вершин одинакова и равна  $10 - 1 = 9$ . Диагональные элементы равны 9. Затем берется разница этих матриц, которая равна лапласиану этого графа.

Для графа  $G_2$  по рисунку составляется список смежности,  $i$ -ый элемент которого также является списком, в котором указаны вершины, с которыми связана  $i$ -ая вершина. Для построения лапласиана была написана функция *laplacian\_G2()*, реализация которой представлена ниже(см. Листинг 7):

Листинг 7. Реализация функции *laplacian\_G2()*

---

```

1 def laplacian_G2():
2     adj_list = [[2, 5, 6], [1, 3, 4, 5], [2, 5, 6], [2, 5, 7, 8, 18], [1, 2, 3, 4, 7, 8], [1, 3, 7, 9],
3               [4, 5, 6, 8, 13], [4, 5, 7, 16], [6, 11, 13], [11, 12, 13], [9, 10, 12, 13], [10, 11,
4               13, 15],
5               [7, 9, 10, 11, 12], [16, 18, 20], [12, 16, 18, 20], [8, 14, 15, 17, 18], [16, 18,
6               20],
7               [4, 14, 15, 16, 17, 19, 20], [18, 20], [14, 15, 17, 18, 19]]
8
9     adj_matrix = np.zeros((20, 20))
10
11    for item in range(0, len(adj_list)):
12        for i in adj_list[item]:
13            adj_matrix[item][i - 1] = 1
14
15    D = np.zeros((20, 20))

```

---

```

15     for i in range(0, len(adj_list)):
16         D[i][i] = len(adj_list[i])
17
18     L = D - adj_matrix
19
20     return L

```

---

Построение матрицы смежности происходит по списку смежности. Как уже было сказано выше,  $i$ -ый элемент списка смежности показывает, с какими вершинами связана  $i$ -ая вершина. С учетом этого несложно составить матрицу смежности. Затем составляется матрица  $D$ , ее диагональные элементы заполняются степенью каждой вершины (длиной соответствующего элемента списка смежности). Затем берется разность полученных матриц, которая равна лапласиану графа  $G_2$ .

Для графа  $G_3$  матрица смежности расположена в заданном файле. Она берется путем чтения из файла. Для построения лапласиана написана функция `laplacian_G3()`, реализация которой представлена ниже (см. Листинг 8):

Листинг 8. Реализация функции `laplacian_G3()`

---

```

1 def laplacian_G3():
2     data = []
3     with open("adjacency_matrix.txt") as file:
4         for line in file:
5             data.append([int(x) for x in line.split(' ')])
6
7     adj_matrix = np.array(data)
8
9     powers = []
10
11     for row in adj_matrix:
12         powers.append(sum(row, 0))
13
14     D = np.zeros((1000, 1000))
15
16     for i in range(0, len(powers)):
17         D[i][i] = powers[i]
18
19     return D - adj_matrix

```

---

По прочитанной из файла матрице смежности составляется список, в котором  $i$ -ый элемент будет равен степени  $i$ -ой вершины графа. Затем составляется матрица  $D$ , диагональные элементы которой заполняются соответствующими элементами списка степеней вершин. Затем берется разность матриц, которая равна лапласиану графа  $G_3$ .

### Доказательство положительной полуопределенности лапласиана

Лапласианом графа является матрица  $L = D - A$ , где  $A$  - матрица смежности, а  $D$  - матрица, на диагонали которой расположены степени вершин графа, а остальные элементы равны нулю. Необходимо доказать, что лапласиан неориентированного невзвешенного графа с  $n$  вершинами является положительно полуопределенной матрицей, имеющей  $n$  неотрицательных собственных чисел, одно из которых равно нулю.

Для матрицы  $A$  известно, что она является положительно полуопределенной матрицей, если она симметрична и для любого ненулевого вектора  $x$  выполняется неравенство:

$$x^T B x \geq 0$$

Очевидно, что матрица смежности  $A$  симметрична, а поскольку в матрице  $D$  все элементы за исключением диагональных нулевые, тоже можно сказать о её симметричности. Отсюда следует, что матрица  $L$  симметрична. Докажем выполняемость неравенства.

$$x^T L x = x^T D x - x^T A x \quad (1)$$

Выражение  $x^T D x$  запишем в виде

$$x^T D x = \sum_{i=1}^n \sum_{j=1}^n d_{ij} x_i x_j = \sum_{i=1}^n d(i) x_i^2,$$

где  $d(i)$  является степенью  $i$ -ой вершины (такое преобразование возможно, т.к. только диагональные эл-ты ненулевые). Аналогично для матрицы смежности  $A$ :

$$x^T A x = \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j$$

Известно, что в матрице смежности, если элемент на пересечении  $i$ -ой строки и  $j$ -ого столбца равен 1, то и на пересечении  $j$ -ой строки и  $i$ -ого столбца он будет равен 1. Поэтому это можно сразу учесть при суммировании. Тогда:

$$x^T A x = \sum_{(i,j) \in E} 2x_i x_j,$$

где  $E$  - множество ребер графа, задаваемое парами вершин, которые связаны ребром.

Аналогично запишем для матрицы  $L$ :

$$x^T L x = \sum_{i=1}^n \sum_{j=1}^n l_{ij} x_i x_j$$

Тогда запишем выражение (1) в виде:

$$x^T L x = x^T D x - x^T A x = \sum_{i=1}^n d(i) x_i^2 - \sum_{(i,j) \in E} 2x_i x_j \quad (2)$$

Заметим, что выражение:

$$\sum_{i=1}^n d(i)x_i^2 = \sum_{i=1}^n \sum_{(i,j) \in E} x_i^2, \quad (3)$$

так как для каждой вершины  $i = 1, 2, \dots, n$  можно пройти по вершинам, связанным с  $i$ -ой, добавив к сумме  $x_i$  такое количество раз, сколько вершин связано с  $i$ -ой. Тогда выражение (4) можно записать так:

$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \mathbf{x}^T \mathbf{D} \mathbf{x} - \mathbf{x}^T \mathbf{A} \mathbf{x} = \sum_{i=1}^n \sum_{(i,j) \in E} x_i^2 - \sum_{(i,j) \in E} 2x_i x_j \quad (4)$$

Если предположить, что в графе есть ребро, соединяющее вершины  $i'$  и  $j'$ , то можно заметить, что в выражении (3) ребро будет учтено, когда  $i = i'$  и еще когда  $j = j'$ . Поскольку суммирование идет по  $i = 1, 2, \dots, n$ , то:

$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \mathbf{x}^T \mathbf{D} \mathbf{x} - \mathbf{x}^T \mathbf{A} \mathbf{x} = \sum_{(i,j) \in E} (x_i + x_j)^2 - \sum_{(i,j) \in E} 2x_i x_j = \sum_{(i,j) \in E} (x_i - x_j)^2 \geq 0 \quad (5)$$

Итак, доказано, что матрица  $\mathbf{L}$  положительно полуопределена. Докажем, что собственные числа матрицы  $\mathbf{L}$  неотрицательны, а одно из них равно нулю.

Собственным вектором матрицы  $\mathbf{L}$  будет называться ненулевой вектор  $\mathbf{u}$ , такой что

$$\mathbf{L} \mathbf{u} = \lambda \mathbf{u},$$

где  $\lambda$  - собственное число матрицы  $\mathbf{L}$ .

Поскольку матрица положительно полуопределена, то  $\mathbf{x}^T \mathbf{L} \mathbf{x} \geq 0$  для любого  $\mathbf{x}$ . Поскольку  $\mathbf{u}$  - собственный вектор матрицы  $\mathbf{L}$ , то:

$$\mathbf{u}^T \mathbf{L} \mathbf{u} = \mathbf{u}^T \lambda \mathbf{u} = \lambda \mathbf{u}^T \mathbf{u}$$

Т.к.  $\mathbf{u}$  - ненулевой вектор, то, очевидно, что  $\mathbf{u}^T \mathbf{u} > 0$ . Следовательно,  $\lambda \geq 0$

Докажем, что одно из собственных чисел равно 0. Для этого запишем характеристическое уравнение:

$$|\mathbf{L} - \lambda \mathbf{E}| = \begin{vmatrix} l_{11} - \lambda & l_{12} & \dots & l_{1n} \\ l_{21} & l_{22} - \lambda & \dots & l_{2n} \\ \vdots & \vdots & \dots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} - \lambda \end{vmatrix} = 0$$

Прибавим к первой строке все остальные, определитель не меняется:

$$\begin{vmatrix} l_{11} + l_{21} + \dots + l_{n1} - \lambda & l_{12} + l_{22} + \dots + l_{n2} - \lambda & \dots & l_{1n} + l_{2n} + \dots + l_{nn} - \lambda \\ l_{21} & l_{22} - \lambda & \dots & l_{2n} \\ \vdots & \vdots & \dots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} - \lambda \end{vmatrix} = 0$$

Можно заметить, что сумма элементов каждого столбца и строки равна нулю. Тогда:

$$\begin{vmatrix} -\lambda & -\lambda & \dots & -\lambda \\ l_{21} & l_{22} - \lambda & \dots & l_{2n} \\ \vdots & \vdots & \dots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} - \lambda \end{vmatrix} = 0$$

Отсюда очевидно, что одно из собственных чисел равно нулю. Что и требовалось доказать.

## Поиск спектра графа

Поиск спектра графа означает нахождение собственных чисел и векторов лапласиана этого графа. Для нахождения собственных чисел использовалась функция `np.linalg.eig()`. Затем с помощью функций `np.argsort()` и `np.flip()` производилась сортировка по убыванию. Для построения графиков собственных чисел лапласианов каждого из графов была написана функция, реализация которой представлена ниже (см. Листинг 9):

Листинг 9. Поиск собственных чисел и векторов. Вывод собственных чисел

---

```
1 def plot_laplacian(L_i):
2     L_numbers, L_vectors = np.linalg.eig(L_i)
3
4     L_vectors = L_vectors[:, np.argsort(L_numbers)]
5     L_vectors = np.flip(L_vectors)
6     L_numbers = L_numbers[np.argsort(L_numbers)]
7     L_numbers = np.flip(L_numbers)
8
9     fig, ax = plt.subplots()
10    ax.plot(1. + np.arange(len(L_numbers)), L_numbers, 'o--')
11
12    ax.grid()
13    plt.show()
```

---

Графики собственных чисел для лапласианов всех графов представлены ниже:

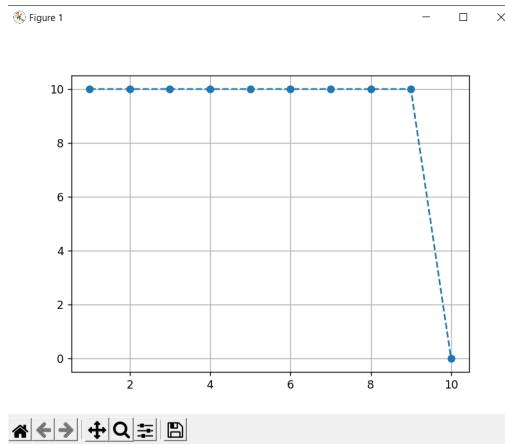


Рис. 5. Собственные числа для лапласиана первого графа

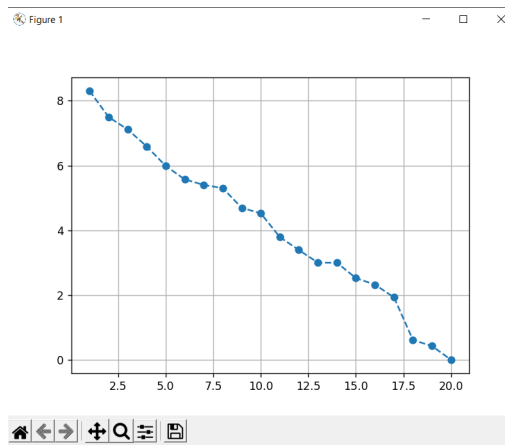


Рис. 6. Собственные числа для лапласиана второго графа



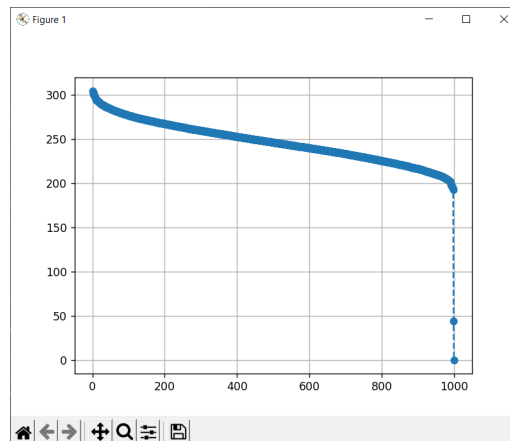


Рис. 7. Собственные числа для лапласиана третьего графа

Из рисунков видно, что последнее(отсортированы по убыванию) собственное число везде равно нулю. У лапласиана первого графа все остальные значения близки к  $n$ . При этом у графа один кластер, поскольку граф полный. У лапласиана второго графа 3 околонулевых собственных числа. При этом у него 3 кластера, как видно из рисунка 1. Можно сделать вывод, что количество околонулевых собственных чисел лапласиана графа равно количеству кластеров этого графа. Для третьего графа количество околонулевых собственных чисел равно 2. Следовательно, у него 2 кластера.

### Поиск числа кластеров в третьем графе

Для наглядного показа количества кластеров в третьем графе отсортируем матрицу смежности графа  $G_3$  по второму собственному вектору лапласиана. Для этого написана функция, реализация которой представлена ниже(см. Листинг 9):

Листинг 9. Сортировка матрица смежности

---

```

1 def plot_adj_matrix(L3_vectors, M):
2     plt.matshow(M)
3
4     i = np.argsort(L3_vectors[:, 1].T)
5     M1 = M[np.ix_(i, i)]
6     plt.matshow(M1)
7
8     plt.grid()
9     plt.show()

```

---

Для вывода используем функцию `plt.matshow()` библиотеки `matplotlib`. Сначала отобразим неотсортированную матрицу смежности.

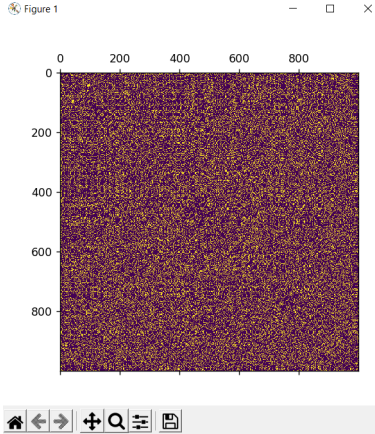


Рис. 8. Неотсортированная матрица смежности

Далее отобразим отсортированную матрицу смежности:

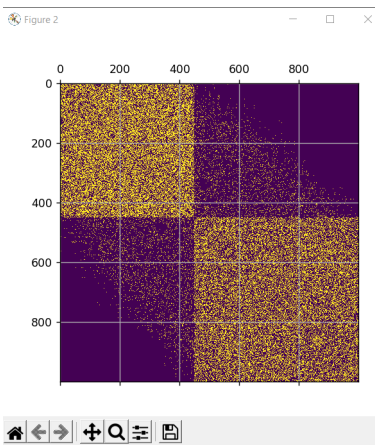


Рис. 9. Отсортированная матрица смежности

На рисунке 9 наглядно продемонстрировано наличие двух кластеров у графа  $G_3$ .

При выполнении лабораторной работы вызов всех указанных функций производится в основном методе `main()`.

## 4 Заключение

1. В рамках лабораторной работы был изучен метод главных компонент на примере информации о пациентах. Было показано, что первых двух главных компонент достаточно для того, чтобы сепарировать доброкачественные и злокачественные опухоли.
2. В продвинутой части была рассмотрена одна из классических задач спектральной теории графов - задача разделения графа на сильно связанные компоненты.



3. Было доказано, что лапласиан неориентированного невзвешенного графа с  $n$  вершинами является положительно полуопределенной матрицей, имеющей  $n$  неотрицательных собственных чисел, одно из которых равно нулю.
4. Было найдено количество кластеров для каждого из заданных графов. В частности, для графа  $G_3$  было наглядно продемонстрировано количество кластеров с использованием сортировки матрицы смежности по второму собственному вектору лапласиана.

### Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.

### Выходные данные

Магомедов З.А. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2021. — 19 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка:  ассистент кафедры РК-6, PhD А.Ю. Першин  
 Решение и вёрстка:  студент группы РК6-54Б Магомедов З.А.

2021, осенний семестр