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:	Зотов Даниил Александрович
Группа:	РК6-53Б
Тип задания:	лабораторная работа
Тема:	Спектральное и сингулярное разложения

Студент _____ (подпись, дата) Зотов Д. А. (Фамилия, И.О.)

Преподаватель _____ (подпись, дата) _____ (Фамилия, И.О.)

Москва, 2021

Содержание

Спектральное и сингулярное разложения

1 Задание

Спектральное разложение (разложение на собственные числа и вектора) и сингулярное разложение, то есть обобщение первого на прямоугольные матрицы, играютнастолько 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), безпреувеличения 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 классическихзадач спектральной теории графов – задачу разделения графа на сильно связанныекомпоненты (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 соответствующихстандартных 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 детальногоисследования снимков и анализов. Дополнительно на основе снимков был автоматически выявлен и задокументирован ряд характеристик опухолей: радиус, пло-щадь, фрактальная размерность и так далее (всего 30 характеристик). Постановку диагноза можно автоматизировать, если удастся создать алгоритм, классифицирущий опухоли исключительно на основе этих автоматически получаемых харак-теристик. 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 со-ответствуют характеристикам опухоли.

3. Найти главные компоненты указанного набора данных, используя функцию *pca(A)*.

4. Вывести на экран стандартные отклонения, соответсвующие номерам главныхкомпонент.

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 вывестина 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 смежностии D – матрица, на главной диагонали которой расположены степени вершин графа,а остальные элементы равны нулю.

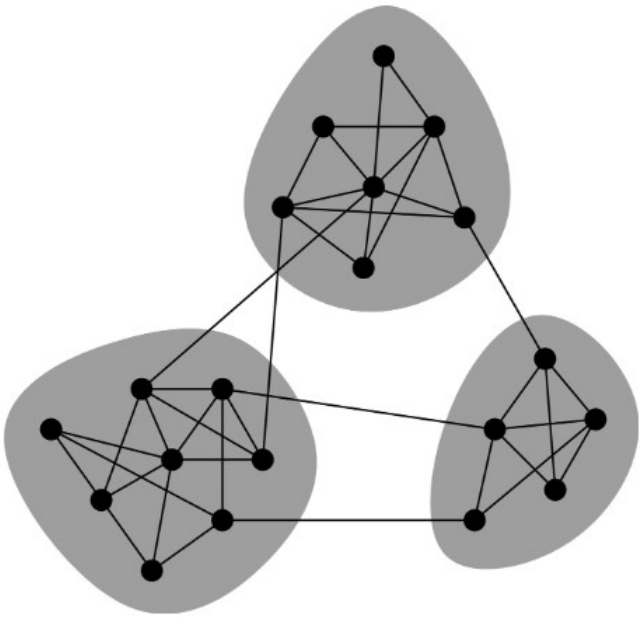

Рис. 1. Граф, содержащий три кластера

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

Цель выполнения лабораторной работы – изучение спектрального и сингулярного разложения на прикладном примере линейной алгебры. Знакомство с популярным методом понижения размерности - PCA и классической задачи спектральной теории графов - кластеризация на графе.

3 Выполненные задачи

1. Реализована функция, возвращающая список главных компонент и стандартные отклонения.

2. Для датасета Breast Cancer Wisconsin применена реализованная функция и показано, что для подавляющего большинства проекций на две компоненты уже достаточно для того, чтобы разделить опухоли на типы.

3. Построены матрицы Кирхгофа для трех графов G_1, G_2, G_3.

4. Найдены спектры для матриц лапласианов указанных графов.

5. Проведена спектральная кластеризация графа G_3 по вектору Фидлера.

4 Метод главных компонент

Метод главных компонент (Principal Component Analysis) - один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации. С математической точки зрения, в основе метода главны компонент лежит нахождение линейно некоррелированных переменных, называемых главными компонентами. Метод главных компонент применяется к данным, записанным в виде прямоугольной матрицы $\mathbf{X}$ размерностью $m \times n$.

Матрица данных $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ имеет вид:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}, \tag{1}$$

где строки матрицы $\boldsymbol{\xi_i} = [x_{i1}, \ldots, x_{in}]$ являются измерениями, столбцы матрицы $\boldsymbol{x_i} = [x_{1i}, \ldots, x_{mi}]$ являются показаниями (признаками). Вектор выборочных средних показаний измерений имеет вид:

$$\overline{\boldsymbol{x}} = \frac{1}{m} \boldsymbol{e}^T \boldsymbol{X}, \tag{2}$$

где $\boldsymbol{e}$ - единичный вектор.

Целью метода главных компонент является извлечение из матрицы данных $\boldsymbol{X}$ нужной информации. Метод полезен в машинном обучении, когда возникает ситуация, что при недостатке данных модель будет недообучена, а при переизбытке - переобучена. Поэтому, с учетом того, что нельзя увеличивать количество признаков для модели до бесконечности, можно уменьшить размерность данных. Это можно сделать с помощью метода главных компонент, к примеру.

Для метода главных компонент требуется предобработка данных, а именно - приведение к матрице центрированных данных $\boldsymbol{A}$. Рассмотрев отклонения от средних значений, а именно $a_{ij} = x_{ij} - \overline{x}_j$, получим матрицу центрированных данных:

$$\boldsymbol{A} = \boldsymbol{X} - \boldsymbol{e}\overline{\boldsymbol{x}} = (\boldsymbol{E} - \frac{1}{m}\boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{X}, \tag{3}$$

где $\boldsymbol{ee^T}$ - матрица единиц, $\boldsymbol{E}$ - единичная матрица.

Математический смысл нахождения матрицы центрированных данных в том, что среднее для элементов становится равным нулю, а геометрический смысл - ищется центр "облака данных куда переносится начало координат для базиса, который будет сформирован в ходе реализации метода главных компонент.

Для метода главных компонент центральными понятиями являются сингулярные числа и вектора. Сингулярными числами $\sigma_1, ..., \sigma_r$ матрицы $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ называются неотрицательные вещественные числа $\sigma_i = \sqrt{\lambda_i}$, где λ_i - ненулевые собственные числа соответствующей матрицы Грама $\boldsymbol{K} = \boldsymbol{A^T A}$. Сингулярные числа сортируются в убывающую последовательность $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$. Ассоциированные собственные вектора матрицы Грама $\boldsymbol{K}$ называются сингулярными векторами.

Главными компонентами будут называться сингулярные вектора матрицы A, при этом j-я главная компонента соответствует j-му сингулярному вектору $\boldsymbol{q_j}$ и стандартному

отклонению $\sqrt{\nu}\sigma_j$, где ν - коэффициент, зависящий от выбора оценки ковариации (по умолчанию $\nu = \frac{1}{m-1}$), σ_j - j-е сингулярное число.

Для получения главных компонент и соответствующих стандартных отклонений была реализована функция *pca(X)*, принимающая на вход матрицу данных $\boldsymbol{X}$. Алгоритм получения главных компонент и соответствующих стандартных отклонений выглядит следующим образом:

1. Для реализации метода необходима предобработка данных, поэтому матрица данных приводится к матрице центрированных данных. Для удобства была реализована дополнительная функция *get_ normalized_ data_ matrix(X)*, которая принимает на вход матрицу данных $\boldsymbol{X}$. Она возвращает центрированную матрицу данных $\boldsymbol{A}$, вычисленную по формуле (3).

2. С помощью функции *numpy.linalg.eig* библиотеки *NumPy* находятся собственные числа и вектора матрицы Грама $\boldsymbol{A^T A}$.

   Функция *numpy.linalg.eig* имеет следующий вид:

   ```
   numpy.linalg.eig(a):
       a: array - матрица, для которой будут вычислены собственные числа и
                  вектора.
   Возвращает:
       w: array - собственные значения (не обязательно упорядочены);
       v: array - собственные вектора.
   ```

3. Производится сортировка значений в порядке убывания, так как сингулярные числа сортируются в убывающую последовательность. Сортировка значений производится с помощью функции *numpy.argsort* библиотеки *NumPy* в паре с функцией *numpy.flip* (осуществляющей переворот массива в обратном порядке, т. е., для сортировки по убыванию).

   Функция *numpy.argsort* имеет следующий вид:

   ```
   numpy.argsort(a, axis=- 1, kind=None, order=None) - возвращает индексы
                                                  для сортировки по возрастанию.
       a: array-like - массив для сортировки;
       axis: int (optional) - ось для сортировки.
   ```

4. Путем взятия квадратного корня из собственных чисел, находятся сингулярные числа. Определяется $\nu = \frac{1}{m-1}$. В цикле по списку сингулярных чисел для каждого σ_i находится стандартное отклоенение.

5. Функция возвращает список главных компонент и список стандартных отклонений.

Листинг 1. Реализация функции получения главных компонент.

```
def pca(X):
    A = get_normalized_data_matrix(X)
    w, v = np.linalg.eig(A.T @ A)

    desc_seq_ids = np.flip(np.argsort(w))
    w = w[desc_seq_ids]
    v = v[:, desc_seq_ids]

    w = np.sqrt(w)
    nu = 1/(X.shape[0] - 1)

    sigmas = []
    for num in w:
        sigmas.append(sqrt(nu) * num)
    sigmas = np.array(sigmas)

    return v.T, sigmas
```

Полный листинг функции получения главных компонент и стандартных отклонений приведен ниже (см. листинг 1).

5 Применение метода главных компонент на датасете Breast Cancer Wisconsin

Для применения реализованного метода главных компонент был скачан набор данных Breast Cancer Wisconsin Dataset. Для датасета была проведена предобработка данных: первые два столбца, содержащие атрибуты *id*, *diagnosis* были записаны в отдельный датафрейм. В этом датафрейме был также добавлен новый столбец *color*, который на основе диагноза ('B' или 'M') присваивал колонке код цвета для вывода на график в будущем.

Для нахождения главных компонент, первые два атрибута (ранее сохраненные в отдельный датафрейм) удалены из основной матрицы. Учитывая то, что результирующей матрицей является матрица размерностью 30×30 (количество оставшихся атрибутов в наборе после исключения первых двух), найденные главные компоненты представлены в директории */src/prcomp_ result.xlsx* в виде Excel-таблицы.

Построим зависимость стандартных отклонений $\sqrt{\nu}\sigma_i$ от номера i главных компонент (рис. 2). Из графика видно, что первым двум компонентам соответствуют самые большие значения стандартных отклонений и выборочных дисперсий. Далее - происходит сильное падение данных показателей. Поэтому оставшиеся компоненты можно отбро-

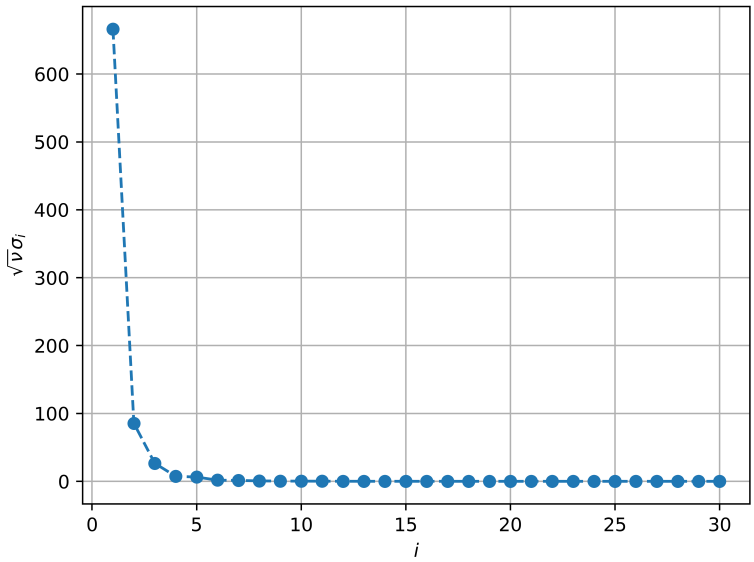

Рис. 2. График зависимости стандартных отклонений от номера главных компонент

сить, чтобы произвести визуализацию данных в измененном двумерном пространстве. Это пространство будет трансформированным в соответствии с главными компонентами, а главные компоненты будут базисом этого пространства.

Для отображения данных в трансформированном пространстве реализована дополнительная функция *plot_ data*. На вход функция получает матрицу данных, спроектированную на два сингулярных вектора, матрицу Грама двух главных компонент и цвета для раскраски scatter plot (красный для опухолей типа benign, синий для опухолей тима malignant). Листинг функции *plot_ data* приведен ниже (см. листинг 2).

В результате проецирования данных на первые две главные компоненты получим двумерный график, представленный на рис. 3. Первой главной компоненте соответствует направление, по которому выборочная дисперсия и стандартное отклонение максимальны, а вторая главная компонента перпендикулярна к первому направлению и выбрана, по сути, чтобы описать оставшиеся изменения в данных. Из графика видно, что для большинства точек можно качественно произвести сепарацию типов опухолей (синие точки соответствуют опухолям типа malignant, красные точки соответствуют опухолям типа benign). Такую классификацию легко произвести из-за того, что стандартное отклонение первых двух главных компонент велико. Таким образом, полученный график хорошо демонстрирует результат уменьшения размерности: на полученных главных компонентах легко применить алгоритмы классификации или кластеризации.

Листинг 2. Функция построения проекции данных на трансформированное пространство.

```
def plot_data(A, principal_components, colors):
    fig, ax = plt.subplots(figsize=(8, 6))
    x_bar = [np.mean(A[:, 0]), np.mean(A[:, 1])]
    sigmas_bar = [statistics.stdev(A[:, 0]), statistics.stdev(A[:, 1])]
    A_bar = (A - x_bar) / sigmas_bar
    ax.scatter(A_bar[:, 0], A_bar[:, 1], c=colors, s=3)
    ax.plot([0], [0], 'ro', markersize=5)
    max_val = np.max(np.abs(A_bar))
    for pc in principal_components:
        ax.plot([0, max_val * pc[0]], [0, max_val * pc[1]], linewidth=3)
    ax.set_xlabel(r'$x_1$')
    ax.set_ylabel(r'$x_2$')
    ax.grid(alpha=0.6)
    plt.savefig('data_plot.png', dpi=750)
    plt.show()
```

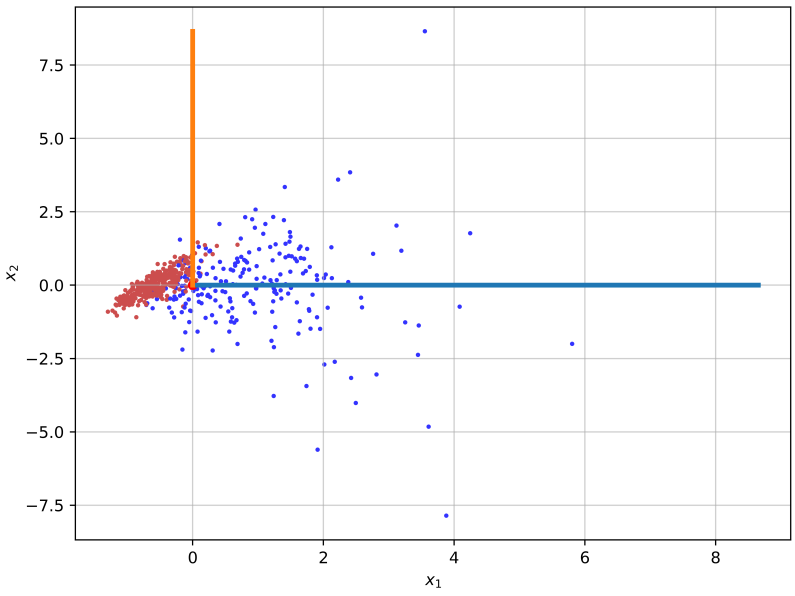

Рис. 3. Проекция данных на первые две главные компоненты.

6 Построение матриц Кирхгофа

Матрица Кирхгофа (лапласианы) - одно из представлений конечного графа с помощью матрицы. Пусть дан простой граф G с $|V(G)| = n$ вершинами. Тогда матрица Кирхгофа $K = (k_{ij})_{n \times n}$ данного графа определяется как $K = D - A$, где A - матрица смежности графа, а D:

$$d_{ij} = \begin{cases} deg(v_i) & \text{если } i = j \\ 0. \end{cases} \tag{4}$$

Для построения лапласианов реализованы функции *LG1(num), LG2(num), LG3(filename)*. Функции имеют схожие алгоритмы.

Алгоритм получения лапласиана для полного графа (функция LG1):

1. На вход подается число num, обозначающее количество вершин в графе.
2. Создается матрица единиц размерности $num \times num$, из которой вычитается единичная матрица (обнуление главной диагонали), в результате чего получим матрицу A.
3. Создается единичная матрица размерности $num \times num$, которая умножается на $num - 1$ (степень вершины в полном графе на одну меньше, чем число вершин), в результате чего получим матрицу D.
4. Функция возвращает разность матриц D и A.

Полный листинг функции LG1 приведен на листинге 3.

Листинг 3. Реализация функции построения лапласиана для полного графа.

```
def LG1(num):
    A = np.ones((10,10))
    diag_elem = num - 1

    i = 0
    for row in A:
        row[i] = 0
        i += 1

    D = np.eye(num) * diag_elem

    return D - A
```

1. На вход подается число num, обозначающее количество вершин в графе.
2. В список *connections* для каждой из вершин записываются номера смежных ей вершин в виде внутреннего списка.

3. Создается матрица нулей размерностью $num \times num$, которая является матрицей смежности A и заполняется построчно: в каждой i-ой строке единицами заполняются те индексы, которые заполнены в i-ом элементе списка *connections*.

4. Создается матрица нулей размерностью $num \times num$, которая является матрицей D и заполняется построчно: в каждой строке i-ой строке элемент d_{ii} равен длине i-го элемента списка *connections*.

5. Функция возвращает разность матриц D и A.

Полный листинг функции LG2 приведен на листинге 4.

Листинг 4. Реализация функции построения лапласиана для графа на рис. 1.

```
def LG2(num):
    shape = num
    connections = [
        [2,5,6], [1,4,5,3], [2,5,6], [2,5,7,8,11], [2,3,4,7,8,1], [1,3,7,20],
    [4,5,8,17,6], [4,5,7,9],
        [8,11,12,10,15], [9,11,13], [9,10,12,13,15,4,14], [9,11,13],
    [11,12,10,15,14], [11,13], [9,11,13,16],
        [15,17,18,19], [7,16,18,19,20], [16,17,19,20], [16,17,18], [6,17,18]
    ]
    A = np.zeros((num, num))
    i = 0
    for row in A:
        conns = connections[i]
        for num in conns:
            row[num-1] = 1
        i += 1
    D = np.zeros((shape, shape))
    j = 0
    for row in D:
        conns = connections[j]
        deg = len(conns)
        row[j] = deg
        j += 1

    return D - A
```

Алгоритм получения лапласиана для графа с известной матрицей смежности (функция LG3):

1. На вход подается имя файла *filename*, в котором хранится матрица смежности графа. С помощью функции open и конструкции with as, данные из файла построчно записываются в виде двумерной матрицы $adjM$ (список списков).

2. Создается матрица нулей размерностью $len(adjM) \times len(adjM)$, которая является матрицей D и заполняется построчно. Для каждой i-ой строки элемент d_{ii} равен сумме элементов в i-ой строки матрицы смежности.

3. Функция возвращает разность матриц D и A.

Полный листинг функции LG3 приведен на листинге 5.

Листинг 5. Реализация функции построения лапласиана для графа с известной матрицей смежности.

```
def LG3(filename):
    with open(filename, 'r') as f:
        adjM = [[int(number) for number in row.split(' ')] for row in f]

    D = np.zeros((len(adjM), len(adjM)))
    i = 0
    for row in adjM:
        sum = 0
        for item in row:
            sum += item
        D[i][i] = sum
        i += 1
    A = np.array(adjM)

    return D - A
```

Учитывая то, что матрицы Кирхгофа имеют размерности 10×10, 20×20, 1000×1000 соответственно, для экономии места они выведены в виде Excel-таблиц. Таблицы можно найти в директории */src/laplasians*.

7 Доказательство полуопределенности лапласиана неориентированного невзвешенного графа

Требуется: доказать, что лапласиан неориентированного невзвешенного графа с n вершинами является положительно полуопределенной матрицей, имеющей n неотрицательных собственных чисел, одно из которых равно нулю.

Доказательство:
По определению, ненулевой вектор $\boldsymbol{x}$, который при умножении на некоторую матрицу $\boldsymbol{A}$ ($\boldsymbol{L}$ - лапласиан, в нашем случае) превращается в самого же себя с числовым коэффициентом λ, называется собственным вектором матрицы, а число λ - собственным числом матрицы:

$$\boldsymbol{Lx} = \lambda\boldsymbol{x} \tag{5}$$

Из (5) следует, что

$$\lambda = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{Lx}. \tag{6}$$

Допустим, что граф произвольно ориентирован и $\boldsymbol{I}$ является матрицей инцидентности. Учитывая, что $I_{ij} = 1$, если вершина v_i инцидентна ребру e_j, а иначе $I_{ij} = 0$, запишем,

что $\boldsymbol{L} = \boldsymbol{I}\boldsymbol{I}^{\boldsymbol{T}}$ [5]. Тогда, подставив в (6) получим:

$$\lambda = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{L}\boldsymbol{x} = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{I}\boldsymbol{I}^{\boldsymbol{T}}\boldsymbol{x} = ||I^T x||^2 \geq 0 \rightarrow \lambda_i \geq 0. \tag{7}$$

Таким образом, лапласиан - положительно полуопределенная матрица.

Собственные числа матрицы $\boldsymbol{A}$ находятся при решении уравнения $|\boldsymbol{A} - \lambda\boldsymbol{E}| = 0$, где $\boldsymbol{E}$ - единичная матрица. В нашем случае: $|\boldsymbol{L} - \lambda\boldsymbol{E}| = 0$:

$$|\boldsymbol{L} - \lambda\boldsymbol{E}| = \begin{vmatrix} l_{11} - \lambda & l_{12} & \dots & l_{1n} \\ l_{21} & l_{22} - \lambda & \dots & l_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{m1} & l_{m2} & \dots & l_{mn} - \lambda \end{vmatrix} = 0. \tag{8}$$

После прибавления к первой строке остальных строк из (8) получим, что первая строка будет полностью равна $-\lambda$ за счет того, что сумма элементов в столбце лапласиана равна нулю:

$$|\boldsymbol{L} - \lambda\boldsymbol{E}| = \begin{vmatrix} -\lambda & -\lambda & \dots & -\lambda \\ l_{21} & l_{22} - \lambda & \dots & l_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{m1} & l_{m2} & \dots & l_{mn} - \lambda \end{vmatrix} = -\lambda \begin{vmatrix} 1 & 1 & \dots & 1 \\ l_{21} & l_{22} - \lambda & \dots & l_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{m1} & l_{m2} & \dots & l_{mn} - \lambda \end{vmatrix} = 0. \tag{9}$$

Из (9) следует, что ноль является собственным значением матрицы.

8 Нахождение спектра графа

Задача нахождения спектра графа - это задача нахождения собственных чисел и векторов лапласиана графа. Найдем собственные числа и вектора с помощью функции *numpy.linalg.eig* библиотеки *NumPy*. После их нахождения, отсортируем списки в убывающую последовательность с помощью *numpy.argsort* и *numpy.flip*. Листинг части программы, связанной с нахождением собственных векторов представлен под номером 6.

Выведем собственные числа графов на отдельных графиках в виде зависимости значения собственного числа от его номера (см. рис. 4).
Заметно, что для всех графов последним собственным числом является нулевое значение. Это ноль-значение обозначает то, что в каждом из графов мы имеем только один связный компонент (т. е., весь граф связен).

Следующее ненулевое значение называется значением Фидлера, а соответствующий вектор - вектором Фидлера. Значение Фидлера приблизительно соответствует минимальному срезу графа, необходимому для разделения графа на два связанных компонента. Если бы в каком-либо из графов уже было два связных компонента (т. е., два

Листинг 6. Нахождение спектров для графов L_1, L_2, L_3.

```
L1_v, L1_w = np.linalg.eig(LG1)
L2_v, L2_w = np.linalg.eig(LG2)
L3_v, L3_w = np.linalg.eig(LG3)
L1_w = L1_w[:, np.argsort(L1_v)]
L1_w = np.flip(L1_w)
L1_v = L1_v[np.argsort(L1_v)]
L1_v = np.flip(L1_v)
L2_w = L2_w[:, np.argsort(L2_v)]
L2_w = np.flip(L2_w)
L2_v = L2_v[np.argsort(L2_v)]
L2_v = np.flip(L2_v)
L3_w = L3_w[:, np.argsort(L3_v)]
L3_w = np.flip(L3_w)
L3_v = L3_v[np.argsort(L3_v)]
L3_v = np.flip(L3_v)
```

графа, не связанные друг с другом), то значение Фидлера было бы равно нулю. Каждое значение в векторе Фидлера дает нам информацию о том, какой стороне разреза принадлежит этот узел в зависимости от того, положителен или отрицателен элемент вектора [2].

Например, выведем значения Фидлера для графа G_1:

[-0.21339151 -0.19336806 0.83532282 -0.44842371 -0.0587721 0.08805329 -0.06670654 0.02535657 0.04256179 -0.01063254]

Из него следует, что вершины графа 1, 2, 4, 5, 7, 10 относятся к первой компоненте (кластеру), а все остальные - ко второму кластеру. При этом интересен факт, что даже полный граф, имеющий одинаковую плотность связности во всех вершинах, может быть разделен с помощью вектора Фидлера на кластеры (что на практике, конечно же, ошибочно).

На графиках также можно заметить "скачки"в значениях собственных чисел (кроме первого графа, так как граф является полным). Диапазон вершин между "скачками"значений можно также объединить в один кластер. Таким образом, для объединения вершин в один кластер можно также рассматривать "разрывы"в значениях собственных чисел.

9 Кластеризация графа с помощью вектора Фидлера

Кластеризуем граф G_3 с помощью вектора Фидлера. Учитывая, что вектор Фидлера делит граф на два связанных компонента, нетрудно догадаться, что итоговое количество кластеров равно двум. Для кластеризации графа реализована функция *cluster(L, M)*, которая принимает на вход лапласиан и матрицу смежности. Алгоритм работы функции:

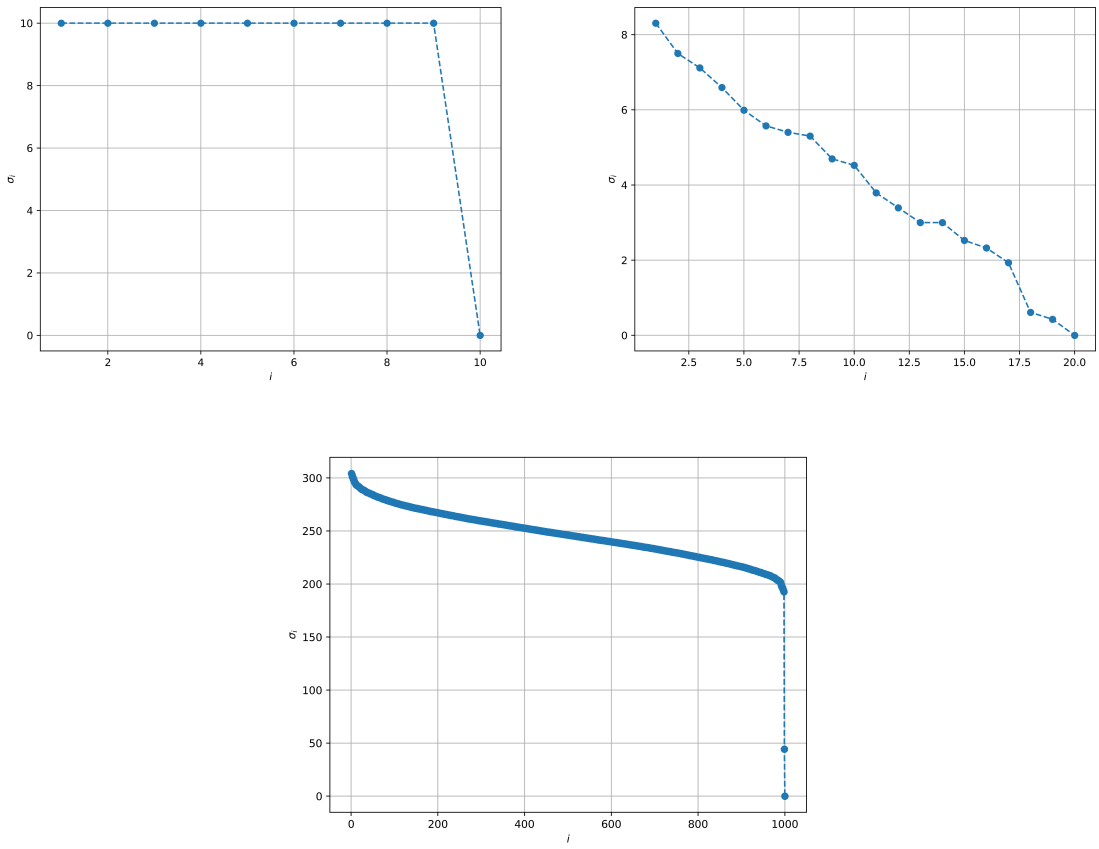

Рис. 4. Графики зависимости значений собственных чисел от номера собственного числа для графов G1, G2, G3.

1. Из собственных векторов матрицы лапласиана выбирается второй с конца, являющийся вектором Фидлера.

2. С помощью функции *np.argsort* определяются индексы, по которым производится сортировка вектора Фидлера.

3. Производится сортировка исходной матрицы в соответствии с полученными индексами.

Полный листинг функции *cluster* приведен ниже (см. листинг 7).

Листинг 7. Реализация функции разбиения графа на кластеры.

```
def cluster(L, M):
    L3 = np.array(M)
    v, w = np.linalg.eig(L)
    idx = np.argsort(v)[::-1]
    v = v[idx]
    w = w[:, idx]
    w = w.T
    idx = np.argsort(w[-2])

    clustered = L3[np.ix_(idx, idx)]
    plt.matshow(clustered)
    plt.savefig('cluster.png', dpi=750)
    plt.show()
```

Выведем измененную матрицу с помощью функции *matshow* библиотеки *matplotlib.pyplot*. Вывод представлен на рис. 5. Из рис. 5 можно сделать вывод, что кластеризация действительно разделила лапласиан на два кластера, как и ожидалось, учитывая то, что для кластеризации был использован вектор Фидлера.

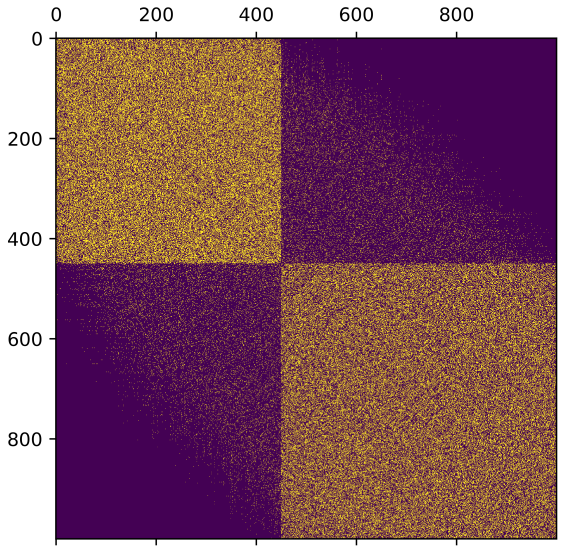

Рис. 5. Отображение матрицы после кластеризации по вектору Фидлера.

10 Заключение

1. В ходе выполнения лабораторной работы был реализован метод главных компонент, который является одним из способов понижения размерности матрицы данных. Метод применен на наборе данных Breast Cancer Wisconsin Dataset, в результате чего сделан вывод о том, что проекций на первые две главные компоненты достаточно для того, чтобы произвести визуализацию данных и сепарацию по классам.

2. Реализовано построение матриц Кирхгофа (лапласианов) для трех графов. После построения лапласианов найдены спектры и указаны их особенности.

3. В ходе построения спектров особое внимание обращено второму с конца собственному вектору - он называется вектором Фидлера и используется для разделения на два кластера.

4. Проведена кластеризация графа G_3 с помощью вектора Фидлера, в результате которой было эмпирически установлено, что граф действительно разделился на два кластера.

Список использованных источников

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

2. Спектральная кластеризация - Machine Learning Mastery [Электронный ресурс]. Режим доступа: https://www.machinelearningmastery.ru/spectral-clustering-aba2640c0d5b/

3. Матрица Кирхгофа - Wikipedia [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Матрица_Кирхгофа

4. Алгебраическая связность - Wikipedia [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Алгебраическая_связность

5. ORIE 6334 Bridging Continuous and Discrete Optimization Lecture 7 - David P. Williamson (2019)

Выходные данные

Зотов Д. А.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2021. — 16 с. URL: https://sa2systems.ru:88 (система контроля версий кафедры РК6)