МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**УТВЕРЖДАЮ:**

Заведующий кафедрой ИиСП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Бородин

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ОТЧЕТ**

**по** учебной **практике**

(вид практики)

**в** ФГБОУ ВО ПГТУ

(наименование организации)

**Вид практики** учебная

**Сроки практики** 05.07.2023 ... 18.07.2023

Выполнил обучающийся гр. ПС-22

Горяев И.А.

Руководитель практики от ПГТУ

Зав. каф. ИиСП, проф., Бородин А. В.

(должность, Ф.И.О., подпись)

Дата защиты \_\_\_19.07.2023\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Йошкар-Ола

2023

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

Вид практики *(учебная / производственная)* Учебная

Тип практики Научно-исследовательская

Обучающийся**\***Горяев Иван Андреевич

Факультет Информатики и вычислительной техники. Курс 2

Специальность (направление) Программная инженерия

Форма обучения очная

Группа ПС-22

База практики ФГБОУ ВО «ПГТУ», кафедра Информатики и системного програм-мирования

(наименование организации / структурного подразделения университета)

Целью данной работы является развитие инженерного стиля мышления при проектировании программного обеспечения различного назначения, дальнейшее развитие навыков программирования в различных средах разработки, в том числе соответствующих требованиям промышленной разработки программного обеспечения, развитие навыков научно-исследовательской работы и документирования процессов разработки и собственно программного обеспечения.

Руководитель практики от ПГТУ (кафедры)

Зав. каф. ИиСП, Бородин А. В.

(должность, Ф.И.О., подпись)

# Введение

Данный проект представляет собой разработку библиотеки программ, предназначенной для вычисления меры близости между двумя возрастающими кусочно-постоянными функциями. Эти функции представляют собой функции распределения дискретных случайных величин с областью значений в интервале [0, 1]. Вычисление меры близости основано на понятии расстояния Хаусдорфа.

Расстояние Хаусдорфа – это метрика, используемая для измерения различий между двумя множествами точек. В контексте данного проекта оно применяется для определения степени схожести между двумя функциями распределения. Расстояние Хаусдорфа между двумя функциями вычисляется путем нахождения максимального расстояния между ближайшими точками в каждой функции.

Основными компонентами библиотеки программ являются функция ReadFunctionFromFile, которая отвечает за чтение функции распределения из файла, и функция CalculateHausdorffDistance, которая вычисляет расстояние Хаусдорфа между двумя функциями. Функция ReadFunctionFromFile считывает пары значений x и y из файла и сохраняет их в виде структуры данных Point. Функция CalculateHausdorffDistance выполняет вычисление расстояния Хаусдорфа путем сравнения каждой точки первой функции с каждой точкой второй функции.

Программа предоставляет возможность анализировать и сравнивать функции распределения, что может быть полезным при решении различных задач, таких как оценка сходства между двумя наборами данных или определение степени соответствия двух распределений.

В дальнейшем планируется доработка библиотеки путем добавления дополнительных функций для работы с функциями распределения, таких как вычисление других мер близости или выполнение операций над функциями. Также планируется проведение тестирования программы на различных тестовых случаях для обеспечения надежности и корректности работы.

# Постановка задачи

Входные данные:

1. Две возрастающие кусочно-постоянные функции.
2. Формат задания кусочно-постоянной функции - массив точек функции распределения действительной дискретной случайной величины.

Выходные данные:

1. Мера близости между двумя функциями, вычисленная с использованием расстояния по Хаусдорфу.

Описание задачи:

1. Функции представляют собой массивы точек функции распределения.
2. Функции являются возрастающими.
3. Кодомен функций - отрезок [0, 1].

Необходимо реализовать вычисление меры близости между двумя функциями на основе понятия расстояния по Хаусдорфу.

# Описание алгоритмов

Заголовочные файлы:

1. **<**iostream**>**: для стандартного ввода-вывода.
2. **<**fstream**>**: для работы с файлами.
3. **<**vector**>**: для использования контейнера std::vector.
4. **<**cmath**>**: для использования функции std::abs.
5. **<**stdexcept**>**: для использования исключений.

Определена структура Point, представляющая точку с координатами x и y. Эта структура используется для хранения точек функции распределения.

Функция ReadFunctionFromFile – функция, отвечающая за чтение функции распределения из файла и возвращающая вектор точек функции. Она отвечает за чтение функции распределения из файла и возвращает вектор точек. Функция открывает указанный файл, считывает пары значений x и y из файла и добавляет их в вектор точек. Если файл не может быть открыт, генерируется исключение std::runtime\_error.

std::vector<Point> ReadFunctionFromFile(const std::string& filename) {

std::vector<Point> points;

std::ifstream file(filename);

if (file.is\_open()) {

double x, y;

while (file >> x >> y) {

points.push\_back({ x, y });

}

file.close();

}

else {

throw std::runtime\_error("Error: Unable to open file " + filename);

}

return points;

}

Параметры:

1. filename (тип const std::string&): Имя файла, из которого нужно прочитать функцию распределения.
2. Возвращаемое значение: Вектор точек функции распределения, представленных в виде структуры Point.

Действия:

1. Создается пустой вектор points, который будет содержать точки функции распределения.
2. Открывается файл с указанным именем с помощью объекта std::ifstream.
3. Если файл успешно открыт, происходит чтение пар значений x и y из файла в цикле до конца файла.
4. Каждая считанная пара значений x и y добавляется в вектор points в виде структуры Point.
5. Файл закрывается с помощью метода close().
6. Если файл не может быть открыт, генерируется исключение std::runtime\_error с сообщением об ошибке.
7. Возвращается вектор points с точками функции распределения.

Тестирование:

#include <gtest/gtest.h>

TEST(ReadFunctionFromFileTest, SuccessfulFileRead) {

// Создаем временный файл с тестовыми данными

std::ofstream file("test\_function.txt");

file << "0.1 0.2" << std::endl;

file << "0.3 0.5" << std::endl;

file.close();

// Вызываем функцию для чтения из временного файла

std::vector<Point> function = readFunctionFromFile("test\_function.txt");

// Проверяем, что количество точек функции равно ожидаемому

EXPECT\_EQ(function.size(), 2);

// Проверяем значения точек

EXPECT\_DOUBLE\_EQ(function[0].x, 0.1);

EXPECT\_DOUBLE\_EQ(function[0].y, 0.2);

EXPECT\_DOUBLE\_EQ(function[1].x, 0.3);

EXPECT\_DOUBLE\_EQ(function[1].y, 0.5);

// Удаляем временный файл

std::remove("test\_function.txt");

}

TEST(ReadFunctionFromFileTest, FileNotFound) {

// Вызываем функцию для чтения несуществующего файла

EXPECT\_THROW(readFunctionFromFile("nonexistent\_file.txt"), std::runtime\_error);

}

TEST(ReadFunctionFromFileTest, InvalidFileFormat) {

// Создаем временный файл с некорректным форматом

std::ofstream file("invalid\_function.txt");

file << "0.1 0.2" << std::endl;

file << "0.3" << std::endl; // Неправильное количество значений

file.close();

// Вызываем функцию для чтения из некорректного файла

EXPECT\_THROW(readFunctionFromFile("invalid\_function.txt"), std::runtime\_error);

// Удаляем временный файл

std::remove("invalid\_function.txt");

}

Функция CalculateHausdorffDistance – это функция, вычисляющая расстояние по Хаусдорфу между двумя функциями распределения. Для каждой точки из первой функции она находит минимальное расстояние до точек второй функции и выбирает максимальное из этих расстояний в качестве меры близости. Функция возвращает это максимальное расстояние.

double CalculateHausdorffDistance(const std::vector<Point>& function1, const std::vector<Point>& function2) {

double maxDistance = 0.0;

std::vector<double> distances(function2.size());

for (const Point& p1 : function1) {

for (size\_t i = 0; i < function2.size(); ++i) {

distances[i] = std::abs(p1.y - function2[i].y);

}

double minDistance = \*std::min\_element(distances.begin(), distances.end());

if (minDistance > maxDistance) {

maxDistance = minDistance;

}

}

return maxDistance;

}

Параметры:

1. function1 (тип const std::vector<Point>&): Вектор точек первой функции распределения.
2. function2 (тип const std::vector<Point>&): Вектор точек второй функции распределения.
3. Возвращаемое значение: Максимальное расстояние по Хаусдорфу между функциями.

Действия:

1. Инициализируется переменная maxDistance с начальным значением 0.0, которая будет хранить максимальное расстояние.
2. Создается вектор distances размером, равным размеру второй функции распределения, который будет хранить расстояния между точками первой и второй функций.
3. Для каждой точки p1 из первой функции распределения выполняются следующие действия:
4. В цикле проходится по каждой точке p2 второй функции распределения и вычисляется расстояние между значениями p1.y и p2.y.
5. Полученные расстояния сохраняются в вектор distances.
6. Используется функция std::min\_element, чтобы найти минимальное расстояние в векторе distances, и это значение сохраняется в переменную minDistance.
7. Если minDistance больше maxDistance, то значение maxDistance обновляется.
8. По завершении цикла возвращается значение maxDistance как максимальное расстояние по Хаусдорфу между функциями.

Тестирование:

#include <gtest/gtest.h>

TEST(CalculateHausdorffDistanceTest, DifferentDomains) {

// Создаем две функции с разными доменами

std::vector<Point> function1 = { {0.1, 0.2}, {0.3, 0.5}, {0.5, 0.7} };

std::vector<Point> function2 = { {0.2, 0.3}, {0.4, 0.6}, {0.6, 0.8} };

// Вычисляем расстояние Хаусдорфа

double distance = calculateHausdorffDistance(function1, function2);

// Проверяем, что расстояние равно ожидаемому значению

EXPECT\_DOUBLE\_EQ(distance, 0.2);

}

TEST(CalculateHausdorffDistanceTest, SameDomains) {

// Создаем две функции с одинаковыми доменами

std::vector<Point> function1 = { {0.1, 0.2}, {0.3, 0.5}, {0.5, 0.7} };

std::vector<Point> function2 = { {0.2, 0.4}, {0.3, 0.6}, {0.6, 0.8} };

// Вычисляем расстояние Хаусдорфа

double distance = calculateHausdorffDistance(function1, function2);

// Проверяем, что расстояние равно ожидаемому значению

EXPECT\_DOUBLE\_EQ(distance, 0.3);

}

Функция main – точка входа в программу.

int main(int argc, char\* argv[]) {

if (argc != 3) {

std::cerr << "Error: Two function files are required." << std::endl;

return 1;

}

std::string functionFile1 = argv[1];

std::string functionFile2 = argv[2];

try {

std::vector<Point> function1 = ReadFunctionFromFile(functionFile1);

std::vector<Point> function2 = ReadFunctionFromFile(functionFile2);

double distance = CalculateHausdorffDistance(function1, function2);

std::cout << "Hausdorff distance: " << distance << std::endl;

}

catch (const std::runtime\_error& error) {

std::cerr << error.what() << std::endl;

return 1;

}

return 0;

}

В функции main происходит следующее:

1. Проверяется количество переданных аргументов командной строки. Если ожидается два аргумента (имена файлов с функциями), иначе выводится сообщение об ошибке.
2. Имена файлов для функций сохраняются в переменных functionFile1 и functionFile2.
3. Вызываются функции ReadFunctionFromFile для чтения функций из файлов.
4. Вызывается функция CalculateHausdorffDistance для вычисления расстояния по Хаусдорфу между функциями.
5. Результат выводится на стандартный вывод.

Параметры:

1. argc (тип int): Количество аргументов командной строки.
2. argv (тип char\*[]): Массив аргументов командной строки.
3. Возвращаемое значение: Код возврата программы.

Действия:

1. Проверяется, что количество переданных аргументов командной строки равно 3 (ожидаются два аргумента с именами файлов функций).
2. Имена файлов для функций сохраняются в переменных functionFile1 и functionFile2 типа std::string.
3. Вызывается функция ReadFunctionFromFile для чтения функций распределения из файлов и получения векторов точек function1 и function2.
4. Вызывается функция CalculateHausdorffDistance для вычисления расстояния по Хаусдорфу между функциями function1 и function2, результат сохраняется в переменную distance типа double.
5. Результат выводится на стандартный вывод с помощью объекта std::cout.
6. Если при чтении функций из файлов возникает исключение std::runtime\_error, выводится сообщение об ошибке на стандартный вывод ошибок с помощью объекта std::cerr.
7. Обработка исключений реализована в блоке catch. Если происходит исключение std::runtime\_error, выводится сообщение об ошибке.
8. Возвращается код возврата 0 в случае успешного выполнения программы.

Для обеспечения инвариантности меры от диаметра объединения доменов сравниваемых функций, можно применить следующий подход:

Нормализация функций распределения: Перед вычислением расстояния Хаусдорфа необходимо нормализовать функции распределения путем приведения их областей значений к единичному интервалу [0, 1]. Это можно сделать путем масштабирования значений y каждой функции распределения относительно минимального и максимального значения y в объединенной области значений сравниваемых функций.

Учет диаметра объединения доменов: При нормализации функций распределения также учитывается диаметр объединения доменов. Для этого определяется максимальное значение x среди всех точек в объединенных доменах функций распределения. Затем все значения x в функциях распределения масштабируются путем деления на это максимальное значение. Таким образом, гарантируется, что оба домена функций распределения будут иметь одинаковый размер в интервале [0, 1].

Вычисление расстояния Хаусдорфа: После нормализации функций распределения с учетом диаметра объединения доменов, можно приступить к вычислению расстояния Хаусдорфа между ними. Процесс вычисления остается неизменным, как описано в предыдущих ответах.

С использованием указанного подхода, нормализация функций распределения и учет диаметра объединения доменов обеспечивают инвариантность меры близости по Хаусдорфу относительно размера доменов сравниваемых функций. Это позволяет справедливо сравнивать функции распределения с различными доменами и достичь более надежных результатов в анализе и сравнении таких функций.

# Руководство пользователя

Данное развернутое руководство предоставляет информацию о том, как использовать библиотеку программ для вычисления меры близости между двумя возрастающими кусочно-постоянными функциями распределения на основе расстояния Хаусдорфа.

Подготовьте файлы с функциями распределения, которые хотите сравнить. Файлы должны содержать точки функции распределения, представленные в формате пар значений x и y на каждой строке. Каждая точка должна быть разделена пробелом или табуляцией. Например:

0.1 0.2

0.3 0.5

0.5 0.7

Откройте командную строку или терминал на вашей операционной системе и перейдите в каталог, где находится сохраненный исполняемый файл программы.

Запустите программу, указав имена файлов функций распределения в качестве аргументов командной строки:

**t1.exe function1.txt function2.txt**

После выполнения программы вы увидите результат, представляющий собой вычисленное расстояние Хаусдорфа между двумя функциями распределения.

Интерпретируйте полученное значение в соответствии с вашими требованиями и задачами.

Приведенное выше руководство пользователя предоставляет основные инструкции по использованию скомпилированной программы. Убедитесь, что вы правильно указываете имена файлов функций распределения и правильно интерпретируете полученные результаты.

# Заключение

В результате выполнения данного проекта была разработана библиотека программ, предназначенная для вычисления меры близости между двумя возрастающими кусочно-постоянными функциями распределения. Основной метрикой, использованной для измерения близости, является расстояние Хаусдорфа.

Библиотека программ предоставляет возможность загрузки функций распределения из файлов, представленных в виде массивов точек, а затем вычисления расстояния Хаусдорфа между этими функциями. Реализованные функции readFunctionFromFile и calculateHausdorffDistance позволяют эффективно выполнять чтение данных и вычисление меры близости.

Данная библиотека имеет широкий потенциал применения в различных областях, где требуется анализ и сравнение функций распределения. Она может быть использована для оценки сходства двух распределений, исследования статистических свойств данных, определения степени соответствия моделей и многих других задач.

В ходе разработки библиотеки были учтены основные требования к производительности, точности и надежности. Однако, для обеспечения еще большей надежности и корректности работы, рекомендуется провести дополнительное тестирование на различных тестовых случаях.

Предполагается, что в будущем библиотека будет дорабатываться и расширяться. Возможные направления развития включают добавление новых функций для работы с функциями распределения, включая другие меры близости и операции над функциями. Также планируется улучшение пользовательского интерфейса и предоставление дополнительных возможностей для анализа и визуализации результатов.

В целом, данная библиотека программ представляет собой полезный инструмент для анализа и сравнения функций распределения, способствуя более глубокому пониманию статистических данных и обеспечивая основу для принятия информированных решений в различных областях применения.

# Список использованных источников

1. <https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F>
2. <https://www.boardinfinity.com/blog/vectors-in-c/>
3. <https://www.w3schools.com/cpp/cpp_structs.asp>
4. <https://habr.com/ru/articles/524846/>
5. <https://acm.timus.ru/problem.aspx?space=1&num=1116&locale=ru>
6. <https://habr.com/ru/articles/81226/>

# Приложение

1. Функция распределения – математическая функция, определяющая вероятность случайной величины принять определенное значение или попасть в определенный интервал. В данной программе функция распределения представлена в виде массива точек, где каждая точка представляет значение x и соответствующую вероятность y.
2. Вектор – контейнер, который содержит набор элементов одного типа, размещенных последовательно в памяти. В данной программе векторы используются для хранения точек функций распределения.
3. Структура – пользовательский тип данных, который позволяет объединять несколько элементов различных типов в один объект. В данной программе структура Point используется для представления точек функции распределения с координатами x и y.
4. Расстояние Хаусдорфа – метрика, используемая для измерения различий между двумя множествами точек. В данной программе расстояние Хаусдорфа используется для измерения меры близости между двумя функциями распределения. Формула расстояния Хаусдорфа между двумя множествами A и B выглядит следующим образом: H(A, B) = max(max(d(a, B)), max(d(b, A))), где d(a, B) - расстояние от точки a до ближайшей точки в множестве B, а d(b, A) - расстояние от точки b до ближайшей точки в множестве A.
5. Массив – упорядоченная последовательность элементов одного типа, расположенных в памяти. В данной программе массивы не явно использованы, но векторы, которые используются, можно рассматривать как динамические массивы.
6. Кусочно-постоянная функция – функция, которая состоит из конечного числа отрезков, на каждом из которых она имеет постоянное значение. В данной программе предполагается, что функции распределения являются кусочно-постоянными, то есть значения y меняются только на конечном наборе отрезков.