



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

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ НА ТЕМУ:

«Оценка плотности распределения расстояния между узлами беспроводной сети в зависимости от уровня сигнала между ними с помощью модели гауссовой смеси»

Студент

ИУ7-23М

А.М. Сапожков

(Подпись, дата)

Руководитель

В.И. Солодовников

(Подпись, дата)

2025 г.

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

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
И.В. Рудаков
(И.О.Фамилия)
« ____ » 20 ____ г.

ЗАДАНИЕ на выполнение курсового проекта

по дисциплине Методы машинного обучения

Студент группы ИУ7-23М

Сапожков Андрей Максимович

(Фамилия, имя, отчество)

Тема курсового проекта Оценка плотности распределения расстояния между узлами беспроводной сети в зависимости от уровня сигнала между ними с помощью модели гауссовой смеси.

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание Построить вероятностную модель для оценки плотности распределения расстояния между узлами беспроводной сети в зависимости от уровня сигнала между ними с помощью модели гауссовой смеси.

Оформление курсового проекта:

Расчетно-пояснительная записка на 20-50 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

На защиту проекта должна быть предоставлена презентация, состоящая из 5-15 слайдов.

На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, структура комплекса программ, интерфейс.

Дата выдачи задания « » 2025г.

Руководитель курсового проекта

В.И. Солодовников
(Подпись, дата)

Студент

А.М. Сапожков
(Подпись, дата)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Аналитическая часть	6
1.1. Модель гауссовой смеси	6
1.2. Оценка плотности ядра	10
1.3. Вывод	12
2 Конструкторская часть	13
2.1. ЕМ-алгоритм	13
2.2. Вывод	15
3 Технологическая часть	16
3.1. Средства реализации	16
3.2. Предобработка исходных данных	16
3.3. Обучение модели	18
4 Исследовательская часть	22
4.1. Предобработка исходных данных	22
4.2. Обучение модели	27
4.3. Вывод	29
ЗАКЛЮЧЕНИЕ	30
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	31

ВВЕДЕНИЕ

В телекоммуникациях индикатор силы принимаемого сигнала (Received Signal Strength Indicator, RSSI) — это уровень мощности принимаемого радиосигнала, измеряемый по логарифмической шкале в дБм (dBm, децибел относительно 1 милливатта). Чем выше данное значение, тем качественнее сигнал, принимаемый от некоторого узла сети. Для устройств, работающих по стандартам Wi-Fi и Bluetooth 4.0 [1], RSSI является единственным параметром, позволяющим определить расстояние от устройства до базовой станции или маяка. Для вычисления расстояния используется модель потерь мощности сигнала на пути от источника до приёмника (Path loss model) [2], описываемая уравнением вида

$$P_d = P_0 - 10 \cdot n \cdot \lg\left(\frac{d}{d_0}\right), \quad (1)$$

где

- d — искомое расстояние между источником и приёмником сигнала, м;
- d_0 — опорное (калибровочное) расстояние, м;
- P_d — измеренная мощность сигнала (RSSI), дБм;
- P_0 — мощность сигнала (RSSI), измеренная на расстоянии d_0 , дБм;
- n — коэффициент потери мощности сигнала при распространении в среде, безразмерная величина, зависящая от свойств окружающей среды (например, наличия препятствий на пути прохождения сигнала).

Таким образом, для определения расстояния между двумя узлами беспроводной сети необходимо заранее знать коэффициент потери мощности сигнала n , а также путём калибровки оборудования определить P_0 , выбрав произвольное расстояние d_0 . Параметр n можно оценить методом линейной регрессии. Это позволит получить модель, с помощью которой можно определять расстояние между источником и приёмником по мощности сигнала между ними.

Для оценки прогнозов модели потерь мощности сигнала на пути от источника до приёмника необходима другая модель, учитывающая закон распределения RSSI. Такая модель может найти применение в построении ячеистых сетей (Mesh networks), в частности при разработке и тестировании алгоритмов

функционирования мультиагентных систем Для протокола BLE было показано, что величина RSSI распределена по нормальному закону[3]. Примером модели, учитывающей нормальный закон распределения RSSI, является модель гауссовой смеси (Gaussian Mixture Model, GMM).

Целью данной работы является разработка вероятностной модели для оценки плотности распределения расстояния между узлами беспроводной сети в зависимости от уровня сигнала между ними. Для достижения поставленной цели необходимо решить следующие задачи.

1. Провести анализ предметной области и выбрать модель.
2. Описать алгоритм построения выбранной модели.
3. Реализовать программное обеспечение, которое будет моделировать зависимость расстояния между узлами беспроводной сети от уровня сигнала между ними.
4. Провести оценку плотности распределения расстояния между узлами беспроводной сети в зависимости от уровня сигнала между ними для протокола BLE.

1 Аналитическая часть

В данном разделе будут описаны основные теоретические аспекты, необходимые для решения поставленной задачи.

1.1. Модель гауссовой смеси

При решении задач машинного обучения зачастую необходимо каким-то образом представлять данные. Самый простой способ — взять сами точки как представление данных (пример представлен на рисунке 1.1). Однако этот подход может оказаться бесполезным, если набор данных огромен или если есть необходимость в представлении характеристик данных. При оценке плотности данные представляются компактно, используя функцию плотности из некоторого параметрического семейства, например гауссово или бета-распределение. Также можно рассчитывать среднее значение и дисперсию рассматриваемого набора данных, чтобы компактно представить его с использованием распределения Гаусса. Затем можно использовать параметры этого распределения (среднее значение и дисперсию) для его визуализации, полагая, что оно лежит в основе данных. Иначе говоря, делается предположение о том, что набор данных является реализацией выбранного распределения. [4]

На практике распределение Гаусса имеет ограниченные возможности моделирования. Например, гауссово приближение плотности, сгенерированное на рисунке 1.1, было бы неточным приближением. Далее будет рассмотрено более выразительное семейство распределений, которое можно использовать для оценки плотности данных — модели смеси. Модели смесей могут использоваться для описания распределения $p(x)$ выпуклой комбинацией K простых (базовых) распределений. [4]

$$p(x) = \sum_{k=1}^K \pi_k p_k(x); 0 \leq \pi_k \leq 1; \sum_{k=1}^K \pi_k = 1, \quad (1.1)$$

где компоненты p_k являются членами семейства базовых распределений, например Гаусса, Бернулли или гамма-распределение, а π_k — веса смеси. Модели смесей более выразительны, чем соответствующие базовые распределения, поскольку они допускают мультимодальные представления данных, то есть они могут описывать наборы данных с несколькими «кластерами», как в примере

на рисунке 1.1. [4]

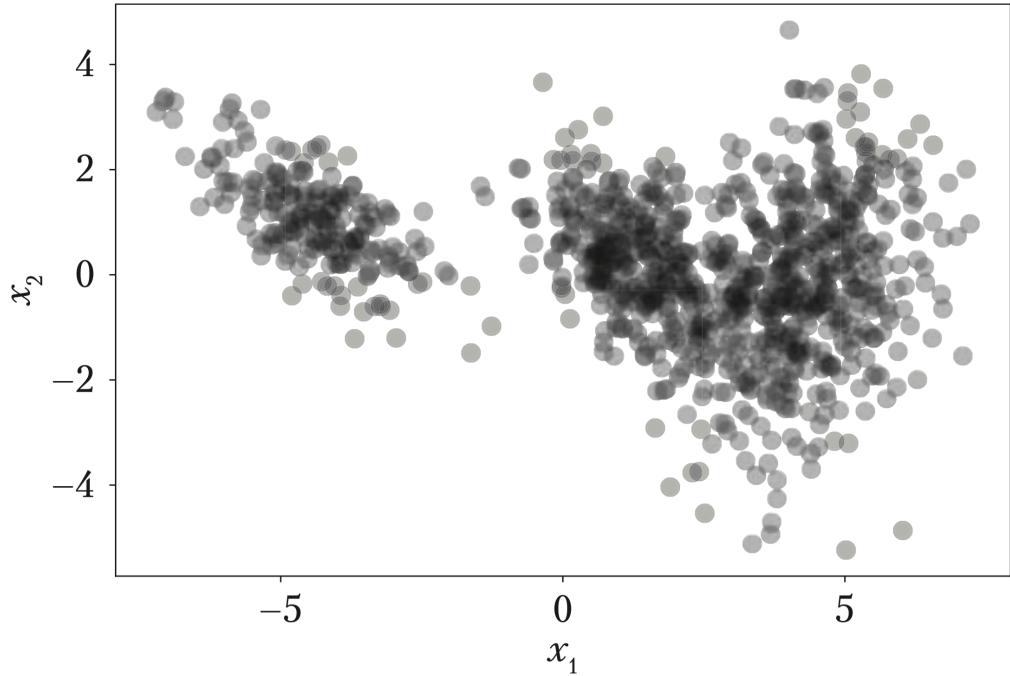


Рис. 1.1 – Двумерный набор данных, который не может быть осмысленно представлен с помощью гауссиана

Рассмотрим модели гауссовой смеси (Gaussian mixture models — GMM), где базовые распределения являются гауссианами. При построении смеси для исходного набора данных необходимо стремиться к максимизации вероятности параметров модели для обучения GMM. Однако в отличие от других приложений для представления данных (таких, как линейная регрессия или РСА), нельзя найти максимум функции правдоподобия в явной форме. Вместо этого предлагаются системы уравнений, которую можно решить только итерационно. [4]

Модель гауссовой смеси (Gaussian mixture model, GMM) — это модель плотности, в которой комбинируется конечное число K гауссовых распределений $N(x|\mu_k, \Sigma_k)$, так чтобы

$$p(x|\theta) = \sum_{k=1}^K \pi_k N(x|\mu_k, \Sigma_k); 0 \leq \pi_k \leq 1; \sum_{k=1}^K \pi_k = 1, \quad (1.2)$$

где $\theta := \mu_k, \Sigma_k, \pi_k : k = 1, \dots, K$ является представлением совокупности всех параметров модели. Эта выпуклая комбинация распределений Гаусса даёт значительно большую гибкость при моделировании сложных многомерных плотностей, чем простое гауссово распределение. [4] На рисунке 1.2 показаны взве-

шенные компоненты и плотность смеси, которая представлена как

$$p(x|\theta) = 0.5N(x| - 2, \frac{1}{2}) + 0.2N(x|1, 2) + 0.3N(x|4, 1). \quad (1.3)$$

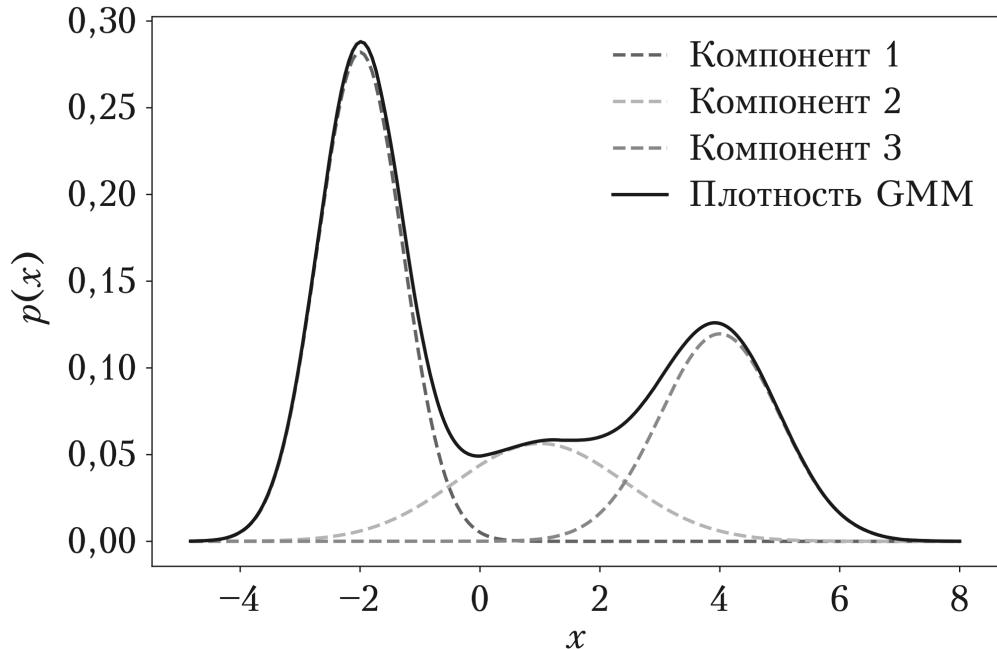


Рис. 1.2 – Модель гауссовой смеси, которая представляет собой выпуклой комбинацией распределений Гаусса и является более выразительной, чем любой отдельный её компонент

Предположим, нам имеется набор данных $X = x_1, \dots, x_N$, где $x_n, n = 1, \dots, N$, взяты независимыми и одинаково распределёнными из неизвестного распределения $p(x)$. Наша цель — найти наиболее точное приближение/представление этого неизвестного распределения $p(x)$ с помощью GMM с K компонентов смеси. Параметрами GMM являются K -средние μ_k , ковариации Σ_k и веса смеси π_k . Все эти свободные параметры суммируются в $\theta := \pi_k, \mu_k, \Sigma_k : k = 1, \dots, K$. [4]

Далее будет подробно описана методика получения оценки максимального правдоподобия θ_{ML} параметров модели θ . [4] Начнём с записи вероятности, то есть прогнозируемого распределения обучающего набора данных. Для этого запишем факторизованную вероятность для независимых и одинаково распределённых случайных величин

$$p(X|\theta) = \prod_{n=1}^N p(x_n|\theta), \quad p(x_n|\theta) = \sum_{k=1}^K \pi_k N(x_n|\mu_k, \Sigma_k), \quad (1.4)$$

где каждый член вероятности $p(x_n|\theta)$ является функцией плотности гауссовой смеси. Тогда можно определить логарифмическое правдоподобие как

$$\log p(X|\theta) = \sum_{n=1}^N \log p(x_n|\theta) = \sum_{n=1}^N \log \sum_{k=1}^K \pi_k N(x_n|\mu_k, \Sigma_k). \quad (1.5)$$

Далее необходимо найти значения параметров, которые максимизируют логарифмическое правдоподобие θ_{ML} , определённое в (1.5). Затем можно вычислить градиент $dL/d\theta$ логарифма правдоподобия относительно параметров модели θ , приравняв его к нулю и решить относительно θ . Однако в данном случае для рассматриваемой оценки максимального правдоподобия, нельзя получить решение в явной форме. Тем не менее, можно использовать итерационную схему, чтобы с достаточной точностью найти параметры модели θ_{ML} . Ключевая идея состоит в том, чтобы обновлять один параметр модели за одну итерацию, оставляя другие параметры неизменными. [4]

Определяем количество

$$r_{nk} := \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n|\mu_j, \Sigma_j)} \quad (1.6)$$

как ответственность k -го компонента смеси за n -ю точку данных. Ответственность r_{nk} k -го компонента смеси для точки данных x_n пропорциональна правдоподобию

$$p(x_n|\pi_k, \mu_k, \Sigma_k) = \pi_k N(x_n|\mu_k, \Sigma_k) \quad (1.7)$$

компоненты смеси с учётом параметров точки исходных данных¹. Следовательно, компоненты смеси несут большую ответственность за точку данных, когда точка данных допустима для конкретного компонента смеси. Стоит отметить, что $r_n := [r_{n1}, \dots, r_{nK}]^T \in \mathbb{R}_k$ является нормированным вектором вероятности, то есть $\sum_k r_{nk} = 1 \geq 0$. Этот вектор вероятности распределяет вероятностную массу между K компонентами смеси, и можно рассматривать r_n как «мягкое присвоение» x_n компонентам K смеси. Следовательно, ответственность r_{nk} из (1.6) представляет собой вероятность того, что x_n был сгенерирован k -м компонентом смеси². [4]

¹ r_n следует распределению Больцмана-Гиббса.

²Ответственность r_{nk} — это вероятность того, что k -й компонент смеси сгенерировал n -ю точку данных.

Далее будет дано определение обновления параметров модели μ_k, Σ_k, π_k для заданных ответственостей. Будет показано, что все уравнения обновления зависят от ответственостей, что делает невозможным решение задачи оценки максимального правдоподобия в явном виде. Однако для имеющихся ответственостей можно обновлять один параметр модели за один шаг, оставляя другие неизменными. После этого должен производиться пересчёт ответственности. Итерация этих двух шагов в конечном итоге приведёт к локальному оптимуму и является конкретным воплощением алгоритма ЕМ. [4]

Обновление средних параметров GMM $\mu_k, k = 1, \dots, K$, определяется выражением

$$\mu_k^{new} = \frac{\sum_{n=1}^N r_{nk} x_n}{\sum_{n=1}^N r_{nk}}, \quad (1.8)$$

где ответственности r_{nk} определены в (1.6). [4]

Обновление параметров ковариации $\Sigma_k, k = 1, \dots, K$, GMM определяется выражением

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^N r_{nk} (x_n - \mu_k)(x_n - \mu_k)^T; N_k = \sum_{n=1}^N r_{nk}, \quad (1.9)$$

где r_{nk} определены в (1.6) соответственно. [4]

Вес смеси GMM обновляется как

$$\pi_k^{new} = \frac{N_k}{N}, k = 1, \dots, K, \quad (1.10)$$

где N — количество точек данных, а N_k определено в (1.9). [4]

Можно определить вес смеси в (1.10) как отношение общей ответственности k -го кластера к количеству точек данных. Поскольку $N = \sum_k N_k$, количество точек данных также можно интерпретировать как общую ответственность всех компонентов смеси вместе, так что π_k — относительная важность k -го компонента смеси для набора данных. [4]

1.2. Оценка плотности ядра

Оценка плотности ядра (Kernel density estimation, KDE) [5; 6], является непараметрическим способом оценки плотности. Учитывая N независимых и одинаково распределённых выборок, оценщик плотности ядра представляет ле-

жащее в основе распределение как

$$p(x) = \frac{1}{Nh} \sum_{n=1}^N k\left(\frac{x - x_n}{h}\right), \quad (1.11)$$

где k — функция ядра, то есть неотрицательная функция, которая интегрируется до 1, а $h > 0$ — свободный параметр сглаживания полосы пропускания (bandwidth), который играет такую же роль, как размер ячейки в гистограммах. Пример зависимости вида гистограммы рассматриваемой случайной величины от различных значений параметра h представлен на рисунке 1.3. [4]

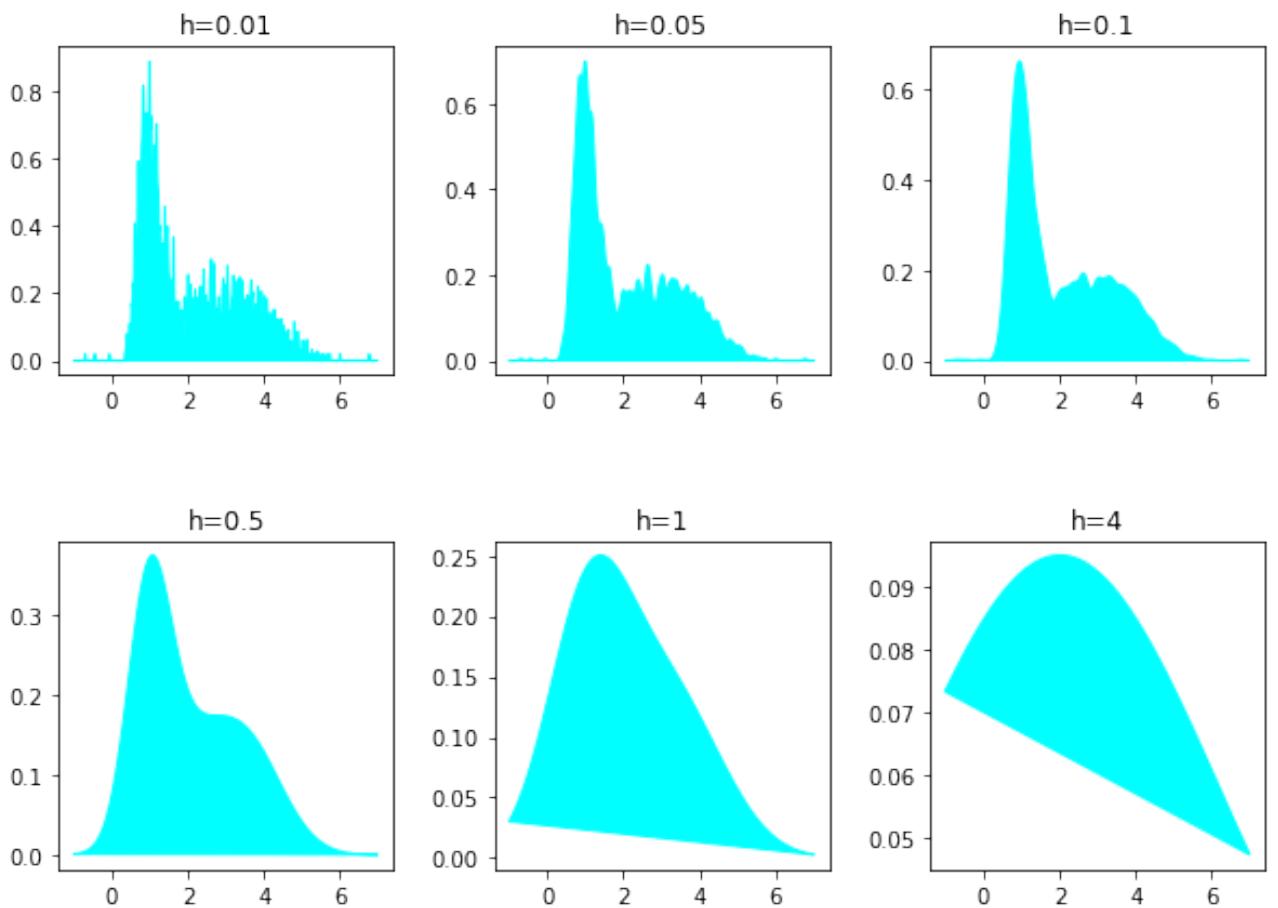


Рис. 1.3 – Пример зависимости вида гистограммы от различных значений параметра h

Стоит отметить, что ядро помещается в каждую точку данных x_n в наборе данных. Обычно используемые ядерные функции — это равномерное распределение и распределение Гаусса. Также могут использоваться однородная, треугольная, бивзвешенная, тризвзвешенная, Епанечникова, нормальная и другие ядерные функции. [4]

Оценки плотности ядра тесно связаны с гистограммами, однако по сравнению с ними у появляется возможность выбирать подходящее ядро, чтобы варьировать оценки плотности. На рисунке 1.4 показана разница между гистограммой и оценкой плотности ядра (с ядром в форме Гаусса) для некоторого набора данных из 250 точек данных. [4]

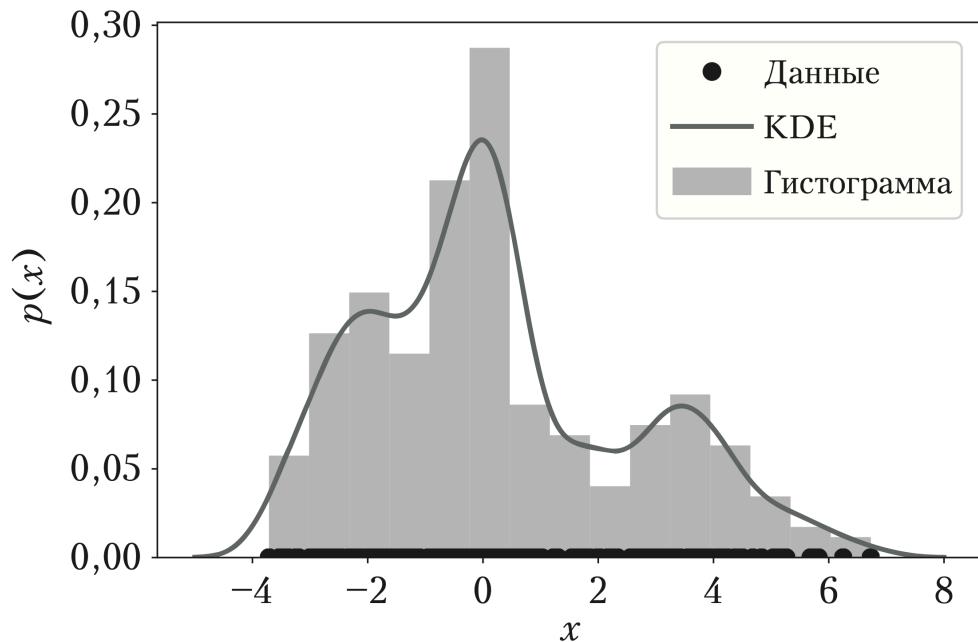


Рис. 1.4 – Пример сравнения гистограммы и KDE

1.3. Вывод

В данном разделе были рассмотрены модели гауссовой смеси (GMM) и модели оценки плотности (KDE). Для дальнейшего моделирования зависимости расстояния между узлами беспроводной сети от уровня сигнала между ними предлагается использовать модель гауссовой смеси, так как она учитывает нормальную природу рассматриваемого явления и обладает меньшей вычислительной сложностью по сравнению с KDE.

2 Конструкторская часть

В данном разделе будут приведено описание алгоритма обучения модели.

2.1. ЕМ-алгоритм

К сожалению, обновления в (1.8), (1.9) и (1.10) не представляют собой решение в явной форме для обновлений параметров μ_k, Σ_k, π_k модели смеси, поскольку ответственности r_{nk} зависят от этих параметров сложным способом. Однако полученные выражения предлагают простую итерационную схему для поиска решения задачи оценки параметров с помощью максимального правдоподобия. Алгоритм максимизации ожидания (алгоритм EM — expectation maximization) представляет собой общую итеративную схему для вычисления параметров (максимального правдоподобия или МАР) в моделях смесей и, в более общем смысле, в моделях со скрытыми переменными. [7]

Для вычисления параметров модели гауссовой смеси необходимо выбрать начальные значения для μ_k, Σ_k, π_k и изменять их до сходимости между Е-шагом и М-шагом:

- Е-шаг: оценка ответственостей r_{nk} (апостериорная вероятность того, что точка данных n принадлежит компоненту смеси k).
- М-шаг: использование обновлённых ответственостей для вычисления новой оценки параметров μ_k, Σ_k, π_k .

Каждый шаг в алгоритме ЕМ увеличивает логарифмическую функцию правдоподобия. Для сходимости можно напрямую проверить логарифмическую вероятность или параметры, полученные на текущей итерации. Конкретная реализация алгоритма ЕМ для оценки параметров GMM выглядит следующим образом. [8]

1. Инициализация μ_k, Σ_k, π_k .
2. Е-шаг: оценка ответственостей r_{nk} для каждой точки данных x_n , используя текущие параметры μ_k, Σ_k, π_k :

$$r_{nk} := \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n | \mu_j, \Sigma_j)} \quad (2.1)$$

3. М-шаг: повторная оценка параметров μ_k, Σ_k, π_k с использованием текущих ответственостей r_{nk} из Е-шага:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N r_{nk} x_n; \quad (2.2)$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N r_{nk} (x_n - \mu_k)(x_n - \mu_k)^T; \quad (2.3)$$

$$\pi_k = \frac{N_k}{N}. \quad (2.4)$$

Несмотря на то, что каждая итерация ЕМ-алгоритма увеличивает значение логарифмической функции правдоподобия, нет никаких гарантий, что алгоритм сходится к решению с максимальным правдоподобием. Возможно, что ЕМ-алгоритм сходится к локальному максимуму логарифма функции правдоподобия. Для уменьшения риска попадания в локальный оптимум можно выполнить ЕМ-алгоритм несколько раз, используя различные инициализации параметра θ . [9; 10]

На рисунке 2.1 представлен результат применения алгоритма ЕМ к двумерному набору данных, показанному на рисунке 1.1 рисунке, с $K = 3$ компонентами смеси. Рисунок 2.1 иллюстрирует некоторые шаги алгоритма ЕМ и показывает отрицательную логарифмическую вероятность как функцию итерации ЕМ. На рисунке 2.1(a) показана соответствующая окончательная подгонка GMM. Рисунок 2.1(b) визуализирует окончательные ответственности компонентов смеси для точек данных. Набор данных окрашен в соответствии с ответственностями компонентов смеси, когда ЕМ-алгоритм сходится. Хотя один компонент смеси явно отвечает за данные слева, перекрытие двух кластеров данных справа могло быть создано двумя компонентами смеси. Становится ясно, что есть точки данных, которые не могут быть однозначно присвоены одному компоненту, так что ответственность этих двух кластеров за эти точки составляет около 0.5. [4]

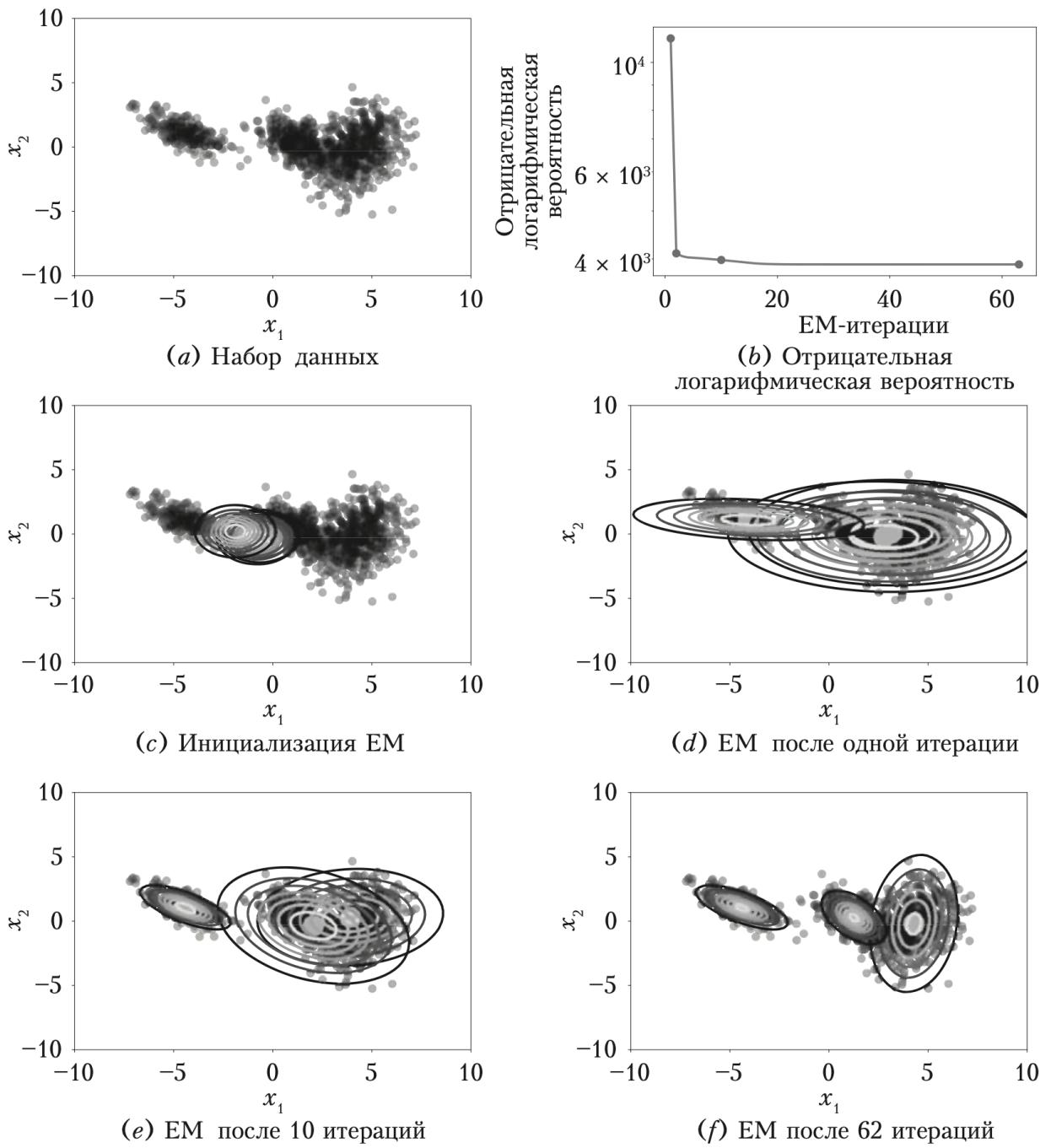


Рис. 2.1 – Иллюстрация ЕМ-алгоритма для обучения модели гауссовой смеси с тремя компонентами к двумерному набору данных

2.2. Вывод

В данном разделе был описан ЕМ-алгоритм обучения модели гауссовой смеси. При фиксированном числе итераций алгоритма его времененная трудоёмкость линейно зависит от числа точек в обучающем наборе данных и от числа компонент смеси.

3 Технологическая часть

В данном разделе будут описаны детали реализации ПО, а также приведены листинги кода.

3.1. Средства реализации

В качестве языка программирования для реализации алгоритмов был выбран язык Python ввиду наличия библиотек для проведения расчётов и обучения моделей, таких как numpy и sklearn.

3.2. Предобработка исходных данных

В листингах 3.1 и 3.2 представлены функции, реализующие предобработку исходных данных, включающую фильтрацию с помощью фильтра Калмана.

Листинг 3.1 – Функции для фильтрации исходных данных с помощью фильтра Калмана

```
1      import numpy as np
2
3      class KalmanFilter(object):
4          def __init__(self, F = None, B = None, H = None, Q = None,
5                       R = None, P = None, x0 = None,
6                       ):
7              if(F is None or H is None):
8                  raise ValueError("Set proper system dynamics.")
9              self.n = F.shape[1]
10             self.m = H.shape[1]
11             self.F = F
12             self.H = H
13             self.B = 0 if B is None else B
14             self.Q = np.eye(self.n) if Q is None else Q
15             self.R = np.eye(self.n) if R is None else R
16             self.P = np.eye(self.n) if P is None else P
17             self.x = np.zeros((self.n, 1)) if x0 is None else x0
18
19
20             def predict(self, u = 0):
21                 self.x = np.dot(self.F, self.x) + np.dot(self.B, u)
22                 self.P = np.dot(np.dot(self.F, self.P), self.F.T) + self.Q
23                 return self.x
24
25             def update(self, z):
26                 y = z - np.dot(self.H, self.x)
27                 S = self.R + np.dot(self.H, np.dot(self.P, self.H.T))
```

```

28         K = np.dot(np.dot(self.P, self.H.T), np.linalg.inv(S))
29         self.x = self.x + np.dot(K, y)
30         I = np.eye(self.n)
31         self.P = np.dot(np.dot(I - np.dot(K, self.H), self.P),
32                         (I - np.dot(K, self.H)).T) + np.dot(np.dot(K, self.R), K.T)
33
34     def filter(data):
35         dt = 1.0/60
36         F = np.array([[1, dt, 0], [0, 1, dt], [0, 0, 1]])
37         H = np.array([1, 0, 0]).reshape(1, 3)
38         Q = np.array([
39             [0.05, 0.05, 0.0],
40             [0.05, 0.05, 0.0],
41             [0.0, 0.0, 0.0],
42         ])
43         R = np.array([0.5]).reshape(1, 1)
44         kf = KalmanFilter(F = F, H = H, Q = Q, R = R)
45         predictions = []
46         for z in data:
47             prediction = np.dot(H, kf.predict())[0]
48             prediction = int(round(prediction[0], 0))
49             predictions.append(prediction)
50             kf.update(z)
51
52     return predictions

```

Листинг 3.2 – Функция считывания и предобработки исходных данных

```

1     def read_input():
2         dist_range = np.arange(0.1, 5+0.001, 0.1)
3         rssi = []
4         rssi_filtered = []
5         lost_total = 0
6         rssi_min, rssi_max = None, None
7         print('input: ')
8         for dist in dist_range:
9             data, lost_count = read_rssi_data(round(dist, 1))
10            lost_total += lost_count
11            filtered_data = filter(data)
12            rssi.append(data)
13            rssi_filtered.append(filtered_data)
14            if not rssi_min or min(data) < rssi_min:
15                rssi_min = min(data)
16            if not rssi_max or max(data) > rssi_max:
17                rssi_max = max(data)
18            total_length = sum([len(data) for data in rssi])
19            loss = round(100*lost_total/total_length, 2)
20            print(f'total: {total_length} + {lost_total} invalid ({loss}%)')
21            rssi_min, rssi_max = rssi_min - 10, rssi_max + 10

```

```

22     rssi_range = np.arange(rssi_min, rssi_max + 1)
23     filtered_length = sum([len(data) for data in rssi_filtered])
24     rssi_filtered = [
25         [val for val in row if rssi_min <= val <= rssi_max]
26         for row in rssi_filtered
27     ]
28     new_filtered_length = sum([len(data) for data in rssi_filtered])
29     loss = filtered_length - new_filtered_length / filtered_length
30     loss = round(100 * (loss, 2))
31     print(f'after filtration: {new_filtered_length} entries')
32     print(f'{filtered_length - new_filtered_length} invalid ({loss}%)')
33     return rssi, rssi_filtered, dist_range, rssi_range

```

3.3. Обучение модели

На листингах 3.3 и 3.4 представлены функции для оптимизации гиперпараметров алгоритма обучения модели гауссовой смеси. Для предотвращения явления переобучения, алгоритм обучения при схожей точности аппроксимации исходного распределения и точности построения регрессии у двух моделей отдавал предпочтение той, которая имеет меньшее число компонент в смеси. Именно число компонент смеси определяет обобщающую способность модели (чем меньше компонент, тем больше обобщающая способность).

Листинг 3.3 – Функция построения модели линейной регрессии для определения параметра модели потерь мощности сигнала на пути от источника до приемника

```

1      import numpy as np
2      from sklearn.linear_model import LinearRegression
3      from input import read_input
4      from matstat import calculate_distributions, calculate_mean_vars
5
6      def make_regression(rssi_range, dist_range, rssi_frequencies, \
7          rssi_0, dist_0):
8          rssi_grid, dist_grid = np.meshgrid(rssi_range, dist_range)
9          rssi_data = rssi_grid.ravel().reshape(-1, 1)
10         dist_data = dist_grid.ravel().reshape(-1, 1)
11         weights = rssi_frequencies.reshape(-1, 1).ravel()
12         dist_transformed = np.log10(dist_data / dist_0) # log10(d/d0)
13         rssi_transformed = rssi_data - rssi_0 # P - P0
14         model = LinearRegression(fit_intercept=False)
15         model.fit(dist_transformed, rssi_transformed, sample_weight=weights)
16         k = model.coef_[0][0]
17         n = -k / 10
18         return n

```

```

19
20 def main():
21     rssi , rssi_filtered , dist_range , rssi_range = read_input()
22     _, rssi_filtered_frequencies = calculate_distributions(rssi , \
23         rssi_filtered , rssi_range)
24     means_filtered , _, vars_filtered , _ = \
25         calculate_mean_vars(rssi_filtered)
26     best_point = np.argmin( vars_filtered [5:] ) + 5
27     dist_0 = dist_range [best_point]
28     rssi_0 = means_filtered [best_point]
29     print(f"d0: {dist_0} ")
30     print(f"P0: {rssi_0} (var: {vars_filtered [best_point] :.2f} )")
31     n = make_regression(rssi_range , dist_range , \
32         rssi_filtered_frequencies , rssi_0 , dist_0)
33     print(f"n: {n:.10f}")
34

```

Листинг 3.4 – Функция подбора гиперпараметров для ЕМ-алгоритма обучения модели гауссовой смеси

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.mixture import GaussianMixture
4 from scipy.stats import multivariate_normal
5 from input import read_input
6 from matstat import calculate_distributions , calculate_mean_vars
7 from regression import make_regression
8 from sklearn.metrics import mean_squared_error
9
10 def create_samples_from_weights(X_grid , Y_grid , W):
11     W_norm = W / W.sum()
12     indices = np.random.choice(np.arange(len(W_norm.ravel())), ,
13         size=1000000,
14         p=W_norm.ravel())
15     x_samples = X_grid.ravel()[indices]
16     y_samples = Y_grid.ravel()[indices]
17     return np.column_stack([x_samples , y_samples])
18
19 def main():
20     rssi , rssi_filtered , dist_range , rssi_range = read_input()
21     _, rssi_filtered_frequencies = \
22         calculate_distributions(rssi , rssi_filtered , rssi_range)
23     means_filtered , _, vars_filtered , _ = \
24         calculate_mean_vars(rssi_filtered)
25     best_point = np.argmin( vars_filtered [5:] ) + 5
26     dist_0 = dist_range [best_point]
27     rssi_0 = means_filtered [best_point]
28     print(f"d0: {dist_0} ")

```

```

29     print(f"P0: {rss_i_0}    (var: {vars_filtered[best_point]:.2f})")
30     best_n = make_regression(rssi_range, dist_range, \
31                               rss_i_filtered_frequencies, rss_i_0, dist_0)
32     print(f"n: {best_n:.10f}")
33     X, Y = np.meshgrid(rssi_range, dist_range)
34     W = rss_i_filtered_frequencies
35     W /= W.sum()
36     samples = create_samples_from_weights(X, Y, W)
37     Z_test = []
38     for sample in samples:
39         Z_test.append(W[np.where(np.isclose(dist_range, sample[1])) , \
40                         np.where(np.isclose(rssi_range, sample[0]))][0][0])
41     Z_test = np.array(Z_test)
42
43     best_gmm = None
44     for n_components in range(1, 11):
45         if best_gmm is not None and \
46             n_components - best_gmm['n_components'] > 3:
47             break
48     print(f'n_components: {n_components} ... ', end=' ')
49     for _ in range(10):
50         gmm = GaussianMixture(n_components=n_components, \
51                               max_iter=1000)
52         gmm.fit(samples)
53         weights = gmm.weights_
54         means = gmm.means_
55         covariances = gmm.covariances_
56         Z = np.sum([
57             weights[i] * multivariate_normal(
58                 means[i], covariances[i],
59             ).pdf(
60                 np.vstack([X.ravel(), Y.ravel()]).T,
61             )
62             for i in range(n_components)
63         ], axis=0).reshape(X.shape)
64         Z_predict = np.exp(gmm.score_samples(samples))
65         mse = mean_squared_error(Z_test, Z_predict)
66         n = make_regression(rssi_range, dist_range, Z, \
67                             rss_i_0, dist_0)
68         if best_gmm is not None:
69             mse_diff = mse - best_gmm['mse']
70             n_diff_diff = abs(n - best_n) - \
71                           abs(best_gmm['n'] - best_n)
72         if best_gmm is None or \
73             mse_diff < -5e-5 and n_diff_diff < 1e-3 or \
74             n_diff_diff < -1e-4 and mse_diff < 1e-4:
75             best_gmm = {
76                 'n_components': n_components,

```

```
77      'weights': weights,
78      'means': means,
79      'covariances': covariances,
80      'Z': Z,
81      'n': n,
82      'mse': mse,
83      }
84  print(best_gmm)
```

4 Исследовательская часть

В данном разделе будет проведено обучение модели гауссовой смеси и сравнение с исходным набором данных.

4.1. Предобработка исходных данных

В качестве набора данных для обучения модели гауссовой смеси были проведены измерения мощности сигнала между источником и приёмником, взаимодействующими по протоколу BLE, на расстояниях от 0.1 до 5 метров. На рисунках 4.1-4.3 представлена визуализация результатов измерений до и после фильтрации.

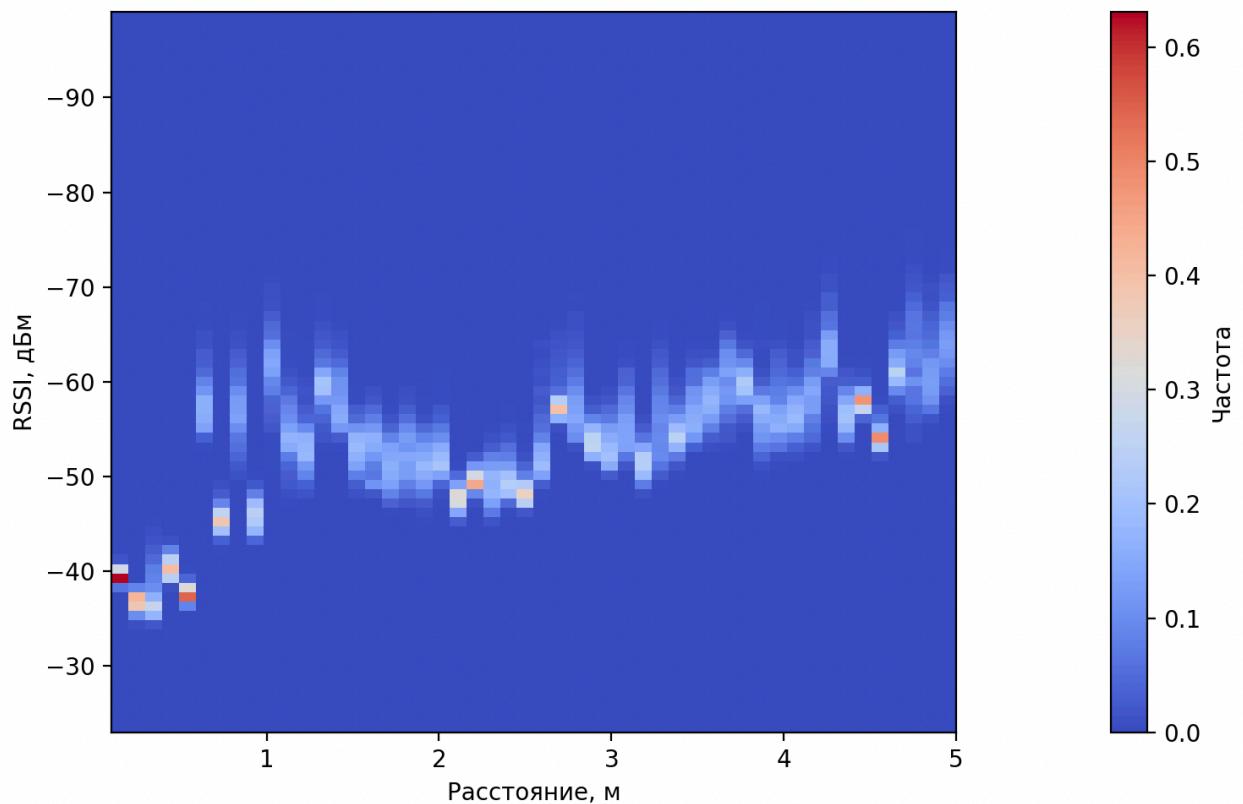
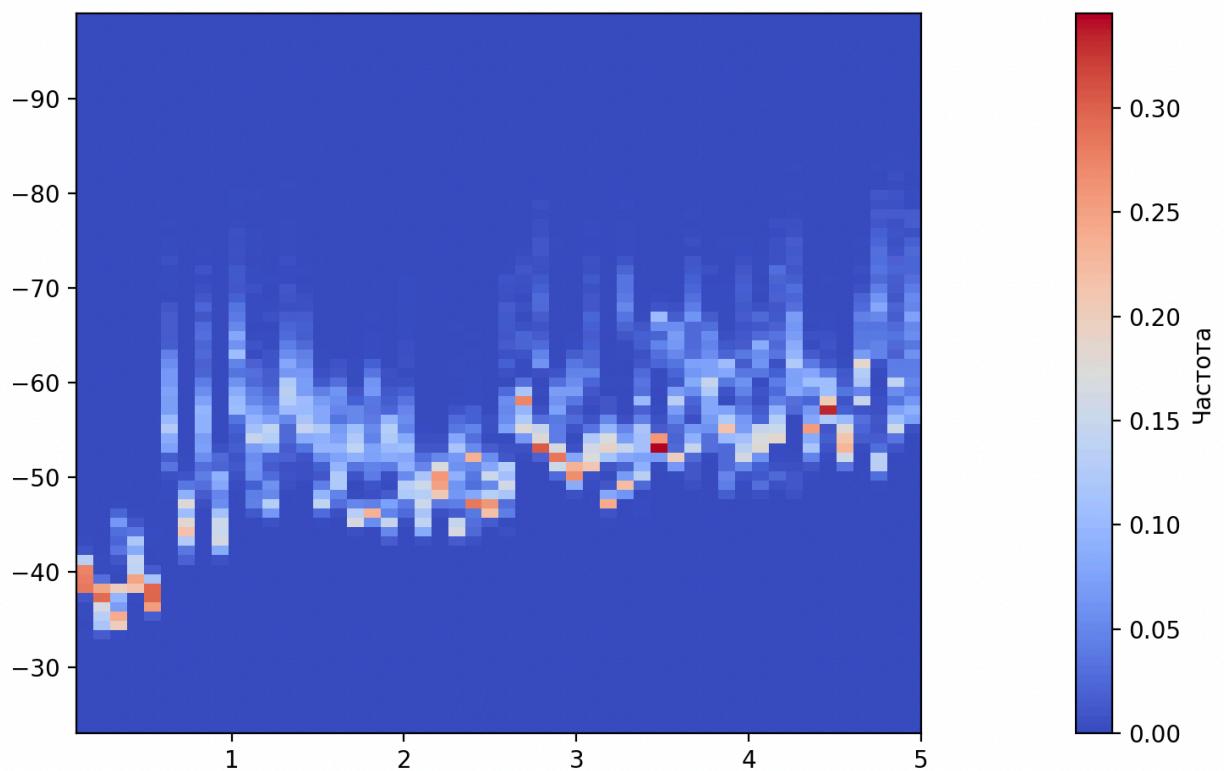


Рис. 4.1 – Тепловые карты, составленные по датасету до фильтрации (сверху) и после неё (снизу)

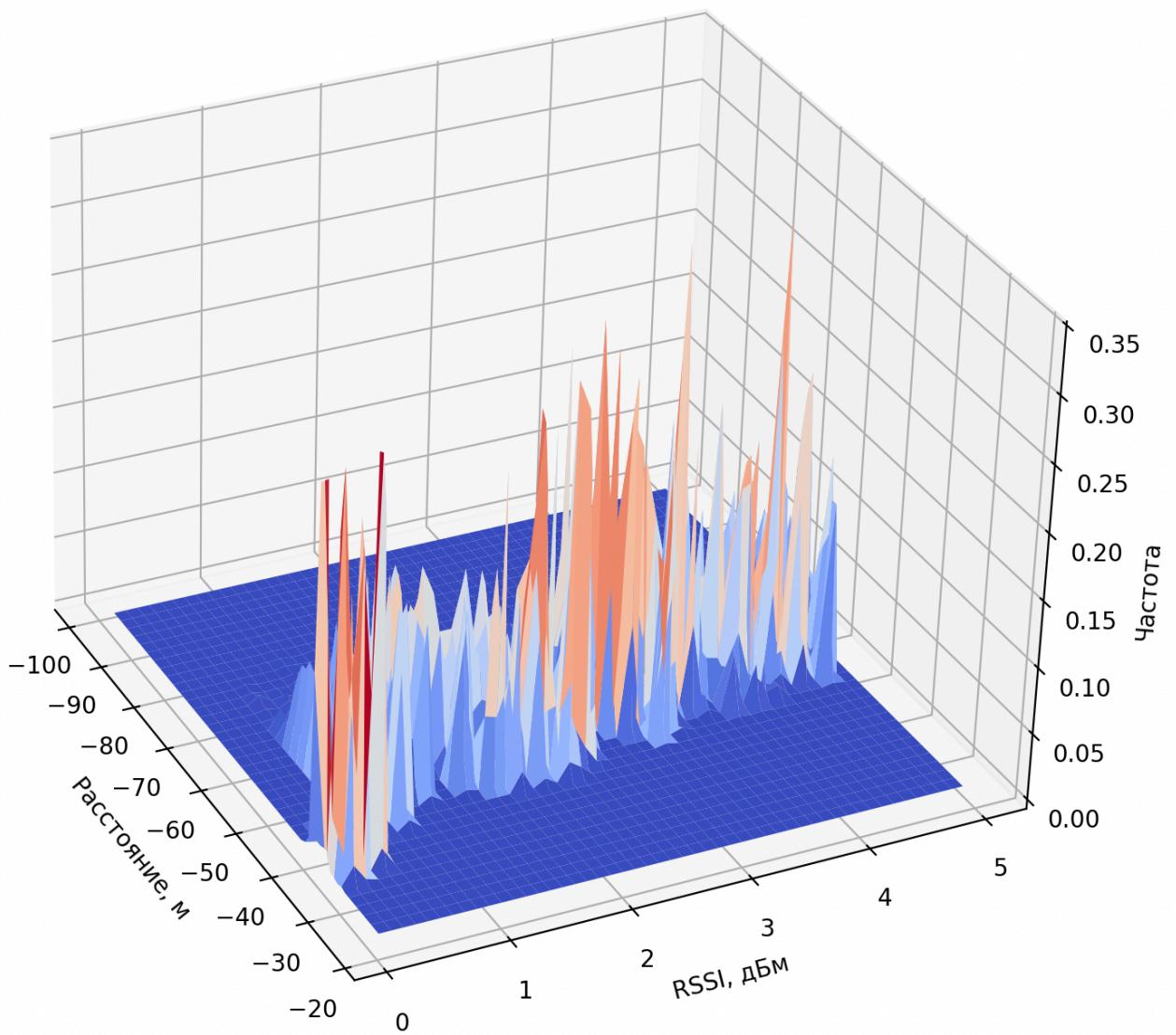


Рис. 4.2 – Гистограмма распределения мощности сигнала в зависимости от расстояния между источником и приёмником до фильтрации

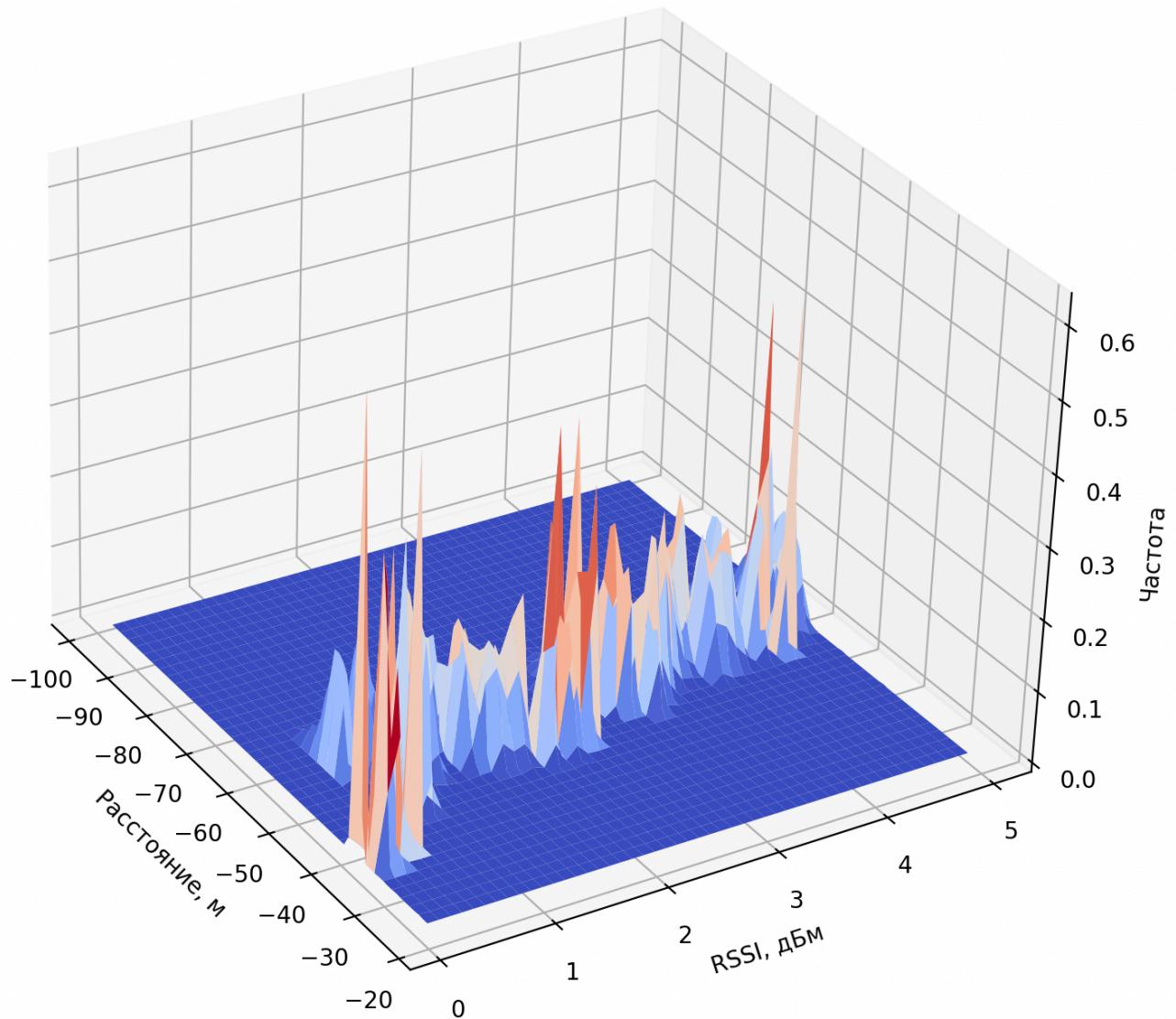


Рис. 4.3 – Гистограмма распределения мощности сигнала в зависимости от расстояния между источником и приёмником после фильтрации

На рисунках 4.4 и 4.5 представлено сравнение средних и медианных значений мощности сигнала, а также их отклонений, до и после фильтрации. Сравнение показывает, что применение фильтра Калмана позволило снизить отклонения мощности сигнала, не изменив при этом средние и медианные значения.

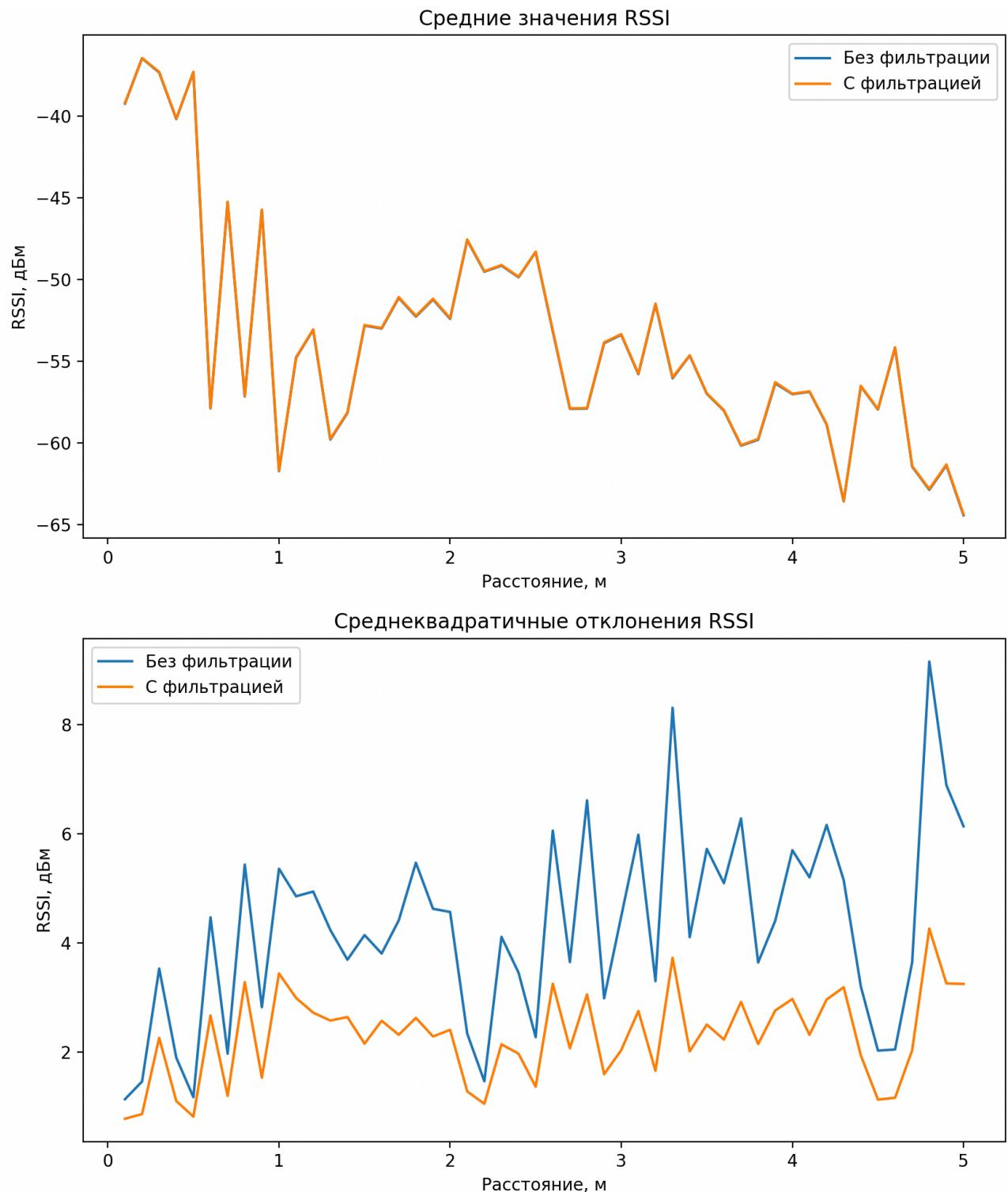


Рис. 4.4 – Сравнение средних значений мощности сигнала до и после фильтрации

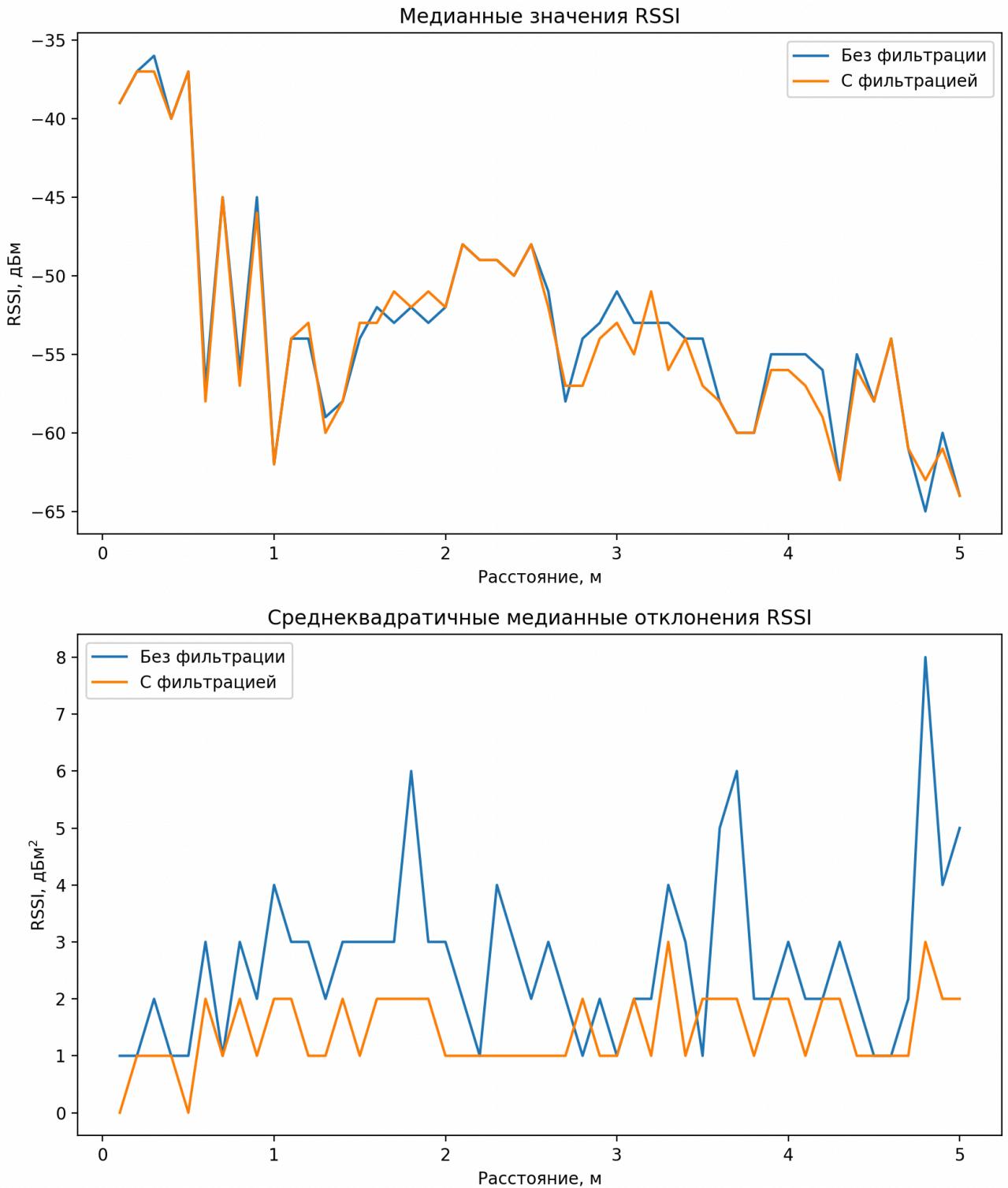


Рис. 4.5 – Сравнение медианных значений мощности сигнала до и после фильтрации

4.2. Обучение модели

На рисунках 4.1-4.3 представлена визуализация оптимальной гауссовой смеси. Характеристики полученной модели:

- Число компонент: 8;

- Значение метрики MSE при сравнении с исходным распределением: 0.0002907611;
- Отклонение коэффициента регрессии в модели потерь мощности сигнала на пути от источника до приёмника: 0.0058126679.

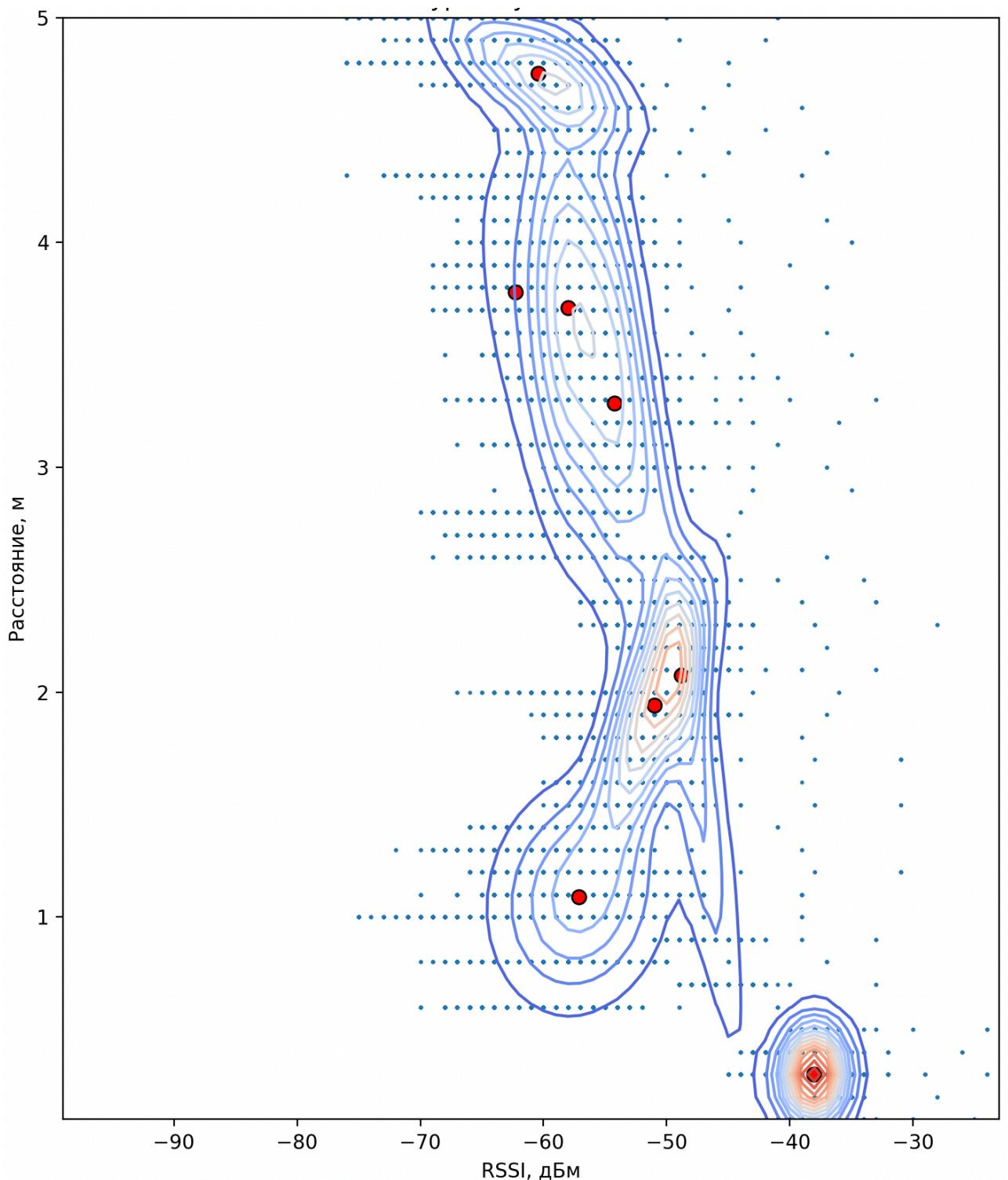


Рис. 4.6 – Контуры оптимальной гауссовой смеси

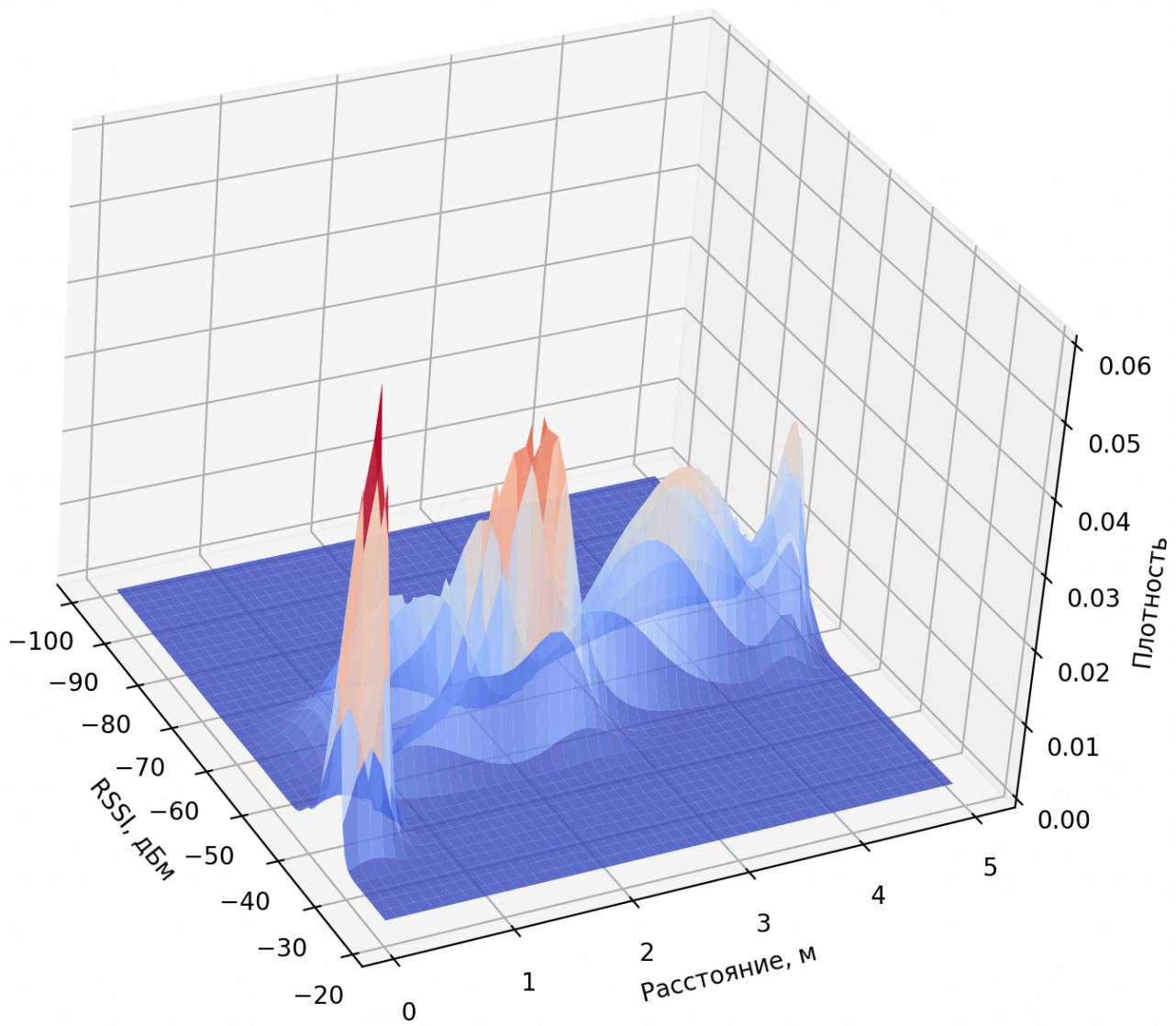


Рис. 4.7 – Гистограмма оптимальной гауссовой смеси

4.3. Вывод

В данном разделе было проведено обучение модели гауссовой смеси и сравнение с исходным набором данных. В результате получилась модель из 8 компонент, значение метрики MSE составило 0.0002907611.

ЗАКЛЮЧЕНИЕ

В рамках курсовой работы была разработана вероятностная модель для оценки плотности распределения расстояния между узлами беспроводной сети в зависимости от уровня сигнала между ними. В ходе выполнения данной работы были решены следующие задачи.

1. Проведён анализ предметной области и выбрана модель.
2. Описан алгоритм построения выбранной модели.
3. Реализовано программное обеспечение для моделирования зависимости расстояния между узлами беспроводной сети от уровня сигнала между ними.
4. Проведена оценка плотности распределения расстояния между узлами беспроводной сети в зависимости от уровня сигнала между ними для протокола BLE.

Таким образом, все поставленные задачи были выполнены, поставленная цель достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Таненбаум Э., Фимстер Н., Уэзеролл Д. Компьютерные сети. — Питер, 2023. — С. 992.
2. Oestges C., Qutin F. Inclusive Radio Communications for 5G and Beyond. — Academic Press, 2021. — С. 221—252. — ISBN 9780128205822.
3. Сапожков А. Определение закона распределения уровня сигнала в беспроводных сетях и констант для расчёта расстояния между узлами // Математические методы в технологиях и технике. — 2025. — № 1. — С. 42—49. — ISSN 2712-8873.
4. Дайзенрот Марк Питер, Альдо Фейзал А., Чен Сунь Он. Математика в машинном обучении. — Питер, 2025. — С. 512. — ISBN 978-5-4461-1788-8.
5. Neal R. M., Hinton G. E. Remarks on Some Nonparametric Estimates of a Density Function // Annals of Mathematical Statistics. — 1956. — 27(3). — С. 832—837.
6. Parzen E. On Estimation of a Probability Density Function and Mode // Annals of Mathematical Statistics. — 1962. — 33(3). — С. 1065—1076.
7. Dempster A. P., Laird N. M., Rubin D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm // Journal of the Royal Statistical Society. — 1977. — 39(1). — С. 1—38.
8. Neal R. M., Hinton G. E. A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants // Learning in Graphical Models. — 1999. — С. 355—368.
9. Bishop C. M. Pattern Recognition and Machine Learning. — Springer, 2006.
10. Rogers S., Girolami M. A First Course in Machine Learning. — Chapman, Hall/CRC, 2016.