Санкт-Петербургский государственный университет

Крушиневский Евгений Александрович

Выпускная квалификационная работа

Интеллектуальный анализ данных и машинное обучение в задачах ускорительной физики

Направление 09.06.01 «Информатика и вычислительная техника»

Образовательная программа МК.3021.2017 «Системный анализ, информатика и управление»

Научный руководитель:
заведущий кафедрой КМиМС,
факультета ПМ-ПУ СПбГУ,
д.ф.-м.н., профессор
Андрианов Сергей Николаевич

Рецензент:
ведущий научный сотрудник,
Объединенный институт ядерных исследований,
кандидат технических наук,
старший научный сотрудник
Юдин Иван Павлович

Санкт-Петербург
2021

Содержание

Введение

На сегодняшний день одной из передовых научных областей является физика элементарных частиц. Данное направление стоит на рубеже научных открытий и имеет огромный потенциал для исследований. В данной области задействованы инновации в физике, информатике, математике, приборостроении. Главными научными целями данной деятельности являются создание и совершенствование накопительных и ускорительных установок, систем для получения транспортировки пучков заряженных частиц, а также прикладные использования, такие как лечение раковых заболеваний [1], разработка новых материалов, поиск темной материи во Вселенной и другие. В физике элементарных частиц исследуются фундаментальные вопросы нашей Вселенной, понимание которых объяснит механизм нашего мира, чтобы мы могли лучше понять, где мы находимся и как это место структурировано. Эти знания могут позволить людям повысить уровень жизни с помощью технологий, которые мы используем ежедневно.

В настоящее время множество организаций и международных коллабораций таких как, Европейская организация по ядерным исследованиям (ЦЕРН) [2] или Объединенный институт ядерных исследований (ОИЯИ) [3], ведут работы над проектами физики элементарных частиц, самым масштабным среди которых является Большой адронный коллайдер (Large Hadron Collider, LHC) [4]. Это огромная арена для изучения различных моделей, стремящихся к лучшему объяснению Вселенной. LHC представляет собой туннель в форме кольца длиной 27 километров и глубиной от 50 до 175 метров, в котором протоны ускоряются до скорости, почти сопоставимой со скоростью света. Каждую секунду в туннеле сталкиваются 40 миллионов сгустков протонов, а энергия столкновения этих протонов достигает 14 тераэлектронвольт. Вокруг кольца расположены четыре больших детектора, которые называются ALICE [5], ATLAS [6], CMS [7] и LHCb [8]. Каждый детектор генерирует огромные объемы данных, сотни петабайт, для последующего анализа данных.

По своей конструкции, LHC — это машина для разрушения протонов и изучения полученных результатов. Сначала создаются протоны из атомов водорода, которые ускоряются до скорости, почти равной скорости света. Затем эти протоны группируются в сгустки, которые сталкиваются между собой. Из энергии этого столкновения появляется множество частиц, которые называются событиями. Затем детектор, который находится вокруг места столкновения, записывает информацию об энергии, которая выделяется каждой частицей, вылетающей из события. После этого начинает действовать алгоритм, который пытается восстановить траектории различных частиц и определить их типы. Изображения отдельных траекторий объединяются в узлы, и формируется структура всего события. Затем строится карта событий, и фильтруются для дальнейшего анализа только те события, которые имеют какое-то значение с физической точки зрения.

На сегодняшний день проведение подобных исследований невозможно без развитой ИТ-инфраструктуры, в том числе специализированного программного обеспечения, которое используется для моделирования столкновений элементарных частиц, трекового анализа и восстановления событий, обработки и визуализации экспериментальных данных и т.д. За последние десятилетия в области физики пучков было разработано много программных пакетов (COSY Infinity [9], MAD [10], Genie [11], PYTHIA [12], GEANT4 [13] и другие) и библиотек [14] (ACCSIM, AIM, ICE, PAC, SIMBAD, SXF, TEAPOT, ZLIB). При проведении вычислительного эксперимента часто требуется провести численное моделирование рассматриваемой задачи различными методами и способами, чтобы определить оптимальное решение, проанализировать систему с разных сторон, оценить производительность и затраты. Актуальной задачей на сегодняшний день является объединение и совместное использование различных программ, унификация форматов входных данных. Про концепцию такого универсального программного комплекса, который предназначен решить подобные проблемы, можно ознакомиться в данных работах [15], [16].

Современные исследования в физике элементарных частиц являются дорогостоящими, ресурсоемкими и высокотехнологичными процессами. Проведение подобных исследований в настоящее время сильно зависит от методов машинного обучения и интеллектуального анализа данных. В данной работе будет продемонстрировано применение таких методов машинного обучения, как градиентный [17] и адаптивный [18] бустинг, в решении задачи классификации частиц и Байесовская оптимизация [19] с Гауссовскими процессами [20] для оптимизации строения детектора.

Постановка задачи

Цель данной работы — продемонстрировать практическое применение методов машинного обучения и интеллектуального анализа данных в решении задач ускорительной физики на примере идентификации частицы по ее откликам в детекторных системах и оптимизации структуры детектора.

Поставленная цель определяет следующие задачи:

1. Сформулировать задачу идентификации частицы как задачу классификации в машинном обучении.
2. Ознакомиться с системами, входящими в структуру детектора.
3. Формализовать задачу оптимизации строения детектора.
4. Выбрать оптимальные методы машинного обучения для решения задачи идентификации частицы.
5. Выбрать оптимальные методы машинного обучения для решения задачи оптимизации структуры детектора.
6. Загрузить и подготовить данные для дальнейшего анализа.
7. Произвести вычислительный эксперимент на подготовленных данных с помощью выбранных методов машинного обучения.
8. Продемонстрировать результаты вычислительных экспериментов и сделать выводы на их основании.

Обзор литературы

По теме использования методов и идей машинного обучения в физике элементарных частиц написано множество статей и научных трудов. Неудивительно, что большинство из них связано с проектами Большого адронного коллайдера (LHC), так как это на сегодняшний день самый крупный генератор данных в данной предметной области. В обзор включены некоторые работы, опубликованные за последние годы, которые являются релевантными задачам данного исследования.

В работах [21], [22] рассмотрены аспекты взаимодействия между современным машинным обучением и физикой элементарных частиц в проектах Большого адронного коллайдера, описаны разные примеры задач распознавания сигнала с использованием обучения модели машинного обучения с учителем.

Возможности глубокого обучения для анализа данных с применением нейронных сетей продемонстрированы в статье [23], где исследуются графовые нейронные сети для работы с детекторами. В статье [24] описано использование машинного обучения для реконструкции и идентификации потоков, создаваемыми в калориметре частицами, образованными в результате столкновений. В работе [25] демонстрируются два набора моделей глубокого обучения для восстановления треков частиц. В первом наборе моделей используются рекуррентные нейронные сети (RNN) [26] для экстраполяции, построения и оценки треков. Второй набор моделей использует графические нейронные сети (GNN) [27] для задач классификации попаданий и классификации частиц.

Помимо нейронных сетей во многих работах используются ансамбли деревьев принятия решений — бустинг. Напрмер, в работе [28] описано применение бустинга для решения экспериментальных задач, связанных с отслеживанием траектории частиц в Большом адронном коллайдере.

Анализ огромного количества данных представляет собой серьезную проблему в современных экспериментах по физике высоких энергий. Для выбора оптимального алгоритма машинного обучения под специфику анализируемых данных исследователю необходимо выбирать и настраивать множество параметров алгоритма, которые называются гиперпараметрами. Выбор гиперпараметров обычно выполняется вручную и часто оказывает значительное влияние на производительность алгоритма машинного обучения. В статье [29] описываются подходы машинного обучения для автоматического выбора оптимальных значений гиперпараметров.

Большое внимание в проведении экспериментов физики элементарных частиц уделяется оптимизации. Проблемы оптимизации встречаются повсеместно и включают определение оптимума целевой функции, оценка параметров которой может быть дорогостоящей в вычислительном отношении. В работе [30] описывается оптимизация параметров моделей машинного обучения для улучшения структур детекторов. В статье [31] предлагается общий подход к оптимизации конструкции детектора, основанный на Байесовской оптимизации и машинном обучении. В работе [32] используется ряд алгоритмов глобальной оптимизации, которые еще не получили широкого распространения в физике частиц.

На основе анализа современных научных работ по применению машинного обучения и интеллектуального анализа данных в физике элементарных частиц сделан вывод, что на сегодняшний день не существует единого универсального метода, который бы обеспечивал самый точный результат для широкого круга задач. К каждой задаче нужен индивидуальный подход. При проведении эксперимента часто требуется провести исследование рассматриваемой задачи различными методами и способами, чтобы определить оптимальное решение, проанализировать систему с разных сторон, оценить производительность и затраты.

Глава 1. Применение машинного обучения для идентификации частиц

В главе приведены основные определения для формализации задачи идентификации частиц, описаны детекторные системы и рассмотрен практический пример решения задачи классификации методами градиентного и адаптивного бустинга.

1.1 Формализация задачи

Рассмотрим частицу D^0, которая распадается на две другие частицы: пион и каон (Рисунок 1):

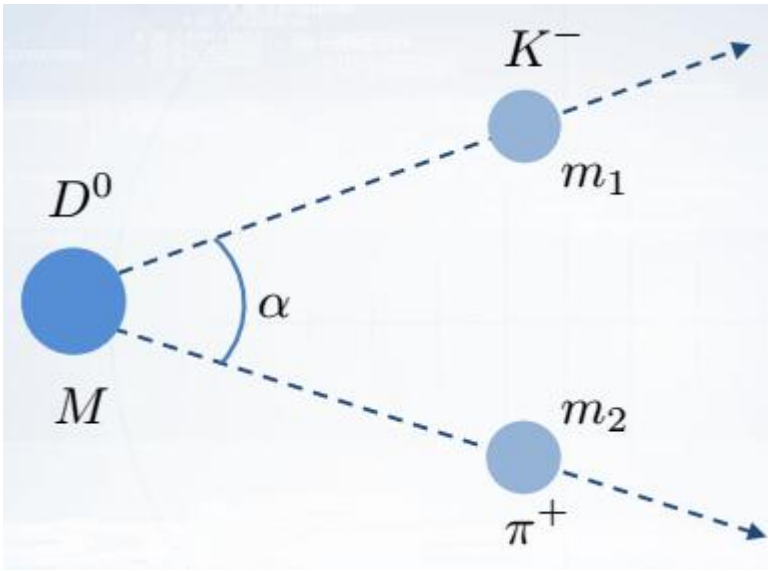

Рисунок 1. Распад частицы

В данном примере D^0 называется материнской частицей, а каон и пион — дочерними частицами. Каждая из этих частиц имеет определенную энергию, импульс и массу. Энергия и импульс частицы определяются скоростью этой частицы. Масса частицы определяется ее типом. В данном случае M, m_1, m_2 являются массами частицы D^0, пиона и каона соответственно.

По закону сохранения энергии:

$$E_M = E_1 + E_2 \quad (1)$$

где E_M — энергия частицы D^0, E_1 — энергия каона, E_2 — энергия пиона.

По закону сохранения импульса:

$$\overrightarrow{p_M} = \overrightarrow{p_1} + \overrightarrow{p_2} \quad (2)$$

где p_M — импульс частицы D^0, p_1 — импульс каона, p_2 — импульс пиона. Энергия, импульс и масса частицы связаны уравнением:

$$E^2 = p^2c^2 + m^2c^4 \quad (3)$$

Уравнения (1) — (3) позволяют оценить массу материнской частицы D^0, используя массы, энергию и значения импульса дочерних частиц:

$$M^2 = m_1^2 + m_2^2 + \frac{2}{c^4}(E_1E_2 - p_1p_2c^2\cos\alpha) \quad (4)$$

Уравнение (4) требует наличия информации об угле между траекториями каона и пиона. Траектории дочерних частиц нужны не только для измерения угла между ними, но и для проверки того, что дочерние частицы происходят из одной точки, от одной и той же частицы. Это становится более важным, когда у нас есть несколько распадов одновременно.

Чтобы оценивать траектории частиц, импульс и энергию, а также распознавать типы дочерних частиц при распаде и восстанавливать параметры материнских частиц, в физике высоких энергий используются различные системы, такие как электромагнитные и адронные калориметры [33], мюонные камеры [34], RICH-детекторы (Ring Image Cherenkov detector) [35] и другие.

Система слежения. Система слежения [36] отвечает за распознавание траекторий частиц и оценку их параметров. Она является первой системой в структуре детектора и устанавливается перед зоной столкновения частиц с детектором:

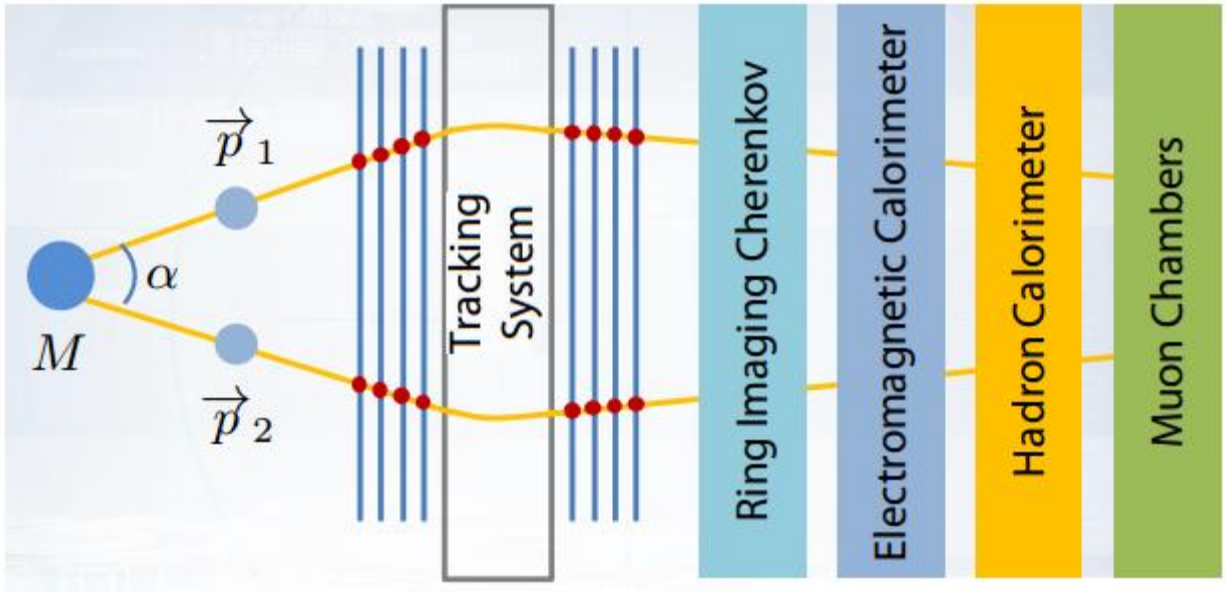

Рисунок 2. Схема детектора частиц

Система слежения состоит из магнита, который отклоняет частицы и позволяет измерять их импульс, и нескольких слоев датчиков. Отклики датчиков при полете частицы через эти слои называется попаданием. Попадания позволяют распознавать треки частиц и оценивать их параметры. Задача распознавания попаданий, принадлежащих одному и тому же треку, называется распознаванием образов трека.

В физике высоких энергий существует множество методов распознавания образов треков. Наиболее популярны методы, основанные на преобразовании Хафа [37] и фильтре Калмана [38]. Также в 1980-х годах был разработан ряд методов распознавания образов треков, основанных на нейронных сетях Хопфилда [39]. Они более сложны, чем методы, основанные на фильтрах Калмана, и требуют больших вычислительных ресурсов. Но с другой стороны, они обеспечивают лучшее качество распознавания, особенно в экспериментах с высокой плотностью треков. Более того, существует множество современных методов машинного обучения для кластеризации и классификации, включая сверточные нейронные сети [40] и рекуррентные нейронные сети, которые также успешно применяются в данной задаче. Например, сверточные нейронные сети используются для классификации распознанных комбинаций совпадений на правильные и неправильные,

рекуррентные нейронные сети учитывают последовательности попаданий в слоях системы слежения, чтобы предсказать следующее попадание в трек.

Помимо распознавания треков частиц, система слежения также измеряет импульс частицы. Для этого в системе слежения есть магнит, магнитное поле которого отклоняет заряженные частицы. В однородном магнитном поле частица летит по окружности, радиус R которой пропорционален магнитному полю B и импульсу частицы p, как показано на Рисунке 3:

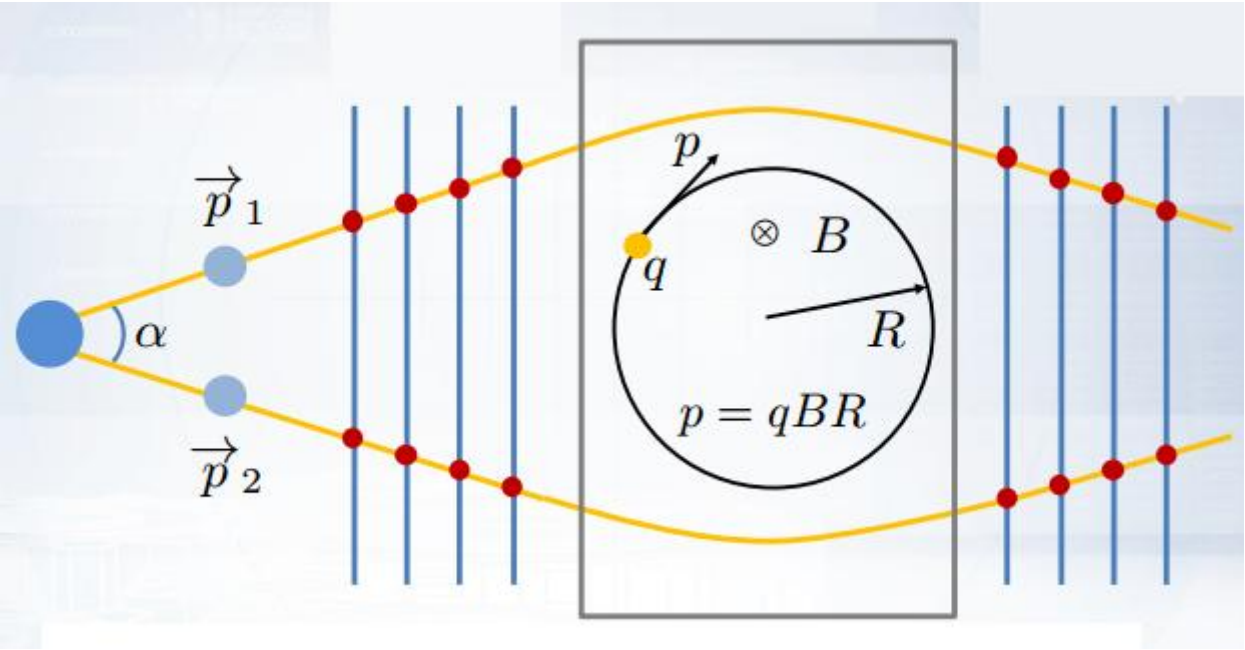

Рисунок 3. Структура системы слежения

Чтобы оценить импульс частицы, нужно измерить радиус данной окружности. Для этого после магнита установлены несколько дополнительных сенсорных слоев, которые оценивают параметры трека после прохождения магнита. Зная угол наклона треков частицы до и после прохождения магнита, можно вычислить отклонение частицы в магнитном поле и оценить ее импульс.

RICH-детектор. RICH-детектор (Ring Image CHerenkov) — детектор, который основан на эффекте излучения Вавилова-Черенкова [41]. Этот эффект описывает поведение частицы в материале, когда она летит со

скоростью, превышающей скорость света в этом материале. Когда частица летит с такой скоростью, она испускает фотоны в конусе (Рисунок 4):

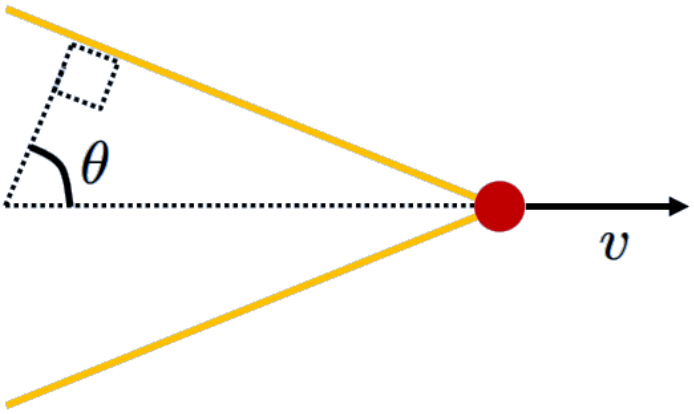

Рисунок 4. Излучение Вавилова-Черенкова

Показатель преломления материала n — это отношение скорости света в вакууме c к скорости света в этом материале v:

$$n = \frac{c}{v} \tag{5}$$

Чтобы излучать свет, частицам требуется показатель преломления (5), близкий к единице. Такие значения имеют газы. Более того, газы практически не влияют на полет частиц, из-за чего используются в RICH-детекторах.

Частица испускает фотоны под углом θ:

$$cos\theta = \frac{1}{n\beta}, \qquad \beta = \frac{v}{c} \tag{6}$$

Так как импульс частицы определяется ее массой и скоростью, то справедливы следующие выражения:

$$p = \frac{mc\beta}{\sqrt{1-\beta^2}}, \qquad \beta = \frac{p}{\sqrt{p^2+m^2c^2}} \tag{7}$$

Согласно выражениям (7), из выражения (6) следует уравнение для вычисления косинуса угла излучения Вавилова-Черенкова:

$$cos\theta = \frac{\sqrt{p^2+m^2c^2}}{np} \tag{8}$$

Уравнение (8) показывает, что разные типы частиц с разными массами имеют разные зависимости угла вылета от импульса частицы.

Таким образом, RICH-детектор по треку и импульсу частицы идентифицирует ее тип: мюон, каон, пион, протон или электрон.

Калориметры. Калориметры задерживают все частицы, кроме мюонов. Электромагнитный калориметр останавливает электроны и фотоны. Все остальные частицы летят дальше к адронному калориметру. Адронные калориметры останавливают протоны, нейтроны и другие частицы, содержащие кварки. Во всех экспериментах по физике высоких энергий адронные калориметры стоят после электромагнитных (Рисунок 2).

Калориметры измеряют энергию частиц. Частицы взаимодействуют с большим количеством вещества калориметра и теряют свою энергию. Калориметр измеряет, сколько энергии теряют частицы, прежде чем они остановятся.

Электромагнитный калориметр отвечает за измерение энергии электронов и фотонов. В экспериментах физики высоких энергий электромагнитные калориметры состоят из множества прозрачных кристаллов, которые создают электромагнитный поток частиц. Одна частица может активировать несколько кристаллов калориметра, поэтому для того, чтобы более точно оценить энергию частицы, нужно определить кристаллический кластер, соответствующий данной частице. Для этого трек частицы, полученный из системы слежения, экстраполируется на электромагнитный калориметр. Затем кристаллы, близкие к треку в калориметре, группируются в кластер. Сумма энергий, измеренная во всех кристаллах кластера, соответствует энергии частицы. Фотоны также создают кластеры в калориметре, но у них нет попаданий в систему слежения, и их треки не распознаются. Методы машинного обучения позволяют эффективно формировать кластеры кристаллов, распознавать кластеры фотонов и отделять их от кластеров электронов.

Адронный калориметр отвечает за измерение энергии протонов, нейтронов и других частиц, содержащих кварки. Принцип работы адронного калориметра аналогичен принципу работы электромагнитного калориметра,

только вместо электромагнитного потока частиц формируется адронный. Адронный поток частиц формируется после прохождения частиц через специальные слои тяжелой материи, между которыми установлены счетчики для подсчета количества частиц в потоке.

Мюонная система. Мюонная система отвечает за идентификацию мюонов и оценку их энергии. Мюоны слабо взаимодействуют с веществом и почти не имеют откликов в калориметрах, из-за чего мюонные системы устанавливаются в конце детектора (Рисунок 2).

Мюонная система состоит из нескольких последовательных слоев мюонных камер и металла. Цель камер - обнаруживать мюоны, а металл пытается остановить мюоны. Мюонная камера - это трубка, наполненная газом и имеющая внутри проволоку. Когда мюон проходит через камеру, он ионизирует атомы газа. В процессе ионизации в камере образуются ионы и электроны. Из-за электромагнитного поля внутри камеры ионы уходят на катод, а электроны - на анод. Это производит электрический ток и создает сигнал в камере, которая обнаруживает мюон. Похожий принцип используется и в системах слежения.

В мюонной системе камера активируется, когда через нее пролетает мюон. Трек частицы из системы слежения экстраполируется на мюонную систему, и если трек проходит через активные мюонные камеры, то частица считается мюоном. Для распознавания активных камер для трека эффективно применяются методы машинного обучения.

1.2 Задача классификации

Существует пять типов частиц: электрон, протон, каон, пион и мюон. Существует пять различных детекторных систем: системы слежения, RICH-детекторы, электромагнитные и адронные калориметры, мюонные камеры.

Целью идентификации частицы является определение типа частицы по информации, полученной из данных откликов от различных детекторных систем.

Задачу идентификации частиц можно рассматривать как задачу множественной классификации в машинном обучении (Рисунок 5).

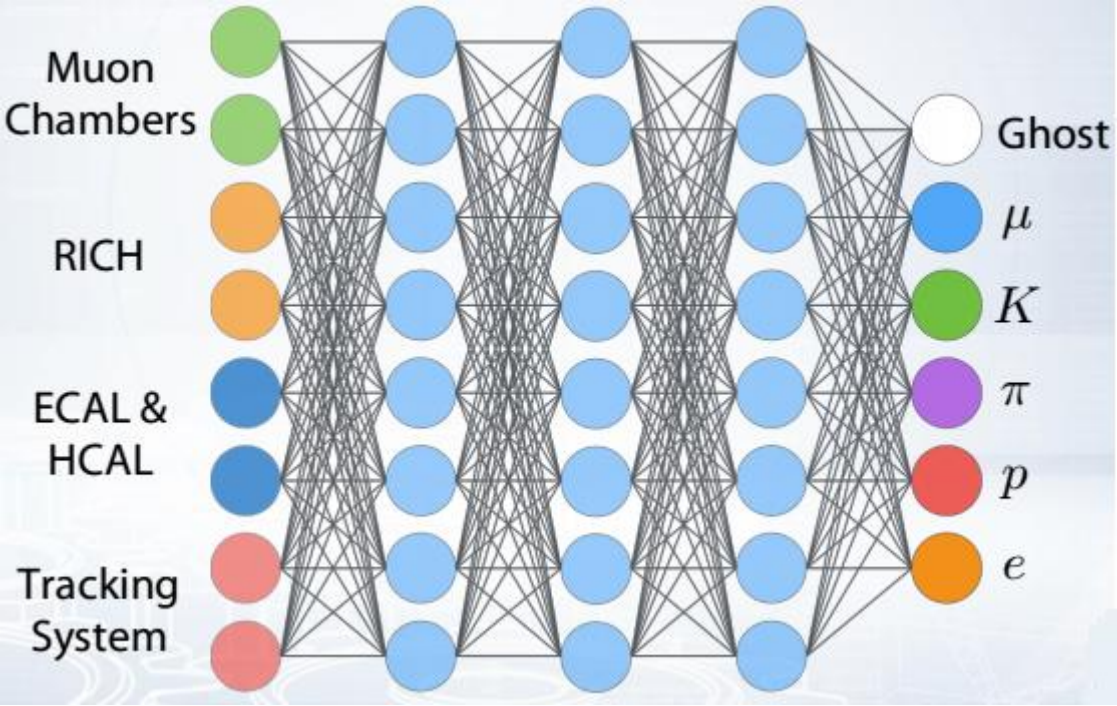

Рисунок 5. Задача классификации частиц

Отклики треков частиц в детекторных системах используются в качестве входных данных классификатора. Например, системы слежения предоставляют информацию о параметрах трека частицы, качестве трека, импульсе и заряде частицы, а также координатах вершины распада. RICH-детектор отслеживает угол испускания частицы, калориметры измеряют энергию частицы и количество откликов для этих частиц, мюонная система сообщает, есть ли у трека попадания внутри системы, и сколько активных камер соответствует треку.

В качестве выходной информации классификатор для каждой частицы присваивает один из шести ярлыков. Пять из них соответствуют пяти различным типам частиц: мюон, каон, пион, протон и электрон. Последний ярлык соответствует шуму или всем остальным типам частиц, которые называются призраками. Призрак - это трек, который был ошибочно распознан методом распознавания образов трека и не соответствует ни одной реальной частице в детекторе. Таким образом, для каждого трека,

распознаваемого детектором, классификатор дает вероятность принадлежности к каждому из типов частиц.

При выборе метода машинного обучения для решения задач классификации необходимо учитывать тип анализируемых данных. Например, нейронные сети эффективно решают данную задачу на однородных данных, таких как изображения или звук. В случае, когда на вход классификатору предоставляются данные различного характера и разной структуры, то оптимальным инструментом решения задачи классификации является бустинг (англ. boosting — усиление). Суть бустинга заключается в преобразовании слабых (с меньшим показателем точности) обучающих алгоритмов в сильные (с высоким показателем точности).

В виду неоднородности исходных данных в задаче классификации частиц, а также для обеспечения равномерной зависимости эффективности сигнала от параметров частицы, в практической части данной работы использовались алгоритмы адаптивного и градиентного бустинга.

1.3 Практический пример

Все вычисления и визуализация данного примера реализованы на языке программирования Python [42] с использованием библиотек numpy [43], pandas [44], Scikit-Learn [45].

Есть шесть типов частиц:

- Электрон
- Протон
- Мюон
- Каон
- Пион
- «Призрак»

«Призрак» — это все остальные частицы, которые не относятся к предыдущим пяти типам. Есть пять типов систем (детекторов):

- Системы слежения
- RICH-детекторы
- Электромагнитные калориметры
- Адронные калориметры
- Мюонные камеры

Различные типы частиц имеют разные отклики в детекторах. Цель задачи — обучить классификатор распознавать тип частицы по ее откликам в детекторах.

Подготовка входных данных. Исходные данные для данного примера находятся в открытом доступе по ссылке [46]. Характеристики используемого набора данных:

- 1 200 000 записей
- Данные представлены в формате CSV
- Данные содержат информацию об откликах частиц в детекторах (системы слежения, RICH-детекторы, электромагнитные и адронные калориметры, мюонные камеры)

Для обучения классификатора исходный набор данных был разделен на две части — обучающую и тестовую в соотношении 9:1 (обучающая выборка — 1080000 записей, тестовая — 120000 записей).

Логистическая функция ошибки. Для того, чтобы понять, насколько эффективно и точно модель машинного обучения справляется с поставленной задачей, необходимо определенным образом оценить ее полученный результат, рассчитать ошибку ее предсказаний. В большинстве случаев ошибка рассчитывается как разница между фактическим выходным значением и прогнозируемым выходным значением. Функция, используемая для вычисления этой ошибки, называется функцией ошибки или функцией затрат. Одной из часто используемых функций ошибок является логарифмическая функция ошибки (log loss или кросс-энтропия) [47].

Логарифмическая функция ошибки измеряет расхождение между двумя вероятностными распределениями. Если кросс-энтропия велика, это означает, что разница между двумя распределениями велика, а если кросс-энтропия мала, то распределения похожи друг на друга.

Логарифмическая функция ошибки определяется следующим образом:

$$H(P, Q) = -\sum_{x} P(x) \log Q(x) \tag{9}$$

где P – распределение истинных ответов, а Q – распределение вероятностей прогнозов модели.

Пусть N — это количество наблюдений, M — количество возможных меток класса, y — двоичный индикатор (0 или 1) того, является ли данная метка класса C правильной классификацией для текущего наблюдения O, p — прогнозируемая вероятность модели. Тогда в случае бинарной классификации (M=2) формула (9) принимает вид:

$$L = -(y\, log(p) + (1 - y)\, log(1 - p)) \tag{10}$$

В случае мультиклассовой классификации (M>2) берется сумма значений логарифмических функций ошибок для каждого прогноза наблюдаемых классов:

$$L = -\sum_{C=1}^{M} y_{O,C} \log(p_{O,C}) \quad (11)$$

Кросс-энтропия для бинарной или мульти-классовой задачи классификации рассчитывается как средняя кросс-энтропия среди всех примеров. Логарифмическая функция ошибок использует отрицательные значения логарифма, чтобы обеспечить удобную метрику для сравнения. Этот подход обусловлен тем, что логарифм чисел, меньших единицы, возвращает отрицательные значения, что затрудняет работу при сравнении производительности двух моделей.

В данной работе используется реализация кросс-энтропии в библиотеке sklearn — log_loss [48].

Адаптивный бустинг. В 1997 году Йоав Фройнд и Роберт Шапире предложили алгоритм, который может использоваться в сочетании с несколькими алгоритмами классификации для улучшения их эффективности — AdaBoost. Основная идея данного алгоритма — минимизация ошибки предыдущих моделей, концентрация внимания на недообученности алгоритма. После предсказаний каждой модели увеличиваются веса ошибочно классифицированных объектов, чтобы следующая модель лучше отработала на них. Адаптивность алгоритма заключается в том, что каждый следующий классификатор строится по неверно классифицированным предыдущим классификатором данным, таким образом исправляя эти ошибки. В результате из совокупности слабых по отдельности классификаторов получается классификатор, который обеспечивает высокую итоговую точность.

Достоинства алгоритма AdaBoost:

- Высокая обобщающая способность. В реальных задачах удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться по мере увеличения числа базовых алгоритмов.

- Простота реализации.
- Собственные накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов.

Недостатки алгоритма AdaBoost:

- Нельзя распараллелить, поскольку каждый из предикторов может быть обучен лишь только после окончания обучения предыдущего.
- Склонен к переобучению при наличии значительного уровня шума в данных. Функция потерь слишком сильно увеличивает веса наиболее трудных объектов, на которых ошибаются многие базовые алгоритмы. Эти объекты чаще всего оказываются шумовыми выбросами. В результате алгоритм начинает настраиваться на шум, что ведёт к переобучению.

Основные шаги алгоритма AdaBoost:

1. Изначально всем точкам присвоены равные веса.
2. Модель строится на подвыборке данных.
3. По этой модели получаются предсказания для всех данных.
4. По предсказаниям и истинным значениям вычисляются ошибки.
5. В построении следующей модели наибольшие веса присваиваются точкам данных, на предсказании которых алгоритм ошибся.
6. Веса могут быть определены по величине ошибки. А именно, чем больше ошибка, тем больше вес.
7. Этот процесс повторяется, пока функция ошибки не перестанет меняться или пока не будет достигнуто максимальное число предикторов.

В данном примере используется реализация алгоритма AdaBoost в библиотеке sklearn — AdaBoostClassifier [49].

Основные параметры метода AdaBoostClassifier описаны в Таблице 1.

Наименование параметра	Описание параметра
base_estimator	Определяет базовый алгоритм
n_estimator	Определяет количество базовых алгоритмов
learning_rate	Коэффициент скорости обучения - параметр, отвечающий за то, насколько изменяются веса
random_state	Делает ответ модели повторимым. Модель всегда будет давать один и тот же ответ на одних и тех же данных и параметрах при совпадении значения этого параметра

Таблица 1. Параметры метода AdaBoostClassifier

Основные этапы работы:

1. Обучение классификатора AdaBoostClassifier классифицировать частицы на обучающей выборке данных
2. Расчет вероятностей принадлежности частицы к каждому из классов
3. Расчет ошибки классификации логистической функцией ошибки log_loss (9). Полученный результат: 0.53283105461409257

Градиентный бустинг. Другой очень популярный бустинговый алгоритм, принцип работы которого очень похож на рассмотренный только что AdaBoost. Градиентный Бустинг работает последовательно добавляя к прошлым моделям новые так, чтобы исправлялись ошибки, допущенные предыдущими предикторами.

Градиентный Бустинг отличается от Адаптивного тем, что, в отличие от AdaBoost, изменяющего веса при каждой итерации, Градиентный пытается обучать новые модели по остаточной ошибке прошлых (двигаясь к минимуму функции потерь).

Достоинства градиентного бустинга:

- Более устойчив к выбросам (зашумленности данных), чем AdaBoost
- Более гибкий, чем AdaBoost
- Эффективен в работе с разреженными данными

Недостатки градиентного бустинга:

- Склонен к переобучению, по причине нацеленности на устранение всех ошибок
- Вычислительно дорогостоящий (время и память)

Основные шаги градиентного бустинга:

1. Модель строится по подборке данных.
2. Эта модель делает предсказания для всего набора данных.
3. По предсказаниям и истинным значениям вычисляются ошибки.
4. Новая модель строится с учетом ошибок как целевых переменных. При этом мы стремимся найти лучшее разделение для минимизации ошибки.
5. Предсказания, сделанные с помощью этой новой модели, сочетаются с предсказаниями предыдущих.
6. Снова вычисляются ошибки с использованием этих предсказанных значений и истинных значений.
7. Этот процесс повторяется, пока функция ошибки не перестанет меняться или пока не будет достигнуто максимальное число предикторов.

В данном примере используется реализация алгоритма AdaBoost в библиотеке sklearn — GradientBoostingClassifier [50].

Основные параметры метода GradientBoostingClassifier описаны в Таблице 2.

Наименование параметра	Описание параметра
min_samples_split	Минимальное число точек, необходимое для разделения. Предназначено для предотвращения переобучения
min_samples_leaf	Минимальное количество элементов в листе или узле дерева
max_depth	Максимальная глубина дерева. Предназначено для предотвращения переобучения
max_features	Количество признаков, учитываемых алгоритмом
n_estimator	Определяет количество базовых алгоритмов
learning_rate	Коэффициент скорости обучения - параметр, отвечающий за то, насколько изменяются веса
random_state	Делает ответ модели повторимым. Модель всегда будет давать один и тот же ответ на одних и тех же данных и параметрах при совпадении значения этого параметра

Таблица 2. Параметры метода GradientBoostingClassifier

Основные этапы работы:

1. Обучение классификатора GradientBoostingClassifier классифицировать частицы на обучающей выборке данных
2. Расчет вероятностей принадлежности частицы к каждому из классов
3. Расчет ошибки классификации функцией log_loss. Полученный результат: 0.55793306232418781

Значение функции log_loss для результатов классификатора GradientBoostingClassifier получилось больше, чем для классификатора AdaBoostClassifier, следовательно адаптивный бустинг более точно классифицировал частицы. На Рисунке 6 изображены ROC-кривые [51] для каждого класса частиц, классифицированных методом адаптивного бустинга:

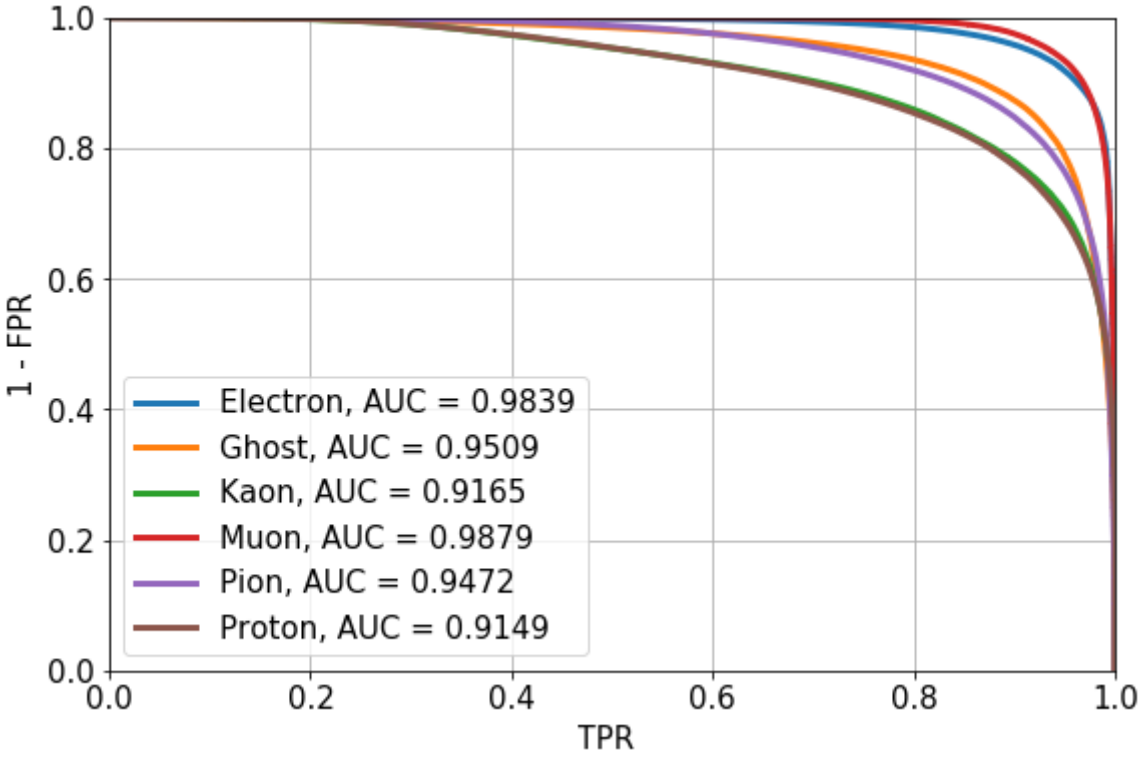

Рисунок 6. ROC-кривые для каждого класса частиц, классифицированных методом адаптивного бустинга

В машинном обучении ROC- кривая строится как зависимость частоты истинных положительных результатов (*True Positive Rate, TPR*) от частоты ложных срабатываний (*False Positive Rate, FPR*). Однако, в физике высоких энергий ROC-кривая строится как зависимость частоты ложных

срабатываний, вычтенной из единицы (1-FPR, отклонение фона), от частоты истинных положительных результатов (TPR, эффективность сигнала).

Заключение к Главе 1

В данной главе рассмотрены детекторные системы, сформулирована задача идентификации частицы как задача классификации в машинном обучении. Описано решение задачи классификации градиентным и адаптивным бустингом.

Глава 2. Оптимизация строения детекторов в экспериментах физики высоких энергий

В данной главе описывается задача оптимизации строения детектора, содержатся необходимые определения и понятия для Байесовской оптимизации с Гауссовскими процессами и рассмотрен практический пример оптимизации с применением нескольких методов машинного обучения.

2.1 Постановка задачи

Детекторы являются очень сложными в строении и дорогостоящими конструкциями в экспериментах физики высоких энергий. Они состоят из разных частей, таких как систем слежения, RICH-детекторов (Ring Image Cherenkov detector), электромагнитного и адронного калориметров, мюонной системы и других. Каждая из этих частей имеет тысячи датчиков, которые регистрируют частицы и измеряют их импульс и энергию. Расположение этих датчиков влияет на точность, эффективность и стоимость этих детекторов.

Цель оптимизации детектора - найти оптимальное расположение датчиков в детекторе. Основные шаги оптимизации следующие:

1. Должна быть определена целевая функция оптимизации. Эта функция объединяет ключевые значения, которые необходимо оптимизировать: стоимость детектора, эффективность, точность восстановления импульса и так далее.
2. Необходимо выбрать набор параметров, которые определяют расположение датчика и влияют на значения целевой функции.
3. Найти значения параметров, соответствующих оптимуму целевой функции.

Один из самых популярных и простых методов оптимизации конструкции детектора - поиск по сетке (англ. *Grid search*) [52]. Он определяет сетку значений параметров и вычисляет значения целевой функции для каждого узла в сетке. Узел с наибольшим или наименьшим

значением функции считается оптимальным. Однако у этого метода есть ряд недостатков. Первый из них заключается в том, что поиск по сетке целесообразно использовать, когда количество параметров невелико, а размер сетки экспоненциально растет с увеличением количества оптимизируемых параметров. Таким образом, поиск по сетке требует больших вычислительных ресурсов, и более рационально применять метод байесовской оптимизации с использованием гауссовских процессов.

Байесовская оптимизация - это метод поиска оптимума целевой функции, цель которого заключается в том, чтобы использовать как можно меньшее количество вычислений данной функции. Байесовская оптимизация показывает лучшие результаты с меньшими вычислениями по сравнению с поиском по решётке и случайным поиском благодаря возможности оценить качество экспериментов ещё до их выполнения.

Для Байесовской оптимизации следует упомянуть некоторые свойства нормального распределения.

2.2 Нормальное распределение

Нормальное или гауссово распределение — это непрерывное распределение вероятностей. Плотность вероятности в одномерном случае определяется уравнением:

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{12}$$

В уравнении (12) x — случайная величина, $\mu = E[x]$ — математическое ожидание, $\sigma^2 = E[(x-\mu)^2]$ — дисперсия, σ — среднеквадратическое отклонение.

Многомерное нормальное распределение (Гауссовское распределение) является обобщением одномерного нормального распределения на более высокие измерения. Плотность вероятности в N измерениях определяется следующим уравнением:

$$\mathcal{N}(\mu,\Sigma)=\frac{1}{\sqrt{(2\pi)^N|\Sigma|}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \quad (13)$$

В уравнении (13) $x = (x_1, x_2, \dots, x_N)^T$ — N-мерный произвольный вектор случайных величин, $\mu=(\mu_1,\mu_2,\dots,\mu_N)^T=(E[x_1],E[x_2],\dots,E[x_3])^T$ — N-мерный вектор математических ожиданий для каждой случайной величины, $|\Sigma|$ — определитель ковариационной матрицы Σ размером $N\times N$:

$$\Sigma=\begin{pmatrix} Cov(x_1,x_1) & \cdots & Cov(x_1,x_N)\\ \vdots & \ddots & \vdots \\ Cov(x_N,x_1) & \cdots & Cov(x_N,x_N)\end{pmatrix},\ Cov(x_1,x_1)=E[(x_i-\mu_i)(x_j-\mu_j)]$$

Вектор μ из уравнения (9) определяет положение центра распределения. Матрица **Σ** устанавливает ширину, форму и ориентацию распределения.

Пусть вектор x состоит из двух векторов, x_a *и* x_b, размером $k\times 1$ и $(N-k)\times 1$ соответственно, а вектор μ и матрица Σ имеют следующие структуры:

$$x=\begin{pmatrix}x_a\\x_b\end{pmatrix},\qquad \mu=\begin{pmatrix}\mu_a\\\mu_b\end{pmatrix},\qquad \Sigma=\begin{pmatrix}\Sigma_{aa} & \Sigma_{ab}\\ \Sigma_{ba} & \Sigma_{bb}\end{pmatrix}$$

Тогда условное распределение [53]:

$$\mathcal{P}(x_a|x_b)=\mathcal{N}\left(\mu_{a|b},\Sigma_{a|b}\right) \quad (14)$$

где $\mu_{a|b}=\mu_a+\Sigma_{ab}\Sigma_{bb}^{-1}(x_b-\mu_b)$ и $\Sigma_{a|b}=\Sigma_{aa}-\Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$

и маргинальное распределение [54]:

$$\mathcal{P}(x_a)=\int\mathcal{P}(x_a,x_b)dx_b=\mathcal{N}(\mu_a,\Sigma_{aa}) \quad (15)$$

будут нормальными.

Данные свойства нормального распределения используются в гауссовских процессах регрессии.

2.3 Гауссовские процессы в задаче регрессии

Гауссовский процесс можно рассматривать как способ решения задачи регрессии в машинном обучении, в котором используется мера подобия

между точками для получения прогноза значения невидимой точки из обучающей выборки.

Представим задачу регрессии в следующем виде:

$$y = f + \varepsilon \tag{16}$$

где $y = (y_1, y_2, \dots, y_N)^T$ — вектор целевых наблюдений, $f = (f(x_1), f(x_2), \dots, f(x_N))^T$ — вектор истинных значений функции f, $\boldsymbol{\varepsilon}$ — шум.

Пусть $\boldsymbol{\varepsilon}$ имеет нормальное распределение. Вектор $\boldsymbol{y}$ распределяется по многомерному нормальному распределению (9) $\mathcal{N}(f, \Sigma)$, где математическим ожиданием является вектор f, а матрицей ковариации — единичная матрица, умноженная на параметр α:

$$\Sigma = \alpha \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

Пусть вектор f также имеет нормальное распределение (13) $\mathcal{N}(0, K)$ с нулевым математическим ожиданием и ковариационной матрицей K:

$$K = \begin{pmatrix} Cov\big(f(x_1), f(x_1)\big) & \cdots & Cov\big(f(x_1), f(x_N)\big) \\ \vdots & \ddots & \vdots \\ Cov\big(f(x_N), f(x_1)\big) & \cdots & Cov\big(f(x_N), f(x_N)\big) \end{pmatrix}$$

где $Cov\left(f(x_i), f\left(x_j\right)\right) = k\left(x_i, x_j\right) = \sigma^2 e^{-d^2(x_i, x_j)}$, σ — константа, d — евклидово расстояние.

Матрица K построена таким образом, что чем больше схожи точки x_i и x_j, тем больше корреляция между значениями $f(x_i)$ и $f\left(x_j\right)$.

Распределения векторов y и f позволяют рассчитать распределение наблюдений (13) $\mathcal{N}(0, C)$ с нулевым математическим ожиданием и следующей ковариационной матрицей C:

$$C = K + \alpha \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

Распределение $\mathcal{N}(0, C)$ можно использовать для оценки значения y_{N+1} для заданной точки x_{N+1}. Вектор целевых наблюдений с $N+1$ точкой $y_{N+1\times 1} = (y_1, y_2, \dots, y_N, y_{N+1})^T$ имеет нормальное распределение (13) $\mathcal{N}(0, C_{N+1})$, где ковариационная матрица C_{N+1} имеет следующую структуру:

$$C_{N+1} = \begin{pmatrix} C_N & k \\ k^T & c \end{pmatrix}$$

Согласно формуле (14) условное распределение значения наблюдения y_{N+1} с учетом N предыдущих точек имеет нормальное распределение:

$$p(y_{N+1}|y_{N\times 1}) = \mathcal{N}\left(\mu_{GP}(x_{N+1}), \sigma_{GP}(x_{N+1})\right) \quad (17)$$

В формуле (17) $\mu_{GP}(x_{N+1}) = k^T C_N^{-1} y_{N\times 1}$ — математическое ожидание, $\sigma_{GP}(x_{N+1}) = c - k^T C_N^{-1} k$ — стандартное отклонение.

Гауссовские процессы регрессии эффективно применяются в байесовской оптимизации.

2.4 Байесовская оптимизация

Байесовская оптимизация - это метод нахождения оптимума целевой функции в условиях, когда вычисление целевой функции в одной точке дорогостояще, и производные целевой функции неизвестны. Цель байесовской оптимизации - найти оптимум целевой функции, используя наименьшее количество вычислений данной функции.

Пусть $f(x)$ — целевая функция, для которой необходимо найти оптимум. Алгоритм байесовской оптимизации состоит из следующих шагов:

1. Найти приближение целевой функции с использованием ранее вычисленных значений $\{x_i, y_i\}_{i=1}^N$, решая задачу регрессии.
2. Используя полученное на предыдущем шаге приближение, найти оптимальную точку функции выбора данных $u_N(x)$: $x_{N+1} = argmax_x u_N(x)$
3. Посчитать целевую функцию в следующей точке $y_{N+1} = f(x_{N+1}) + \varepsilon_{N+1}$
4. Повторить шаги 1-3

Функция выбора данных используется в байесовской оптимизации для оценки следующей точки вычисления целевой функции. Есть множество функций выбора. Например, LCB (Lower Confidence Bound) и UCB (Upper Confidence Bound) [55]:

$$LCB(x) = \mu(x) - k\sigma(x) \tag{18}$$

$$UCB(x) = \mu(x) + k\sigma(x) \tag{19}$$

где $\mu(x)$ — математическое ожидание аппроксимации целевой функции, $\sigma(x)$ — стандартное отклонение, k — регулируемый параметр.

Функция LCB (18) служит для минимизации целевой функции, UCB (19) — для максимизации.

Байесовская оптимизация с гауссовскими процессами позволяет балансировать между эксплорацией и эксплуатацией целевой функции.

В случае эксплуатации каждой итерации оптимизации выбирается точка с высоким или низким математическим ожиданием. Ключевые особенности эксплуатации следующие:

1. На каждой итерации выбирается новая точка, близкая к найденному оптимуму целевой функции.
2. Нет никакой гарантии, что текущий оптимум будет глобальным.
3. Другие области целевой функции не исследуются.
4. Эксплуатация требует меньше итераций для поиска оптимума, чем эксплорация.

Характерными свойствами эксплорации являются:

1. Выбор точки с высокой дисперсией на каждой итерации оптимизации.
2. Исследуются все области целевой функции.
3. Высокая вероятность, что найденный оптимум является глобальным.
4. Для поиска оптимума целевой функции требует больше итераций, чем эксплуатация.

Компромисс между эксплорацией и эксплуатацией достигается подбором регулируемых параметров функций выбора данных.

2.5 Практический пример

Все вычисления и визуализация данного примера реализованы на языке программирования Python с использованием библиотек numpy, pandas, skopt [56].

В качестве примера рассмотрим следующую детекторную систему :

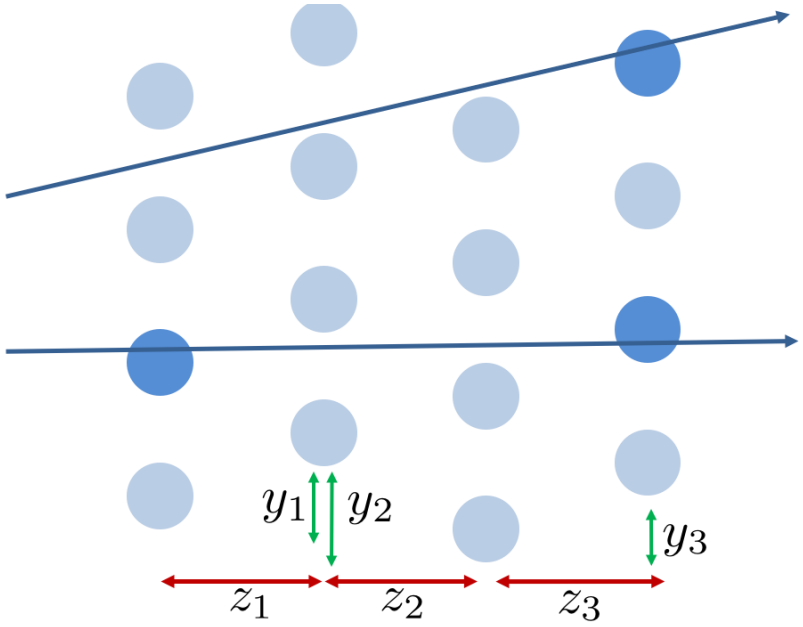

Рисунок 7. Схема расположения трубок в детекторной системе

Система (Рисунок 7) состоит из четырех слоев трубок, которые фиксируют пролетающие через них частицы. В каждом слое по 200 трубок. Система параметризуется шестью регулируемыми параметрами: y_1, y_2, y_3, z_1, z_2, z_3 . Эти параметры описывают сдвиги между слоями: y_1, y_2, y_3 — по вертикали, z_1, z_2, z_3 — по горизонтали.

Пусть радиус трубок равен 1 см, расстояние между трубками в слое — 4 см. Значения параметров z_1, z_2, z_3 должны быть больше 2 см, иначе трубки будут пересекаться.

Пусть начальные значения сдвигов между слоями будут следующими: $y_1 = 0$, $y_2 = 0$, $y_3 = 0$, $z_1 = 2$, $z_2 = 4$, $z_3 = 6$.

На основе указанных начальных данных были сгенерированы координаты расположения трубок в слоях. Фрагмент полученной структуры расположения трубок выглядит следующим образом:

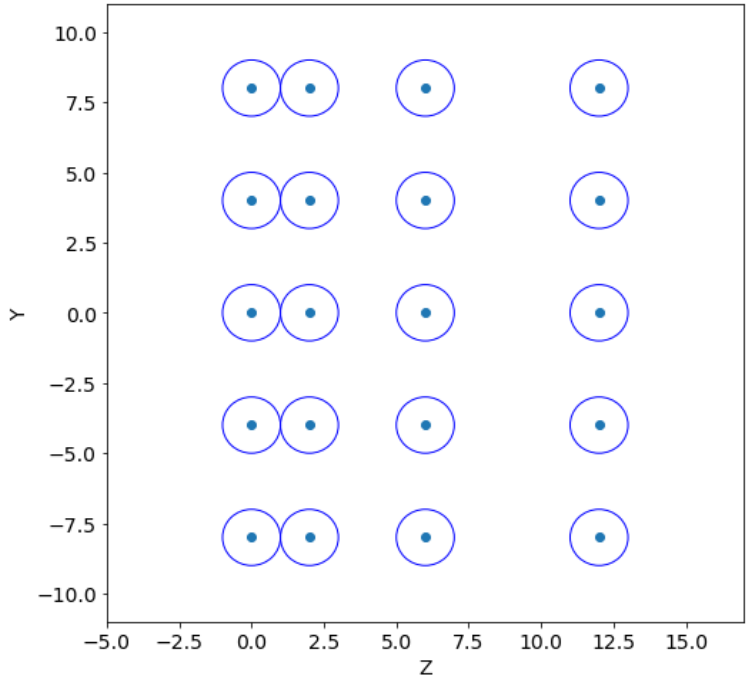

Рисунок 8. Расположение трубок в системе по начальным значениям параметров

Рассмотрим прямолинейные траектории полета частиц, которые описываются следующим уравнением:

$$y = kz + b \tag{20}$$

где y, z — координаты траектории, k — наклон траектории, b — значение пересечения линией траектории оси Y.

Параметры траектории генерируются из следующих распределений:

$$b \in U(b_{min}, b_{max}), k = tan(\alpha), \alpha \in N(\mu_\alpha, \sigma_\alpha) \tag{21}$$

где U — равномерное распределение, N — нормальное распределение, α — угол наклона траектории.

Пусть $b_{min} = -100$, $b_{max} = 100$, $\mu_\alpha = 0$, $\sigma_\alpha = 0{,}2$. Сгенерируем 1000 траекторий с такими параметрами распределений (21):

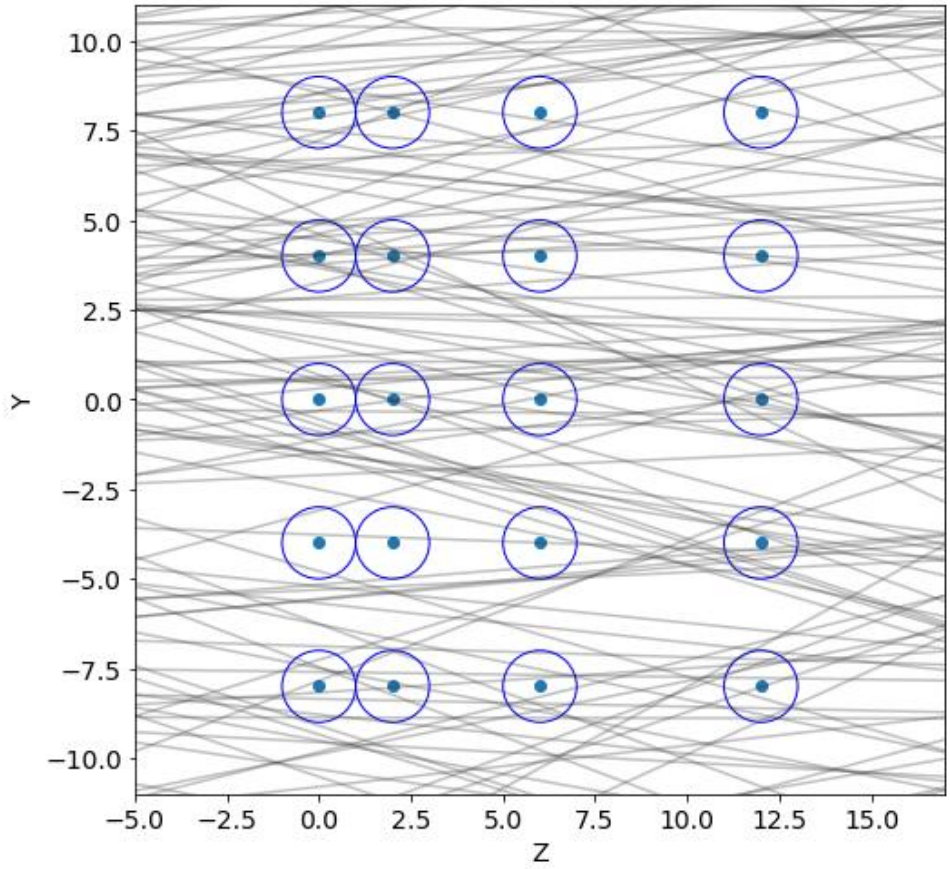

Рисунок 9. Траектории частиц с исходным расположением трубок в детекторе

В качестве метрики качества ($score$) для построенной геометрии системы было выбрано отношение количества траекторий, которые минимум дважды попали в трубки системы (N_{tracks_hits}), к общему количеству траекторий (N_{tracks}):

$$score = \frac{N_{tracks\_hits} \geq 2}{N_{tracks}} \tag{22}$$

Таким образом, задача оптимизации строения детектора сводится к поиску значений параметров y_1, y_2, y_3, z_1, z_2, z_3, по которым удастся построить систему, обеспечивающую наибольшее значение метрики $score$.

Для достижения указанной цели была написана функция оптимизации, которая на вход получает значения параметров y_1, y_2, y_3, z_1, z_2, z_3, а возвращает значение $1 - score$. Следовательно, выбранный метод машинного обучения должен минимизировать выходное значение целевой функции оптимизации.

Далее представлены реализации решения данной задачи оптимизации несколькими методами: поиском по сетке, методом случайного поиска, Байесовской оптимизацией с Гауссовскими процессами.

Поиск по сетке. Поиск по сетке (Grid Search) является традиционным методом осуществления оптимизации гиперпараметров, который делает полный перебор по заданному вручную подмножеству пространства гиперпараметров обучающего алгоритма.

Алгоритм решения данной задачи методом поиска по сетке следующий:

1. Задается количество уникальных значений для каждого параметра — размерность сетки
2. Задается сетка параметров в диапазоне заданных значений
3. Создается список для хранения результатов оптимизации
4. В цикле по каждому значению для каждого параметра вычисляется наименьшее значение целевой функции, и сохраняются значения параметров, обеспечивающие найденный минимум целевой функции

График значений целевой функции, полученных в ходе выполнения алгоритма:

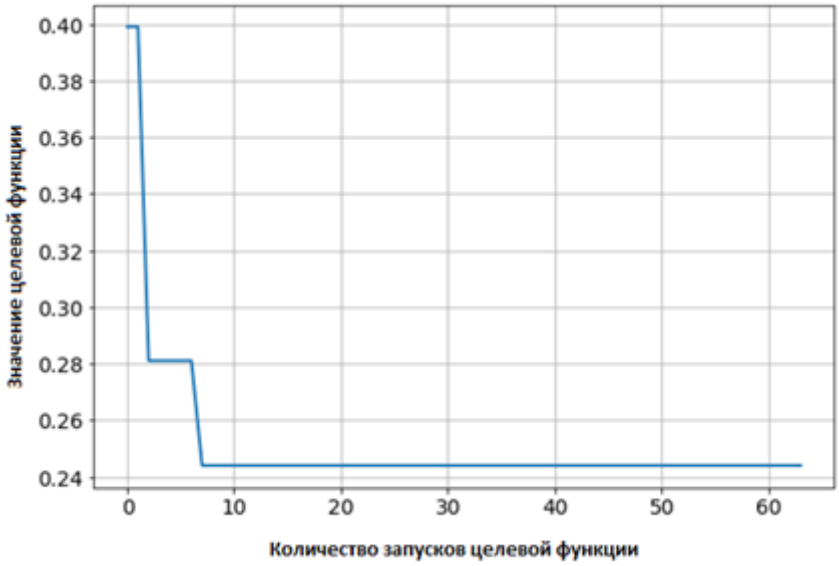

Рисунок 10. График зависимости значения целевой функции от количества ее запусков в алгоритме поиска по сетке

Полученный оптимум целевой функции и соответствующие значения управляющих параметров приведены в Таблице 3:

Минимальное значение целевой функции	0.244
y_1	0.0
y_2	0.0
y_3	0.0
z_1	10.0
z_2	10.0
z_3	10.0

Таблица 3. Результаты алгоритма поиска по сетке

Оптимальное расположение трубок в детекторе согласно полученным результатам алгоритма:

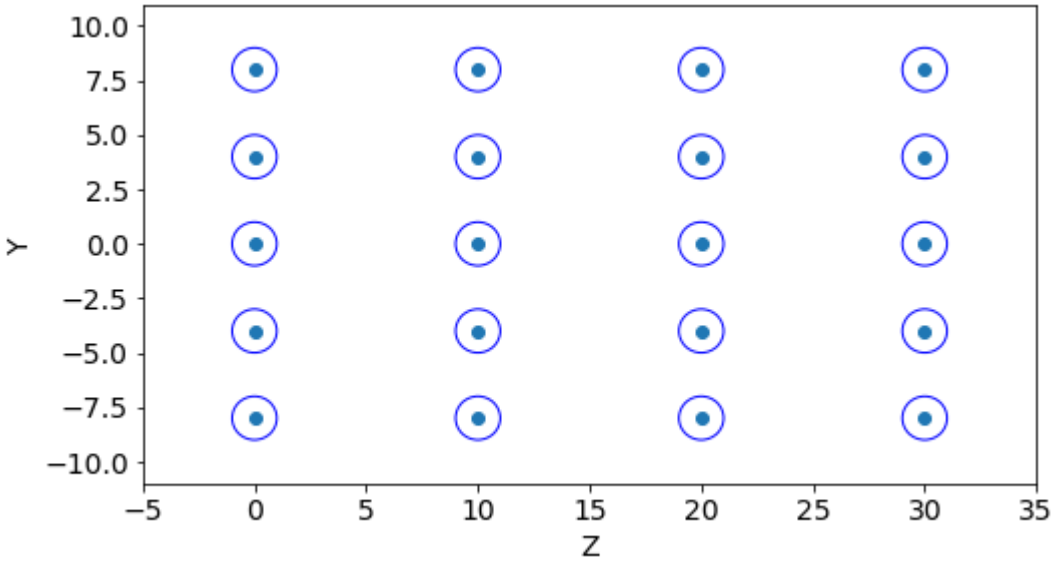

Рисунок 11. Оптимальное расположение трубок в детекторе согласно алгоритму поиска по сетке

Случайный поиск. Вместо генерации сетки значений параметров в данном подходе значения параметров генерируются случайным образом из заданного пространства значений [57].

Алгоритм решения данной задачи методом случайного поиска следующий:

1. Задается количество генерируемых точек N (значений параметров)
2. Для каждого управляющего параметра случайным образом генерируется N значений из заданного диапазона
3. Создается список для хранения результатов оптимизации
4. В цикле по каждому из N сгенерированных наборов значений управляющих параметров вычисляется наименьшее значение целевой функции, и сохраняются значения параметров, обеспечивающие найденный минимум целевой функции

График значений целевой функции, полученных в ходе выполнения алгоритма:

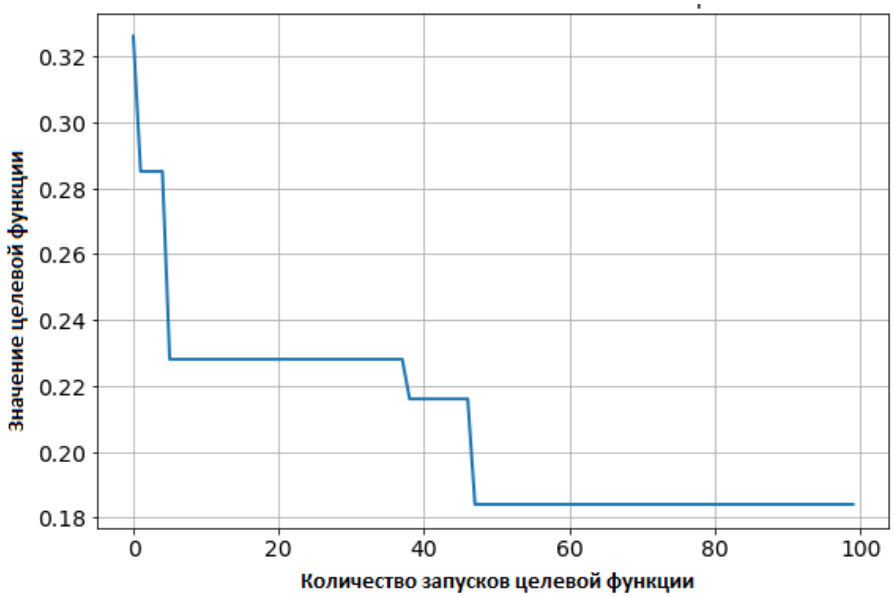

Рисунок 12. График зависимости значения целевой функции от количества ее запусков в алгоритме случайного поиска

Полученный оптимум целевой функции и соответствующие значения управляющих параметров приведены в Таблице 4:

Минимальное значение целевой функции	0.18500000000000005
y_1	1.4958925957956615
y_2	2.409155514576844
y_3	1.9471804856632257

z_1	2.247045068566281
z_2	4.075956186309445
z_3	3.1705657406063437

Таблица 4. Результаты алгоритма случайного поиска

Оптимальное расположение трубок в детекторе согласно полученным результатам алгоритма:

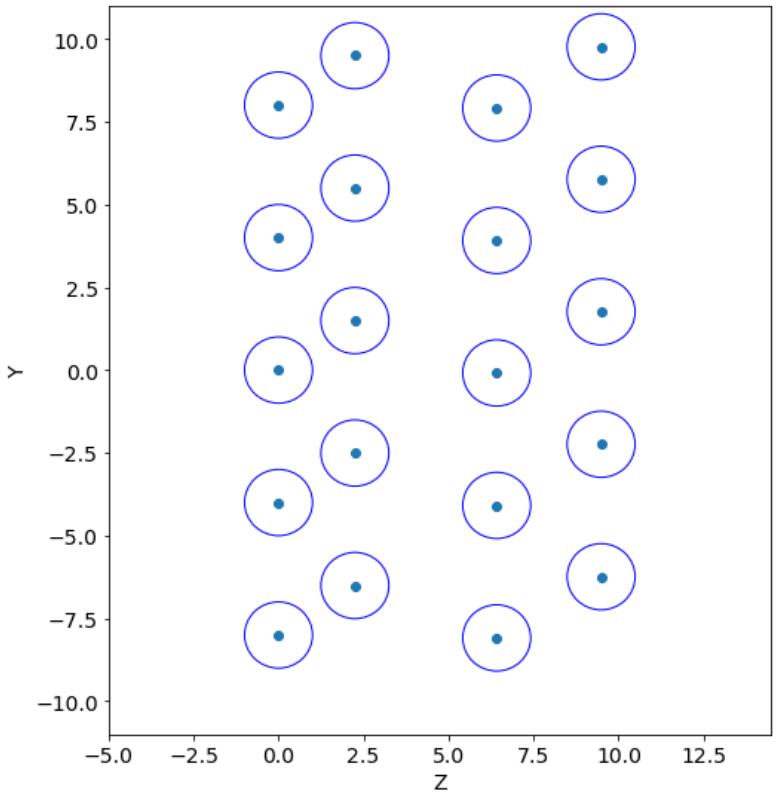

Рисунок 13. Оптимальное расположение трубок в детекторе согласно алгоритму случайного поиска

Байесовская оптимизация с Гауссовскими процессами. Описанная в разделе 2.4 идея Байесовской оптимизации с Гауссовскими процессами реализована в Python-библиотеке с открытым доступом — skopt [58]. В качестве функции выбора данных в алгоритме оптимизации используется функция LCB (14), в которой параметр k определяет баланс между эксплорацией и эксплуатацией при оптимизации.

Алгоритм решения данной задачи методом Байесовской оптимизации с Гауссовскими процессами следующий:

1. Задаются границы значений для каждого управляющего параметра
2. Вызывается функция skopt.gp_minimize [58]. Параметры вызова функции описаны в Таблице 5.

Наименование параметра	Описание параметра
func	Целевая функция оптимизации
dimensions	Границы значений управляющих параметров целевой функции
n_calls	Количество вызовов целевой функции оптимизации
n_random_starts	Количество случайных начальных точек
acq_func	Функция выбора данных в оптимизации
random_state	Случайное начальное значение
kappa	Управляющий параметр функции LCB (14)
noise	Уровень зашумленности данных
n_jobs	Количество ядер для параллельной работы при выполнении оптимизаци

Таблица 5. Параметры функции Байесовской оптимизации с Гауссовскими процессами (skopt.gp_minimize)

График значений целевой функции, полученных в ходе выполнения алгоритма:

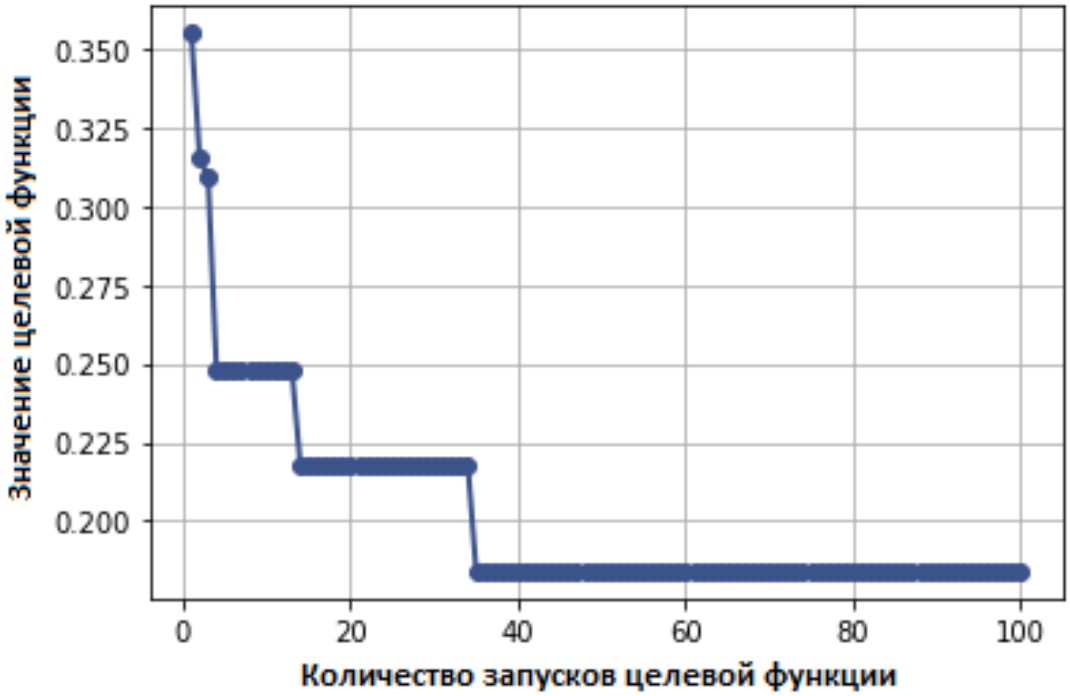

Рисунок 14. График зависимости значения целевой функции от количества ее запусков в алгоритме Байесовской оптимизации с Гауссовскими процессами

Полученный оптимум целевой функции и соответствующие значения управляющих параметров приведены в Таблице 6:

Минимальное значение целевой функции	0.17900000000000005
y_1	2.1458925957956615
y_2	2.979155514576844
y_3	1.1971804856632257
z_1	4.234045068566281
z_2	7.193956186309445
z_3	1.8905657406063437

Таблица 6. Результаты алгоритма Байесовской оптимизации с Гауссовскими процессами

Оптимальное расположение трубок в детекторе согласно полученным результатам алгоритма:

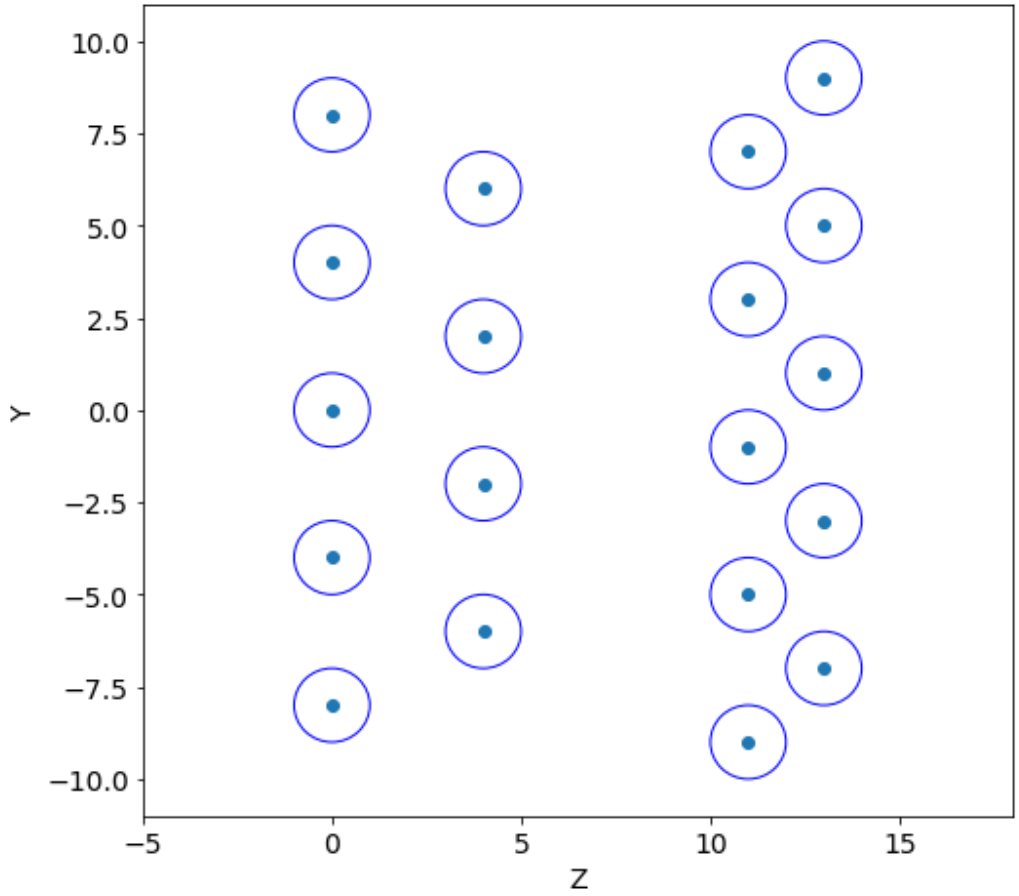

Рисунок 15. Оптимальное расположение трубок в детекторе согласно алгоритму Байесовской оптимизации с Гауссовскими процессами

Заключение к Главе 2

В данной главе сформулирована задача оптимизации детектора, приведены необходимые формулы и определения, описан практический пример решения поставленной задачи оптимизации методами поиска по сетке, случайного поиска и Байесовской оптимизацией с Гауссовскими процессами.

Выводы

В данном исследовании с помощью методов машинного обучения были решены две задачи — задача идентификации частицы по ее откликам в детекторных системах и задача оптимизации структуры детектора.

Задача идентификации частицы по ее откликам в детекторных системах была решена методами адаптивного и градиентного бустинга. Результаты работы данных алгоритмов приведены в Таблице 7:

Наименование алгоритма	Значение логистической функции ошибки
Адаптивный бустинг	0.53283105461409257
Градиентный бустинг	0.55793306232418781

Таблица 7. Результаты работы алгоритмов машинного обучения над задачей идентификации частицы

Значение логистической функции ошибки для результатов адаптивного бустинга получилось меньше, чем для градиентного бустинга, следовательно адаптивный бустинг более точно классифицировал частицы.

Задача оптимизации структуры детектора была решена методами случайного поиска, поиска по сетке, Байесовской оптимизацией с Гауссовскими процессами. Результаты работы данных алгоритмов приведены в Таблице 8:

Наименование алгоритма	Оптимум целевой функции
Случайный поиск	0.18500000000000005
Поиск по сетке	0.244
Байесовская оптимизация с Гауссовскими процессами	0.17900000000000005

Таблица 8. Результаты работы алгоритмов машинного обучения над задачей оптимизации структуры детектора

Наименьший оптимум целевой функции получен методом Байесовской оптимизации с Гауссовскими процессами, следовательно данный метод наиболее эффективен в поставленной задаче оптимизации структуры детектора.

Заключение

Данное исследование является продолжением предыдущей научной деятельности в бакалавриате и магистратуре и посвящено применению машинного обучения и интеллектуального анализа данных в задачах ускорительной физики. Основные результаты исследования:

- Рассмотрены основные этапы сбора и анализа данных, которые происходят в экспериментах Большого адронного коллайдера.
- Описаны различные системы детекторов, как они работают, и какие параметры частиц измеряют.
- Определен спектр задач физики элементарных частиц, которые могут быть эффективно решены методами машинного обучения.
- Решена задача идентификации частицы с помощью таких методов машинного обучения, как градиентный и адаптивный бустинг. Результаты вычислений приведены в данной работе. На основании полученных результатов выбран наиболее точный классификатор.
- Решена задача оптимизации структуры детекторной системы с помощью методов Байесовской оптимизации с Гауссовскими параметрами, поиска по сетке и случайного поиска. Результаты вычислений приведены в данной работе. На основании результатов оптимизации выбрана наилучшая модель.

Полученные результаты и используемые подходы машинного обучения могут быть применимы в решении задач из различных предметных областей: банковская сфера, торговля товарами и услугами, социальные сети и т.д.

В заключении автор выражает благодарность за наставления и помощь в написании данной работы своему научному руководителю, доктору физико-математических наук, профессору С.Н.Андрианову

Список литературы

1. Клёнов Г И, Хорошков В С "Адронная лучевая терапия: история, статус, перспективы" УФН 186 891–911 (2016).
2. (2021) CERN Accelerating science, URL: https://home.cern (Дата обращения: 31.05.2021).
3. (2021) Объединенный институт ядерных исследований | Наука сближает народы, URL: http://www.jinr.ru (Дата обращения: 31.05.2021).
4. (2021) The Large Hadron Collider | CERN, URL: https://home.cern/science/accelerators/large-hadron-collider (Дата обращения 31.05.2021).
5. (2021) ALICE | CERN, URL: https://home.cern/science/experiments/alice (Дата обращения: 31.05.2021).
6. (2021) ATLAS | CERN, URL: https://home.cern/science/experiments/atlas (Дата обращения: 31.05.2021).
7. (2021) CMS | CERN, URL: https://home.cern/science/experiments/cms (Дата обращения: 31.05.2021).
8. (2021) LHCb | CERN, URL: https://home.cern/science/experiments/lhcb (Дата обращения: 31.05.2021).
9. Berz M., Makino K. COSY INFINITY 9.1 Programmer’s Manual. — Michigan State University, 2011. — 93 p.
10. Iselin F. The MAD Program - Physical Methods Manual. — 1994. — 73 p.
11. (2021) GENIE Neutrino Monte Carlo Generator and Global Analysis of Neutrino Scattering Data, URL: http://www.genie-mc.org/ (Дата обращения 31.05.2021).
12. (2021) PYTHIA 8.3, URL: http://home.thep.lu.se/Pythia/ (Дата обращения 31.05.2021).
13. (2021) Overview | geant4.web.cern.ch, URL: https://geant4.web.cern.ch/ (Дата обращения 31.05.2021).

14. N.Malitsky and R.Talman: Unified Accelerator Libraries, AIP 391, Williamsburg, 1996
15. E. Krushinevskii, S. Andrianov, A. Ivanov, N. Kulabukhova, E. Sboeva, “Software-computing System for Numerical Modelling of Beam Dynamics in Accelerators”, Proceedings of IPAC2018, Vancouver, BC, Canada, April - May 2018, paper THPAK087
16. Mathematical and computer modelling of the nonlinear dynamics of particle beams in cyclic accelerators. / Andrianov, S. N.; Krushinevskii, E. A. в: Journal of Physics: Conference Series, Том 1391, № 1, 012036, 13.12.2019
17. (2021) Gradient boosting - Wikipedia, URL: https://en.wikipedia.org/wiki/Gradient_boosting (Дата обращения 15.05.2021).
18. (2021) AdaBoost - Wikipedia, URL: https://en.wikipedia.org/wiki/AdaBoost (Дата обращения 15.05.2021).
19. (2021) Bayesian optimization - Wikipedia, URL: https://en.wikipedia.org/wiki/Bayesian_optimization (Дата обращения 15.05.2021).
20. (2021) Gaussian process - Wikipedia, URL: https://en.wikipedia.org/wiki/Gaussian_process (Дата обращения 15.05.2021).
21. Matthew D. Schwartz (2021) 'Modern Machine Learning and Particle Physics', (2644-2353), URL: https://arxiv.org/abs/2103.12226.
22. D. Guest, K. Cranmer, D. Whiteson (2018) 'Deep Learning and its Application to LHC Physics', DOI: 10.1146/annurev-nucl- 101917-021019.
23. Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, Maurizio Pierini (2019) 'Learning representations of irregular particle-detector geometry with distance-weighted graph networks', URL: DOI:10.1140/epjc/s10052-019-7113-9 .

24. Yogesh Verma, Satyajit Jena (2021) 'Shower Identification in Calorimeter using Deep Learning', URL: https://arxiv.org/abs/2103.16247 .
25. Steven Farrell, Paolo Calafiura, Mayur Mudigonda, Prabhat, Dustin Anderson, Jean-Roch Vlimant, Stephan Zheng, Josh Bendavid, Maria Spiropulu, Giuseppe Cerati, Lindsey Gray, Jim Kowalkowski, Panagiotis Spentzouris, Aristeidis Tsaris (2018) 'Novel deep learning methods for track reconstruction', URL: https://arxiv.org/abs/1810.06111 .
26. (2021) Recurrent neural network - Wikipedia, URL: https://en.wikipedia.org/wiki/Recurrent_neural_network (Дата обращения 15.05.2021).
27. Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun (2018) 'Graph Neural Networks: A Review of Methods and Applications', URL: arXiv:1812.08434 .
28. Sabrina Amrouche, Laurent Basara, Paolo Calafiura, Victor Estrade, Steven Farrell, Diogo R. Ferreira, Liam Finnie, Nicole Finnie, Cécile Germain, Vladimir Vava Gligorov, Tobias Golling, Sergey Gorbunov, Heather Gray, Isabelle Guyon, Mikhail Hushchyn, Vincenzo Innocente, Moritz Kiehn, Edward Moyse, Jean-Francois Puget, Yuval Reina, David Rousseau, Andreas Salzburger, Andrey Ustyuzhanin, Jean-Roch Vlimant, Johan Sokrates Wind, Trian Xylouris, Yetkin Yilmaz (2021) 'The Tracking Machine Learning challenge : Accuracy phase', URL: DOI:10.1007/978-3-030-29135-8_9 .
29. Laurits Tani, Diana Rand, Christian Veelken, Mario Kadastik (2021) 'Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics', URL: DOI:10.1140/epjc/s10052-021-08950-y .
30. Dawit Belayneh, Federico Carminati, Amir Farbin, Benjamin Hooberman, Gulrukh Khattak, Miaoyuan Liu, Junze Liu, Dominick Olivito, Vitória Barin Pacela, Maurizio Pierini, Alexander Schwing, Maria Spiropulu, Sofia

Vallecorsa, Jean-Roch Vlimant, Wei Wei, Matt Zhang (2019) 'Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics', URL: 10.1140/epjc/s10052-020-8251-9 .

31. E. Cisbani, A. Del Dotto, C.Fanelli, M. Williams, M. Alfred, F. Barbosa, L. Barion, V. Berdnikov, W. Brooks, T. Cao, M. Contalbrigo, S. Danagoulian, A. Datta, M. Demarteau, A. Denisov, M. Diefenthaler, A. Durum, D. Fields, Y. Furletova, C. Gleason, M. Grosse-Perdekamp, M. Hattawy, X. He, H. van Hecke, D. Higinbotham, T. Horn, C. Hyde, Y. Ilieva, G. Kalicy, A. Kebede, B. Kim, M. Liu, J. McKisson, R. Mendez, P. Nadel-Turonski, I. Pegg, D. Romanov, M. Sarsour, C.L. da Silva, J. Stevens, X. Sun, S. Syed, R. Towell, J. Xie, Z.W. Zhao, B. Zihlmann, C. Zorn (2020) 'AI-optimized detector design for the future Electron-Ion Collider: the dual-radiator RICH case', URL: DOI:10.1088/1748-0221/15/05/P05009 .
32. Csaba Balázs, Melissa van Beekveld, Sascha Caron, Barry M. Dillon, Ben Farmer, Andrew Fowlie, Eduardo C. Garrido-Merchán, Will Handley, Luc Hendriks, Guðlaugur Jóhannesson, Adam Leinweber, Judita Mamužić, Gregory D. Martinez, Sydney Otten, Pat Scott, Roberto Ruiz de Austri, Zachary Searle, Bob Stienen, Joaquin Vanschoren, Martin White (2021) 'A comparison of optimisation algorithms for high-dimensional particle and astrophysics applications', URL: arXiv:2101.04525 .
33. (2021) LHCb - Calorimeters, URL: https://lhcb-public.web.cern.ch/en/detector/calorimeters-en.html (Дата обращения 15.05.2021).
34. (2021) LHCb - Muon system, URL: https://lhcb-public.web.cern.ch/en/detector/Muon-en.html (Дата обращения 15.05.2021).
35. (2021) LHCb - Ring Imaging Cherenkov (RICH) detectors, URL: https://lhcb-public.web.cern.ch/en/detector/RICH-en.html (Дата обращения 15.05.2021).

36. (2021) LHCb - Tracking system, URL: 1https://lhcb-public.web.cern.ch/en/Detector/Trackers2-en.html (Дата обращения 15.05.2021).
37. (2021) Преобразование Хафа — Википедия, URL: https://ru.wikipedia.org/wiki/Преобразование_Хафа (Дата обращения 15.05.2021).
38. (2021) Kalman filter - Wikipedia, URL: https://en.wikipedia.org/wiki/Kalman_filter (Дата обращения 15.05.2021).
39. (2021) Artificial Neural Network - Hopfield Networks - Tutorialspoint, URL: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_hopfield.htm (Дата обращения 15.05.2021).
40. (2021) Convolutional neural network - Wikipedia, URL:https://en.wikipedia.org/wiki/Convolutional_neural_network (Дата обращения 15.05.2021).
41. (2021) Эффект Вавилова — Черенкова — Википедия, URL: https://ru.wikipedia.org/wiki/Эффект_Вавилова_-_Черенкова (Дата обращения 15.05.2021).
42. (2021) Welcome to Python.org, URL: https://www.python.org/ (Дата обращения 15.05.2021).
43. (2021) NumPy, URL: https://numpy.org/ (Дата обращения 15.05.2021).
44. (2021) pandas - Python Data Analysis Library, URL: https://pandas.pydata.org/ (Дата обращения 15.05.2021).
45. (2021) Scikit-Learn: machine learning in Python, URL: https://scikit-learn.org/ (Дата обращения 15.05.2021).
46. (2021) Открытый набор данных LHC, URL: https://github.com/hse-aml/hadron-collider-machine-learning/releases/tag/Week_2 (Дата обращения 15.05.2021).
47. (2021) Cross entropy - Wikipedia, URL: http://en.wikipedia.org/wiki/Cross_entropy (Дата обращения 15.05.2021).

48. (2021) sklearn.metrics.log_loss, URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html (Дата обращения 15.05.2021).
49. (2021) sklearn.ensemble.AdaBoostClassifier, URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html (Дата обращения 15.05.2021).
50. (2021) sklearn.ensemble.GradientBoostingClassifier, URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html (Дата обращения 15.05.2021).
51. (2021) Receiver operating characteristic - Wikipedia, URL: https://en.wikipedia.org/wiki/Receiver_operating_characteristic (Дата обращения 15.05.2021).
52. (2021) Grid Search, URL: https://scikit-learn.org/stable/modules/grid_search.html (Дата обращения 15.05.2021).
53. (2021) Условное распределение — Википедия, URL: https://ru.wikipedia.org/wiki/Условное_распределение (Дата обращения 15.05.2021).
54. (2021) Маргинальное распределение — Циклопедия, URL: http://cyclowiki.org/wiki/Маргинальное_распределение (Дата обращения 15.05.2021).
55. (2021) Lower Confidence Bound, URL: https://www.sciencedirect.com/topics/mathematics/lower-confidence-bound (Дата обращения 15.05.2021).
56. (2021) scikit-optimize, URL: https://scikit-optimize.github.io/ (Дата обращения 15.05.2021).
57. (2021) Random Search, URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html (Дата обращения 15.05.2021).

58. (2021) Bayesian optimization using Gaussian Processes, URL: http://scikit-optimize.github.io/stable/modules/generated/skopt.gp_minimize.html (Дата обращения 15.05.2021).