Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

АЛГОРИТМЫ НАСТРОЙКИ ГИПЕРПАРАМЕТРОВ НА ОСНОВЕ ОБЪЕДИНЕНИЯ АПРИОРНЫХ И АПОСТЕРИОРНЫХ ЗНАНИЙ О ЗАДАЧЕ

Автор: Смирнова Валентина Сергеевна _____

Направление подготовки: 01.03.02 Прикладная математика и информатика

Квалификация: Бакалавр

Руководитель: Фильченков А.А., к.ф.-м.н _____

К защите допустить

Руководитель ОП Парфенов В.Г., проф., д.т.н. _____

«____» _____ 20___ г.

Санкт-Петербург, 2020 г.

Обучающийся Смирнова В.С. Группа М3435 Факультет ИТиП

Направленность (профиль), специализация
Математические модели и алгоритмы в разработке программного обеспечения

ВКР принята «____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «____» _____ 20___ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

УТВЕРЖДАЮ

Руководитель ОП

проф., д.т.н. Парфенов В.Г. _____

«____» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Смирнова В.С.

Группа М3435 **Факультет/институт/кластер** ИТиП

Квалификация Бакалавр

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) образовательной программы Математические модели и алгоритмы в разработке программного обеспечения

Специализация

Тема ВКР Алгоритмы настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

Руководитель Фильченков А.А., к.ф.-м.н, доцент факультета информационных технологий и программирования, Университет ИТМО

2 Срок сдачи студентом законченной работы до «____» _____ 20___ г.

3 Техническое задание и исходные данные к работе

Требуется разработать алгоритм настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

4 Содержание выпускной работы (перечень подлежащих разработке вопросов)

В работе должна быть показана эффективность разработанного решения по сравнению с сущесствующими

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

а) Hutter, F., Hoos, H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. // LION'11. — 2011

б) Efficient and robust automated machine learning / M. Feurer // Advances in neural information processing systems. — 2015.

в) Leite R., Brazdil P. Active Testing Strategy to Predict the Best Classification Algorithm via Sampling and Metalearning. // ECAI. — 2010.

7 Дата выдачи задания «____» _____ 20___ г.

Руководитель ВКР _____

Задание принял к исполнению _____

«____» _____ 20___ г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся Смирнова Валентина Сергеевна

Наименование темы ВКР Алгоритмы настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

Наименование организации, где выполнена ВКР: Национальный Исследовательский Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Разработать алгоритм настройки гиперпараметров на основе объединения априорных и апостериорных знаний о задаче

2 Задачи, решаемые в ВКР:

а) изучить существующие решения поставленной задачи;

б) предложить и реализовать новый алгоритм;

в) провести эксперименты, показывающие эффективность решения.

3 Число источников, использованных при составлении обзора: 0

4 Полное число источников, использованных в работе: 9

5 В том числе источников по годам:

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	6	3	0

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	**Раздел работы**
Пакет `numpy`	Не предусмотрено
Пакет `pandas`	Не предусмотрено
Пакет `robo`	Не предусмотрено
Пакет `matplotlib`	Не предусмотрено
Пакет `george`	Не предусмотрено

8 Краткая характеристика полученных результатов

Был предложен и реализован эффективный алгоритм, решающий поставленную задачу.

9 Гранты, полученные при выполнении работы

Отсутствуют

10 Наличие публикаций и выступлений на конференциях по теме работы

а) IX Конгресс Молодых Учёных

Обучающийся Смирнова В.С. _____

Руководитель ВКР Фильченков А.А. _____

«____» _____ 20___ г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача классификации – постросить алгоритм (классификатор), который по набору признаков вернул бы метку класса или вектор оценок принадлежности (апостериорных вероятностей) к каждому из классов. Оновная её цель – максимально точно определить метку класса для заданного объекта. Задача широко применяется во многих областях:

- Медицинская диагностика: по набору медицинских характеристик требуется поставить диагноз
- Геологоразведка: по данным зондирования почв определить наличие полезных ископаемых
- Оптическое распознавание текстов: по отсканированному изображению текста определить цепочку символов, его формирующих
- Кредитный скоринг: по анкете заемщика принять решение о выдаче/отказе кредита
- Синтез химических соединений: по параметрам химических элементов спрогнозировать свойства получаемого соединения

Найти и обучить эффективный алгоритм классификации – трудоёмкая задача, которая включает в себя выбор самого классификатора, сбор и разметку данных (в случае обучения с учителем), и непосредственно настройку гиперпараметров классификатора. Так как гиперпараметры задаются до начала обучения и не изменяются в его ходе, а при этом могут существенно влиять на результат обучения, то повляется задача оптимизации гиперпараметров.

Оптимизация гиперпараметров — задача машинного обучения по выбору набора оптимальных гиперпараметров для обучающего алгоритма. Одни и те же виды моделей машинного обучения могут требовать различные предположения, веса или скорости обучения для различных видов данных. Эти параметры называются гиперпараметрами и их следует настраивать так, чтобы модель могла оптимально решить задачу обучения. Для этого находится кортеж гиперпараметров, который даёт оптимальную модель, оптимизирующую заданную функцию потерь на заданных независимых данных. Такая задача несёт название AutoML [4], основная её задача – сделать процесс машинного обучения доступным не только для экспертов в области ML, но и для любого пользователя.

Интерес к задаче AutoMl растет, проводятся конференции и соревнования по её решению. В частности, каждые два года проходит соревнование AutoML Challenge. В августе 2019 года происходило мероприятие, посвящённое этой теме – The Third International Workshop on Automation in Machine Learning. Согласно выпуску Forbes за декабрь 2018 года, это один из пяти трендов в развитии машинного обучения в 2019 году. Тема AutoML появляется все чаще и чаще в дискуссиях и публикациях. Решения этой задачи уже используются в автономных машинах, предсказании цен и многих других областях.

Существующие решения [2, 3, 5, 7] для задачи оптимизации гиперпараметров основываются на случайной расстановке гиперпараметров и дальнейшей их настройке. В данной работе будет предложен подход, основанный не на случайной расстановке первичных гиперпараметров, а на особом подходе, основанном на результатах обучения смежных задач.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Более подробно рассмотрим задачу классификации, алгоритмы ее решения и способы оценки качества полученной классификации. Также опишем понятие модели алгоритма классификации и основные методы настройки её гиперпараметров, применимые ко многим алгоритмам машинного обучения. Заметим, что на сегодняшний момент не предложено способов не случайной расстановки гиперпараметров и дальнейшей их настройки, на основе результатов обучения на схожих задачах.

1.1. Определения и ключевые понятия

Для начала введем определения и ключевые понятия, которые будут использоваться в дальнейшей работе.

- классификатор – параметризованный алгоритм, решающий задачу классификации
- конфигурация – фиксированный набор параметров классификатора
- алгоритм – пара из классификатора и его конфигурации
- оптимизатор – алгоритм, который используется для оптимизации гиперпараметров
- решаемая задача – алгоритм с настроенными гиперпараметрами на конкретном датасете
- решённая задача – алгоритм с оптимизированными гиперпараметрами на конкретном датасете
- соседняя задача – ближайшая задача к решаемой
- текущее решение (текущая задача) – решаемая задача, для которой могут использоваться сведения из решённых (соседних) задач

1.2. Обзор классификаторов

В данной части рассмотрим популярные модели для решения задачи классификации. Вспомним, что цель задачи классификации – наиболее точно определять метку класса по заданному объекту.

Итак, наиболее используемые на сегодняшний момент модели классификаторов:

Линейная регрессия Можно представить в виде уравнения, которое описывает прямую, наиболее точно показывающую взаимосвязь между входными переменными X и выходными переменными Y

Логистическая регрессия По аналогии с линейной регрессией требуется найти коэффициенты для входных данны, но уже с помощью нелинейной или логистической функции.

Линейный дискриминантный анализ (LDA) Состоит из статистических свойств данных, рассчитанных для каждого класса

— Среднее значение для каждого класса
— Дисперсия, рассчитанная по всем классам

Деревья принятия решений Представима в виде бинарного дерева, где каждый узел представляет собой входную переменную и точку разделения для этой переменной (при условии, что переменная — число)

Наивный Байесовский классификатор Состоит из двух типов вероятностей, которые рассчитываются с помощью тренировочных данных:

а) Вероятность каждого класса
б) Условная вероятность для каждого класса при каждом значении x

K-ближайших соседей (KNN) Предсказание метки класса делается на основе меток k ближайших соседей

Метод опорных векторов (SVM) Суть метода заключается в построении гиперплоскости, разделяющей классы

Бэггинг и случайный лес (RandomForest) Один из наиболее эффективных алгоритмов классификации, берётся множество подвыборок из данных, считается среднее значение для каждой, а затем усредняются результаты для получения лучшей оценки действительного среднего значения

Бустинг и AdaBoost : принадлежит семейству ансамблевых алгоритмов, суть которых заключается в создании сильного классификатора на основе нескольких слабых

Многослойный персептрон (Multilayered perceptron) Класс искусственных нейронных сетей прямого распространения, состоящих как минимум из трех слоёв: входного, скрытого и выходного

В работе мы заострим внимание на модели случайного леса как самой эффективной на сегодняшний момент и модели многослойного парцептрона, так как он обладает наибольшим количеством гиперпараметров и также является достаточно эффективным.

1.3. Меры оценки качества классификации

Кроме выбора алгоритма классификации и его гиперпараметров, обучить одель, нужно ещё каким-то образом оценить качество работы обученного алгоритма. Для этого датасет делится на 2 части: *train* (на которой модель обучается) и *test* или *validate* (на которой оценивается качество классификации). В данной части мы рассмотрим существующие меры оценки качества классификации и выделим сущесственные для нашей задачи. Для этого сначала вспомним базовые понятия:

— Верно-положительными (TP) называются объекты, которые были классифицированы как положительные и действительно являются таковыми
— Верно-отрицательными (TN) называются объекты, которые были классифицированы как отрицательные и действительно таковые
— Ложно-положительными (FP) называются объекты, которые были классифицированы как положительные, но фактически отрицательные
— Ложно-отрицательными (FN) называются объекты, которые были классифицированы как отрицательные, но фактически положительные

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Одна из наиболее простых и популярных мер оценки качества – *F-мера* или *F-score*. Считается она следующим образом:

$$F - score = 2 * \frac{precision * recall}{precision + recall} \quad (4)$$

Приемущество данной меры в том, что её достаточно просто считать и результат отлично подходит для целевой функции оптимизации, о которой мы поговорим в следующей части.

Также существует *кривая ошибки* или *ROC-curve (Receiver Operating Characteristic)*. Суть данной меры состоит в том, что считается площать кривой под графиком уровня верно-положительных предсказанных экземпляров от уровня ложно-положительных. Не будем заострять на ней внимание, так как к нашей задаче она не подходит из-за трудоёмкости расчёта.

1.4. Обзор подходов к оптимизации гиперпараметров

Существует несколько подходов к задаче оптимизации гиперпараметров. В данной части рассмотрим наиболее популярные из них, оценим приемущества и недостатки.

Поиск по решётке По сути, данный алгоритм делает полный перебор всех возможных моделей и конфигураций.

Доступные реализации

— LIBSVM
— scikit-learn
— Talos

Достоинства Гарантированна будет найдена наилучшая конфигурация.

Недостатки Слишком большие затраты на обучение.

Случайный поиск Отличается от поиска по решётке тем, что идёт не полный перебор всех конфигурации, а случайная их выборка.

Доступные реализации

— hyperopt
— scikit-learn

— H2O AutoML
— Talos

Достоинства Меньшие затраты на обучение. Существует вероятность нахождения наилучшей конфигурации за наименьшее время.

Недостатки Неопределённое количество времени на поиск наилучшей конфигурации.

Байесовская оптимизация Метод, основанный на обращении к функции «чёрного ящика» с шумом. Для задачи оптимизации гиперпараметров строит стохастическую модель из отображения из конфигурации в целевую функцию, применённую на валидационном наборе данных.

Доступные реализации
— Spearmint
— Bayesopt
— MOE
— Auto-WEKA
— Auto-sklearn
— mlrMBO
— tuneRanger
— BOCS
— SMAC

Достоинства Небольшие затраты на обучение, количество итераций задаётся вручную, адаптируется под значимость каждого гиперпараметра для конкретной задачи.

Недостатки Сложность реализации и использования.

Оптимизация на основе градиентов Для определённых алгоритмов обучения вычисляется градиент гиперпараметров и оптимизируется с помощью градиентного спуска.

Доступные реализации

— hypergrad

Достоинства Небольшие затраты на обучение, неплохой результат.

Недостатки Сложность понимания и реализации, небольшое количество доступных реализаций.

Эволюционная оптимизация Также, как и Байесовская оптимизация, основывается на обращениях к функции «чёрного ящика» с шумом, однако для поиска гиперпараметров для заданного алгоритма использует эволюционные подходы (алгоритмы)[1].

Доступные реализации

— TPOT
— devol
— deap

Достоинства Небольшие затраты на обучение, неплохой результат.

Недостатки Используется только для статистических алгоритмов.

Давайте немного подробнее рассмотрим Байесовскую оптимизацию.

1.4.1. Сравнение существующих подходов к оптимизации

1.4.2. Байесовская оптимизация

Как уже говорилось в предыдущей части 1.4, Байесовская оптимизация основывается на обращении к функции «чёрного ящика» с шумом. Кроме того, подход требует определить модель, на которой будет основываться сама Байесовская оптимизация, максимизатор, способ задания первичных гиперпараметров и непосредственно целевую функцию. Сам алгоритм представляет из себя следующую цеопчку действий:

Точка в алгоритме определяется конфигурацией гиперпараметров. Лучшее значение – минимальное значение, выданное целевой функцией в данной точке. Наиболее интересная и значимая часть в алгоритме1 – выбор

Листинг 1 – Байесовская оптимизация

```
задать первичные гиперпараметры
for количество итераций do
    выбрать следующую точку для рассмотрения
    получить результат целевой функции в этой точке
    сохранить лучшее значение и конфигурацию
end for
```

следующей для рассмотрения точки. Именно в этой части модель обучается на уже рассмотреных точках и обновляет так называемую *функцию выгоды (acquisition function)*, после чего в максимуме функции выгоды будет место с наибольшей неопределённостью, значит, там нav и необходимо будет «смотреть» на точку. Подробнее про подход описано в AutoML Book[**automlbook19a**].

Модель Определяет, как будет обновляться пространство гиперпараметров модели оптимизатора. Важно отметить, что гиперпараметры оптимизатора никаким образом не связаны с гиперпараметрами классификатора (которые мы пытаемся оптимизировать). Наиболее распространённые модели:

а) GaussianProcess
б) GaussianProcessMCMC
в) RandomForest
г) WrapperBohamiann
д) DNGO

У каждой модели есть свои приемущества и свои недостатки, например, в Гауссовском процессе нет возможности работать с категориальными признаками, DNGO же обладает наименее точным результатом.

Максимизатор В Байесовской оптимизации работает с функцией выгоды, которая нахожит следующую наиболее интересную точку. Максимизаторы бывают:

а) RandomSampling
б) SciPyOptimizer
в) DifferentialEvolution

Функция выгоды Служит для определения наименее информативной области. В этой области меньше всего информации о значении функции, которую мы патаемся апроксимировать, поэтому в экстремуме функции выгоды будет находиться наиболее интересная на данный момент конфигурация гиперпараметров. Популярные функции выгоды:

а) EI
б) LogEI
в) PI
г) LCB

Целевая функция В общем случае принимает на вход конфигурацию гиперпараметров и отдаёт значение той самой функции «чёрного ящика», значение которой минимизируется в процессе Байесовской оптимизации. В случае рассматриваемой нами задачи классификасии, целевая функция будет обучать классификатор с полученными гиперпараметрами на третировочной выборке и вернёт обратное значение (так как функция минимизируется) меры оценки качества классификации, посчитанной на валидационной выборке.

Выводы по главе 1

В первой главе рассмотрены ключевые определения и понятия, необходимые для понимания решаемой задачи. Оценены приемущества и недостатки параметрв для оптимизации гиперпараметров, такие как классификаторы, меры оценки качества классификации, подходы к задаче оптимизации и их параметры. Таксе подробно рассмотрен один из основных и наиболее популярных подходов – Байесовская оптимизация. Описаны функции и задачи модели оптимизатора, его целевой функции, функции выгоды и максимизатора.

ГЛАВА 2. ПРЕДЛОЖЕННОЕ РЕШЕНИЕ

За основу предложенного решения была выбрана Байесовская оптимизация, а именно реализация *RoBO (Robust Bayesian Optimization Framework)*[9]. Для однозначного понимания, далее будем называть её «классической Байесовской оптимизацей», чтобы иметь возможность отличать от непосредственно предложенного рещения. Задача данного исследования – добиться лучших результатов оптимизации. Под «лучшими» результатами стоит понимать лучшее (минимальное) значение *incubment*[1], полученное на более ранней итерации оптимизации. Поэтому достаточно задать фиксированные параметры классического решения и сравнивать его результаты с результатами доработанного решения с теми же параметрами. Далее будут рассмотрены выбранные в работе параметры для классической Байесовской оптимизации.

2.1. Параметры для оптимизации

В первой главе 1.4.2 сравнили параметры, необходимые для запуска классической Байесовской оптимизации. В текущей главе будут выбраны наиболее подходящие нам параметры.

2.1.1. Модель оптимизации

В качестве модели оптимизации в работе используется Гауссовский процесса. Как известно, оптимизаторы, построенные на модели случайного лесав пользуются бóльшим спросом по причине того, что даёт лучшие результаты, однако, как очевидно из названия, данная модель основана на случайности, что не несёт под собой точного математического обоснования, в отличие от Гауссовского процесса. Именно по этой причине в работе исполуется именно Гауссовский процесс.

2.1.2. Классификатор

2.1.2.1. Обоснование

При выборе классификатора, стоит помнить, что мы решаем задачу оптимизации гиперпараметров, значит, набор гиперпараметров классификатора должен соответствовать следующим свойствам:

а) иметь ненулевой конечный набор гиперпараметров

б) настраевыемые гиперпарамтры должны быть вещественными числами

[1] значение целевой функции

в) не иметь категориальных гиперпараметров, так как в качестве модели оптимизатора выступает Гауссовский процесс, который не предусматривает работу с категориальными гиперпараметрами

К сожалению, не существует модели, удовлетворяющей всем перечисленным свойствам, поэтому было принято решение игнорировать (фиксировать определённое значение и не изменять в процессе всей отпимизации) категориальные гиперпараметры (представлять их в виде чисел было бы некорректно) и приводить числа с плавоющей точкой к целым на местах гиперпараметров, где требуются целочисленные значения (данное принебрежение корректно, так как значения гиперпараметров, требующие целочисленности, располагаются в диапазоне, много большем, чем потеря при округлении).

Кроме того, стоит помнить, что необходимо подобрать классификатор с как можно большим количеством подходящих гиперпараметро, чтобы была возможность корректно оценить работу оптимизатора. Более подробно: гиперпараметры бывают более значимы для классификатора или менее значимы, их значимость определяет именно оптимизатор в ходе оптимизации. Поэтому в выбранном классификаторе должны присутсстровать и те, и другие.

Проанализировав все требования к классификатору, был выбран многослойный парцептрон Румельхарта (Рисунок 1).

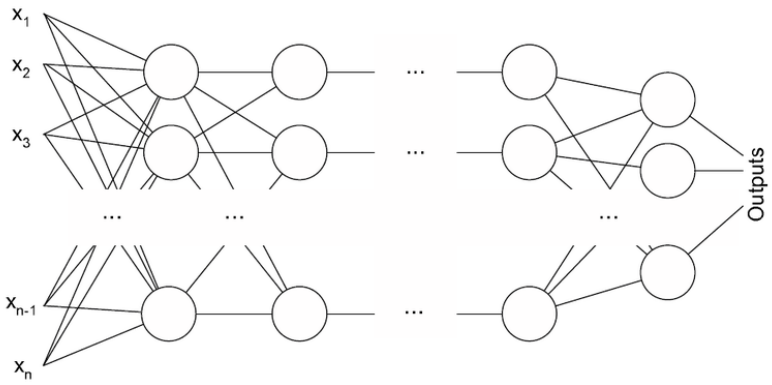

Рисунок 1 – Многослойный парцептрон Румельхарта

2.1.2.2. Набор гиперпараметров

Гиперпарметры многослойного парцептрона Румельхарта, которые будут использованы для оптимизации (в круглых скобках указан тип гиперпараметра, в квадратных – диапазон его значений):

— hidden layer sizes *(int)* – число скрытых слоёв [1, 150]

— α *(float)* – L2 штраф [0.00001, 0.01]
— learning rate initial *(float)* – начальная скорость обучения [0.0001, 0.1]
— max iterations *(int)* – максимальное число итераций [50, 300]
— validation fraction *(float)* – доля данных обучения, отведенных в качестве проверки для преждевременной остановки [0.01, 0.9]
— β_1 *(float)* – скорость экспоненциального убывания для оценок вектора первого момента [0.09, 0.9]
— β_2 *(float)* – Скорость экспоненциального убывания для оценок вектора второго момента [0.0999, 0.999]
— n iterations no change *(int)* – максимальное число эпох, пройденных без прогресса [5, 15]

2.1.3. Целевая функция

Как было упомянуто в части 1.4.2, целевая функция в нашем случае должна отдавать значение меры качества классификации, вернее обратное её значение, так как идёт процесс минимизации. Поэтому вернёмся к части 1.3 и выберем подходящую нам меру оценки качества классификации.

Мера оценки качества классификации Наиболее подходяей мерой в нашем случае является *F-score*, так как она проста в вычислении и от неё легко берётс обратное значение: *(1 – F-score)* – именно это значение будет выдаваться целевой функцией прикаждом обращении к ней. Более детально рассмотрим процесс, происходящий внутри целевой функции. До запуска оптимизации, функция инизиализируется выборкой, разделённой на 2 части: *train* и *validate*. Далее запускается оптимизационный процесс, где вызывается наша целевая функция. При каждом обращении, на вход функции подаётся конфигурация гиперпараметров, сгенерированная алгоритмом оптимизации для классификатора. После чего запускается процесс обучения классификатора с заданной конфигурацией на тренировочной части выборки, далее запускается процесс предсказания меток классов на валидационной части выборки, считается значение *F-score* и возвращается описанное ранее значение целевой функции. Затем оптимизатор анализирует полученный результат, перестраивает конфигурацию и повторяет до тех пор, пока не достигнет последней итерации.

2.2. Метрика для определения подобия задач

В работе планируется использовать не только априорные знания по решаемой задаче, но и апостериорные знания, полученные при решении других задач. Но так как задач (датасетов) существует бесконечное множество, то возникает вопрос: каким образом выбирать подходящие для нашей задачи и какую информацию из обучения использовать. Фактичести у всех датасетов разное число признаков, классов и радикально разные диапазоны значений. А нам необходимо выяснить не только насколько та или зая задача похожа на решаемую, но и дать количественную оценку подобию задач. в этой части мы рассмотрим меру оценки подобию задач.

Первое, что приходит на ум – рассчитать расстояние между задачами, однако для рассчёта рассстояний между задачами, необходимо, чтобы задачи имели одинаковую размерность. Для этого их необходимо привести к общему виду.

Один из способов – для каждого датасета рассчитать метапризнаки и уже по ним считать расстояния. Для этого для каждого признака возьмём некую характеристику его связи с классом, попарные корреляции между признаками и характеристики структуры дерева принятия решений. После чего на каждом полученном массиве посчитаем статистики (минимум, максимум, среднее и т.д.). полученный массив и будет массивом метапризнаков. Однако этого ещё не достаточно для рассчёта расстояния, так как значения по-прежнему довольно различаются. Поэтому необходимо также нормализовать массивы полученных метапризнаков. После нормализации мы получим не только подходящие значения, но и устраним ковариации и уравним дисперсию.В качестве нормализации используем Расстояние Махаланóбиса – это мера расстояния между векторами случайных величин, обобщающая понятие евклидова расстояния. Именно данный подход к нормализации умитывает в себе ковариации. Формально расстояние Махаланобиса он рассматриваемого вектора $x = (x_1,x_2,x_3...x_N)^T$ до множества со средним значением $\mu = (\mu_1,\mu_2,\mu_3...\mu_N)^T$ матрицей ковариации S определяется следующим образом:

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)} \tag{5}$$

Для удобства изначально домножим на $S^{-1/2}$, чтобы сократить затраны на рассчёт. После того, как мы получим набор нормализованных признаков, посчитаем обыкновенное Евклидово расстояние между задачами:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{6}$$

Теперь для каждой пары задач мы знаем расстоянииe между ними и можем использовать это расстояние для оценки подобия задач. Однако в реальности у нас не фиксированный набор задач, постоянно добавляются норые датасеты, а так как нормализация производится по фиксированному множеству задач, то добавление нового датасета может исказить уже посчитанные расстояния и придётся пересчитывать всё с самого начала. Казалось бы это не является универсальным решением, однако если учесть, что предпоститанных задач у нас много больше, чем приходящих (например, имеется 100 предпросчитанных задач и добавляется одна новая), то на значения факт добавления задачи повлияет незначительно и можно почтитать необходимые значения только для новой задачи.

2.3. Расширение Байесовской оптимизации

В параграфе 1.4 описан классический подход к Байесовской оптимизации, цель настоящей работы – расширить существующий алгоритм до получения лучших результатов при таком же условно временном лимите на обучение. За временной лимит стоит считать количество итераций оптимизатора. Для начала давайте определим, какая информация из соседних задач нам доступна и какая представляет для нас ценность в решении текущей задачи

2.3.1. Апостериорная информация

На первом этапе для каждого датасета запустим классическую Байесовскую оптимизацию, на выходе которой получим следующую апостериорную информацию:

— X_{opt} – оптимальная конфигурация гиперпараметров
— y_{opt} – оптимальное значение целевой функции
— X – массив рассмотреных конфигураций в порядке итераций
— y – массив значений целевых функций в порядке итераций

— $incubments$ – массив оптимальных конфигураций, известных на момент текущей итерации (значение обновляется только в случае получения лучшего результата)
— $incubment_values$ – массив соответствующих значений целевой функции
— $runtime$ – массив значений времени, затраченного на соответствующую итерацию (включает в себя время обращения к целевой функции и расчёт значений *incubment*)
— $overhead$ – массив значений времени, затраченного на оптимизацию сверх функции

Значения *runtime* и *overhead* могут быть полезны для оценки времени работы модифицированного алгоритма. По значениям *incubment values* можно оценить на каких итерациях мы получаем улучшения, а значения оптимальных конфигураций использовать как полезную нам апостериорную информацию для решаемой задачи.

Рассмотрим наглядный пример, что происходит на каждой итерации. На рисунке 2 представлен пример Байесовской оптимизации на основе гауссовского процесса. В качестве примера взята одномерная функция для упрощения понимания. Точки на графике определяют гиперпараметры, в данном случае гиперпараметр. Оранжевой пунктирной линией показано настоящее значение

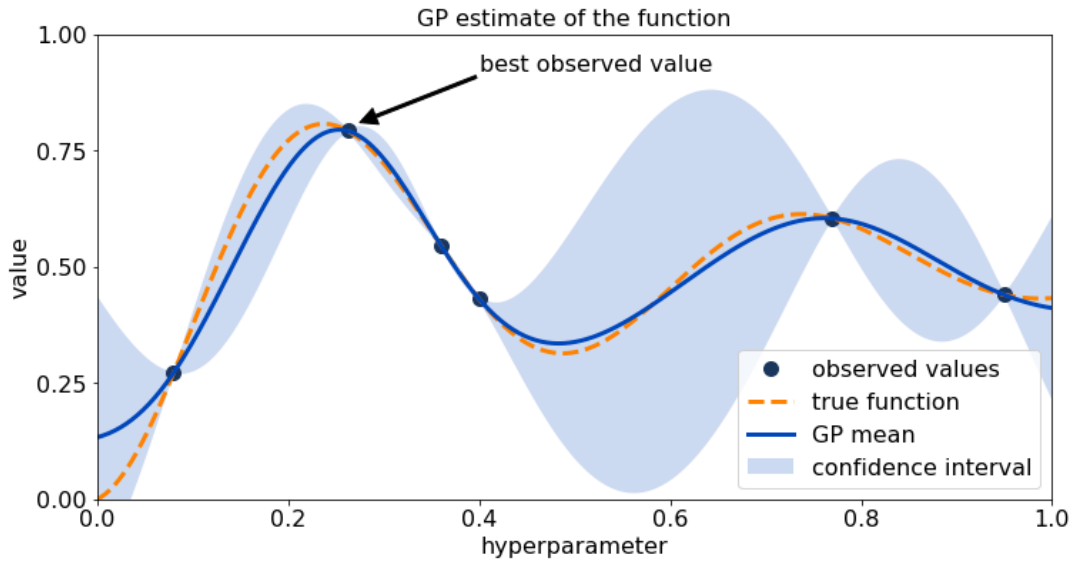

Рисунок 2 – Пример Байесовской оптимизации на основе Гауссовского процесса для одномерной функции [6].

функции «чёрного» ящика, которое на самом деле нам не известно, онако мы пытаемся с течением каждой итерации оптимизации приблизиться к её значе-

нию. Как уже говорилось, на каждой итерации мы получаем значение целевой функции (это и есть наша функция «чёрного ящика») и всё больше приближаемся к действительному значению.

Для старта модели необходимо иметь 3 точки, чтобы понимать, от чего отталкиваться в ходе решения. В общем случае эти точки задаются случайно, как говорилось в главе 1, однако в нашем случае мы знаем результаты обучения на подобных задачах, а также меру подобия задач и можем расставить начальные точки не случайно.

2.3.2. Расстановка начальных конфигураций

Так как для старта модели необходимо 3 точки, имеет смысл рассмотреть 3 ближайшие задачи и получить оптимальный набоор гиперпараметров от каждой. Сам такой подход уже может дать приемущество перед случайной расстановкой, однако этого не достаточно для получения инновационного результата. Кроме того мы обладаем пполной иформацией об обучении подобных задач, что также можно использовать.

Для начала давайте определим понятие *достоверности (reliability)*, которое будет показывать, на сколько можно «доверять» апостериорным знаниям той или иной задачи. Посчитанное в части 2.2 значение расстояния было бы использовать некорректно как минимум по причине того, что оно не нормализовано для модели оптимизации и значения могут значительно именятся с добавлением новых задач и пересчётом метрики. Поэтому введём следующее определение достоверности:

$$R^t = (\alpha^t - K(d_i) + (1 - \alpha^t)K(\zeta_i)) \tag{7}$$

где t – номер итерации оптимизатора, α – мера «забывания» апостериорной информации (от 0 до 1), d_1 – расстояние до задачи *i*, ζ_i – гобальная мера (имеется в виду ζ_{global} для задачи *i*, индекс *global* опущен для упрощения записи), на сколько отличается конфигурация задачи *i* от решаемой, $K(d_i), K(\zeta_i)$ – функции ядра. Для d_i функция ядра – это обыкновенное евклидово расстояние, описанное в части 2.2, а для ζ_i – разница между значениями гиперпараметров для задачи *i* и решаемой.

Так как диапазоны значений для каждого гимерпараметра могут существенно отличатся, то нас интересует не только глобальное значение параметра

ζ, которое и опрределяет подобие конфигураций, но и полная его характеристика для каждой задачи (то есть на сколько отличается каждое из значений гиперпараметров для пары задач). Формально у нас для каждой задачи (относительно решаемой) будет считаться 2 характиристики: гобальное число и локальный массив. Связаны эти значения следующим образом:

$$\zeta_{local} = \{\Delta x_1, \Delta x_2 ... \Delta x_N\}, \zeta_{global} = \text{avg}_i(\Delta x_i) \tag{8}$$

где avg – среднее значение.

Тепрь рассмотрим описанный процесс на примере: предположим, что у нас есть решаемая задача, обозначим её *current* и мы подобрали к ней 2 подобные задачи – d_1 и d_2. На рисунке 3 представлено состояние оптимизатора по

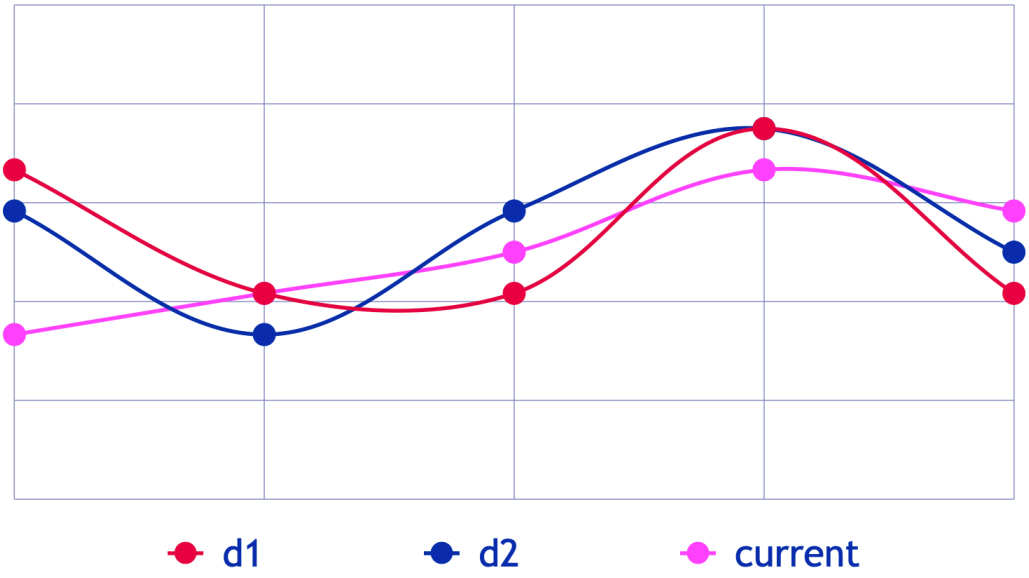

Рисунок 3 – Функция, определяющая гиперпараметры для решаемой задачи и двух подобных.

прохождении 5 итераций (на графике – 5 точек, определяющих конфигурации гиперпараметров, наш пример предполагает модель с одним гиперпараметром для упрощения визуализации). По горизонтали располагается значение гиперпараметра. Кривой представлена аппроксимация апостериорной функции для каждой из задач. Глобально ζ определяет расстояние по вертикали между точками итерации для каждой пары задач, то есть разницу значений функций.

Локальное значение ζ будет проще рассмотреть на примере с бóльшим количеством гиперпараметров. Предположим, мы взяли конкретную итера-

цию для модели с 7 гиперпараметрами (обозначим HP и каждому гиперпараметру присвоим свой цвет для визуализации) и посчитали разницу зачений каждого, на рисунке 4 представлены модули разности значений каждого ги-

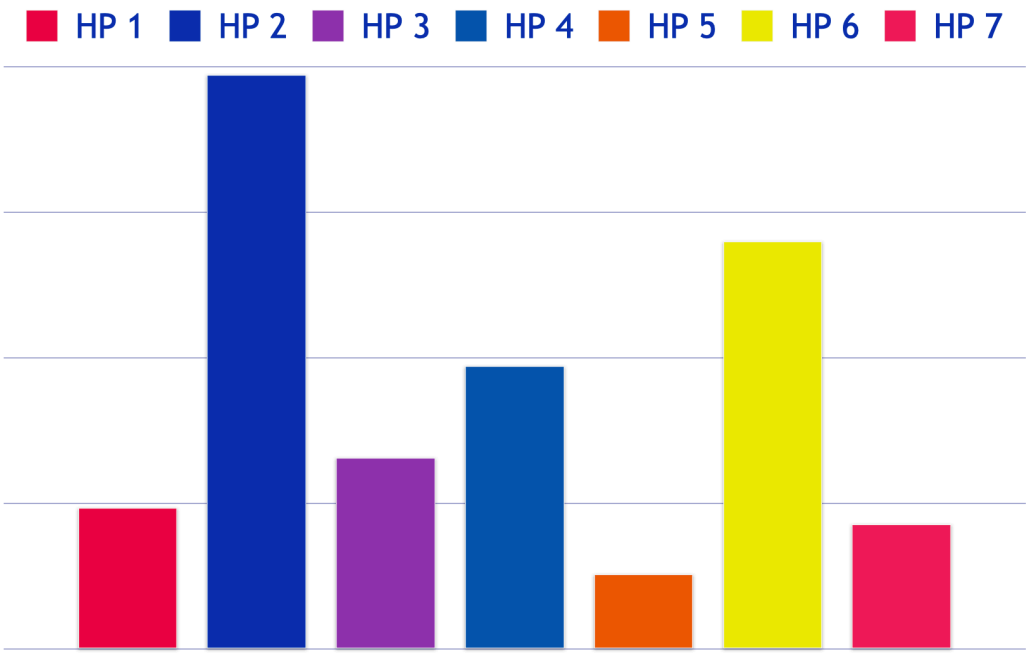

Рисунок 4 – Модуль разницы значений гиперпараметров для решаемой задачи и подобной.

перпараметра для пары задач: решаемой и подобной. Данная диаграма наглядно представляет значения ζ_{local}, из которого по формуле 8 можно получить глобальное значение – ζ_{global} и с его помощью посчитать достоверность 7 подобной задачи, относительно решаемой.

Таким образом, на каждой итерации оптимизатора мы будем получать новое значение достоверности для решаемой задачи и подобной, однако теперь встаёт вопрос: как его применить к решению. Как очевидно из формулы 7, на первой итерации поправка по ζ уйдёт (что логично, так как на первой итерации мы не можем сравнить конфигурацию ренённой задачи и решаемой, так как для решаемой ещё не посчитано ни одно значение целевой функции) и останется только ядро по расстоянию до подобной задачи. Как сописывалось ранее, первые 3 итерации для решаемой задачи грубо посчитаны на значениях лучших конфигураций трёх подобных задач. На каждой следующей итерации уже начинается непосредственно сама оптимизация с использованием функции выгоды. Формально, на каждой итерации мы получаем число ζ_{global} и массив ζ_{local}, из которых считается достоверность и эта достоверность отправ-

ляется в качестве поправки для значения целевой функции. Более подробно: в глобальном цикле Байесовской оптимизации выбирается самая «интересная» на текущий момент точка (конфигурация гиперпараметров) и считается значение целевой функции для этой точки, после чего модель (в нашем случае гауссовский процесс) перетренировывается на новых данных и цикл продолжается. Сам гауссовский процесс тренируется как раз в момент выбора следующего кандидата (следующей наиболее интересной оптимизатору точки), в качестве кандидата возвращается набор координат максимума функции выгоды. Непосредственно поправка, посчитанная относительно подобной задачи, добавляется в момент обучения модели, а именно возводится в квадрат и добавляется по диагонали ковариациоонной матрицы для гауссовского процесса. Таким образом, значение «неопределённости» в конкретных точках будет изменяться пропорционально значению достоверности 7 подобной задачи.

Выводы по главе 2

В данной главе было описано предложенное решение по расширению Байесовской оптимизации. Суть решения заключается в том, что для каждой задачи находятся подобные, между парами рашемая-подобная рассчитывается понятие достоверности. В качестве метрики для определения подобия задач предложено Евклидово расстояние, рассчитанное на нормализованных метапризнаках по всем датасетам. В качестве меры достоверности предложена собственная мера, которая основывается на ядерной функции от расстояни до датасета и разницы по каждому гиперпараметру в корфигурации. В качестве расширения классической Байесовской оптимизации предложено добавлять значение достоверности по диагонали матрицы гауссовского процесса в качестве меры неопределённости в рассматриваемой точке.

ГЛАВА 3. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

3.1. Подробности реализации

Описанный алгоритм реализован на языке `python3` с использованием сторонних пакетов: `numpy`, `pandas`, `george`, за основу для расширенной Байесовской оптимизациии взята реализация из пакета `robo` (не находится в списке общедоступных пакетов, ставится отдельно), графики нарисованы с помощью пакета `matplotlib`.

Расмеченные данные в количестве 76 датасетов взяты с открытого ресурса OpenML [8]. Датасеты выбраны исходя из соображиний максимального разнообразия, так как необходимо подтвердить эффективнось решения на обширном диапазоне задач. В качесте разнообразия имеется в виду различное количество классов (3-1000), количество признаков (3-857), количество самих объектов (61-5000). Полные характеристики для всех выбранных датасетах представлены в приложении А.

3.2. Статистические результаты, дисперсия

Так как и алгоритм оптимизации, и сама модель частично основываются на случайных значениях, то будет некорректно сравнивать ответы по одному запуску. Поэтому полный цикл из 100 итераций Байесовской оптимизации запускался на каждом датасете по 10 раз. Сравнивать мы будем непосредственно средние значения по 10 запускам.

Для того, чтобы оценить статичстические результаты, используем критерий Вилкоксона:

Если оценить результаты, то можно заметить, что для некоторых датасетов разница между худшим и лучшим значениями в ходе оптимизации (иеется в виду значение целевой функции то есть *1 - F-score*) колеблется в пределах нескольких десятых, для других же – в пределах тысячных и то меньше, поэтому можно заключить, что для каждогодатасета при оценке результатов будет учитываться разное количество значимых цифр. Поэтому дисперсия также будет посчитана по 10 запускам, что позволит выяснить количество значащих цифр для сравнения результатов.

3.3. Результаты классической Байесовской оптимизации

Чтобы оценить эффективность предложенного решения, необходимо иметь результаты обучения с использованием существуещего решения, поэтому для начала был запущен алгоритм классической Байесовской оптимизации

на всех датасетах 10 раз. В части 2.3 описаты результаты, которые мы получаем после завершения алгоритма. Очевидно, что для оценки результата мы будем использовать *incumbent values*, то есть улучшение значения целевой функции на каждой итерации.

В качестве примера рассмотрим результат одного конкретного запуска на одном датасете. На рисунке 5 предсавлен непосредственно пример результа-

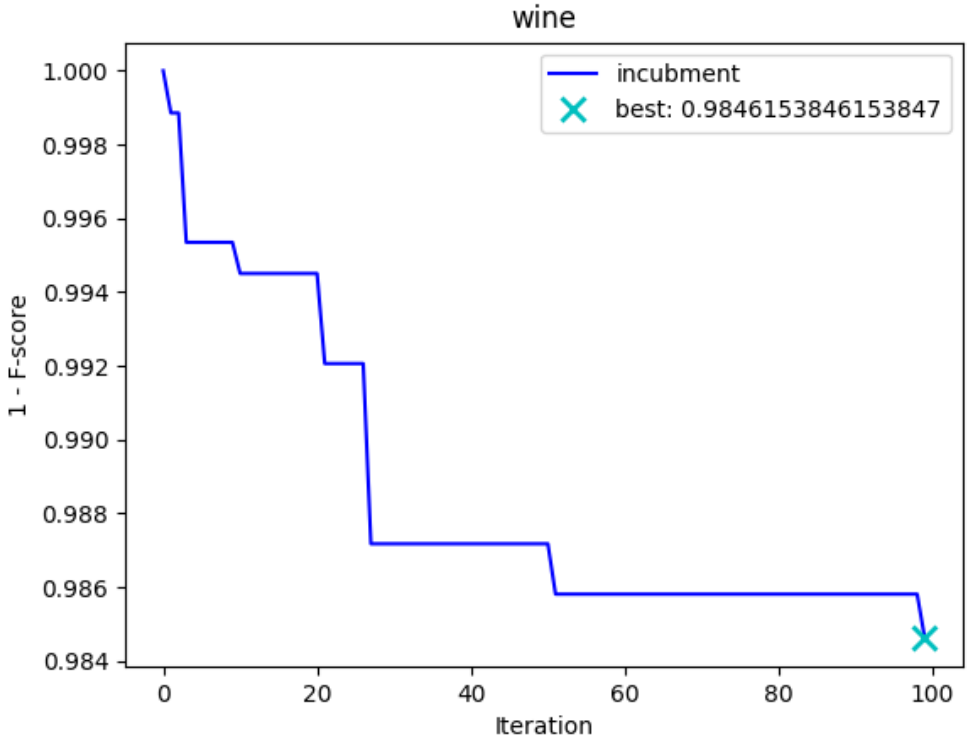

Рисунок 5 – Результат работы классической Байесовской оптимизации на наборе данных *wine*.

та классической Байесовской оптимизации после 100 итерации. По вертикали располагаются лучшие на данный момент значения целевой функции. Вспомним, что в основе нашей целевой функции лежит *F-score*, а так как в Байесовской оптимизации идёт процесс минимизации, то мы рассматриваем обратное значение, то есть *(1 - F-score)*, поэтому значение на графике убывает. Также вспомним, что *incumbent values* описывает **лучшее на данный момент времени** (на данной итерации) значение целевой функции, то есть, на итерации №40 нельзя утверждать, что значение целевой функции совпадает со значением на итерации №39, наоборот, вероятнее всего значение получилось хуже и просто не включилось в массив ответов (но учлось при дальнейшей оптимизации, так как каждая итерация добавляет определённости апостериорной

информации). Лучшее значение целевой функции обозначено на графике как *best*, после этой отметки и до последней итерации значение не улучшалось.

На том же примере можно заметить, что разница между худшим и лучшим значениями заключена в пределах 0,02, когда, например, на рисунке 6 это

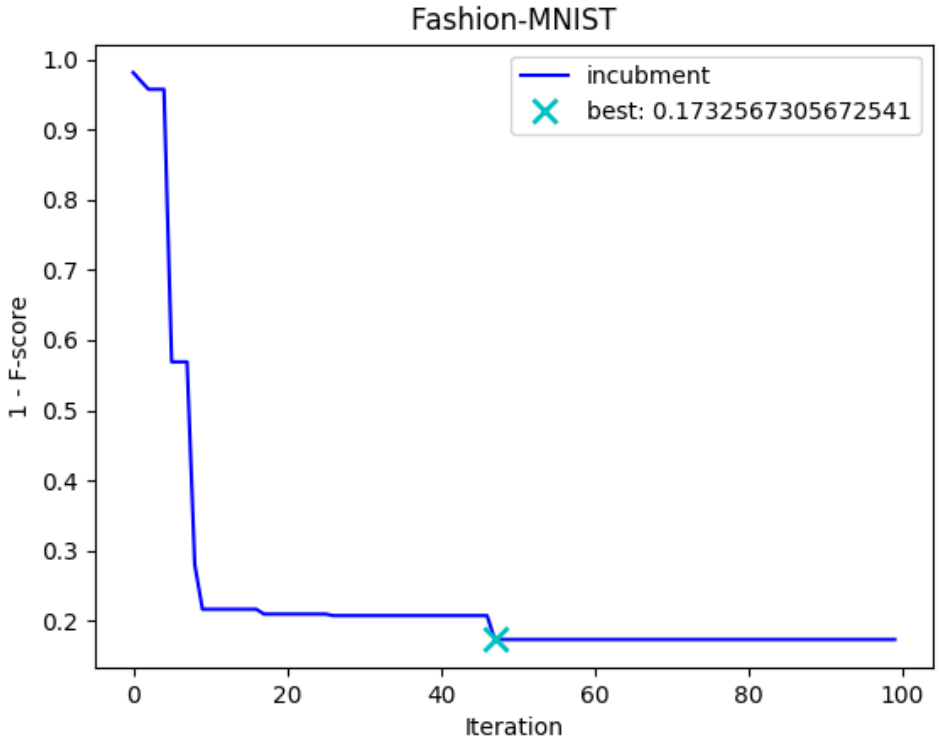

Рисунок 6 – Результат работы классической Байесовской оптимизации на наборе данных *Fashion-MNIST*.

значение заключено в диапазоне 0,8. Это наглядно показывает, что для сравнения результатов нельзя использовать фиксированное количество значащих цифр, поэтому для каждого датасета это количество нужно определять индивидуально на основе статистичиских результатов.

3.4. Сравнение результатов классической и расширенной Байесовской оптимизации

После того, как получены результаты классической Байесовской оптимизации, можно сравнивать их с результатами нашего модифицированного алгоритма. Но возникает вопрос: что и как сравнивать? Можно предположить, что после обучения нам интересно только лучшее полученное значение целевой функции, однако если этот оптимальный результат получен на более ранней итерации, то это тоже считается успехом. Для того, чтобы определить кри-

терии сравнения результатов, выделим все возможные сучаи по завершении последней итерации:

а) получено лучшее оптимальное значение целевой функции на более ранней итерации – абсолютная победа предложенного решения
б) получено лучшее оптимальное значение целевой функции на той же итерации – победа предложенного решения по абсолютному значению, ничья по номеру итерации
в) получено лучшее оптимальное значение целевой функции, но на более позней итерации – победа предложенного решения по абсолютному значению, проигрыш по номеру итерации
г) оптимальные значения одинаковы (в пределах значащих цифр), но в предложенном решении это значение получено на более ранней итерации – ничья по абсолютному значению, победа предложенного решения по номеру итерации
д) оптимальное значение хуже, но получено на более ранней итерации – проигрыш по абсолютному значению, победа по номеру итерации

Для случаев а, б, г победа предложенного решения очевидна, однако в случае в несправедливо утверждать, что мы проиграли по номеру итерации, так как возможны (и весьма вероятны) случаи, когда на момент нахождения отптимального значения классической Байесовской оптимизацией, абсолютное значение в нашем решении уже было лучше, это говорит о том, что оптимальное значение классической оптимизации нашим решением было получено раньше. В случае д же вовсе нельзя утверждать о победе нашего решения, так как всё-таки в первую очередь нас интересует абсолютное значение.

Кроме того, так как и сама оптимизация, и модель основываются на случайных процессах, кроме вышеописанного, нас также интеренует статистическая значимость результатов для каждого датасета. Например, если худшее значение целевой функции было 0,8, а к концу оптимизации стало 0,101 для классического решения и 0,105 для нашего, то улучшение не так существенно, как было бы в случае оптимальных результатов 0,3 и 0,1 соответственно. Поэтому используем формальную оценку качества оптимизации.

3.4.1. Оценка качества оптимизации

Как было отмечено ранее, в первую очередь нас интересует само оптимальное значение целевой функции, поэтому из вышеперечисленных случаев

можно выделить 2 группы: значения целевых функций различны и значения функций одинаковы.

Значения целевых функций различны. В этом случае применим критерий Вилкоксона, описанный в части 3.2, к абсолютному значению целевой функции по всем запускам и получим значимость результата.

Значения целевых функций одинаковы. В этом случае применим критерий Вилкоксона к номеру итерации по всем запускам и получим значимость результата.

Исходя из полученной значимости допустимо рассуждать о победе или поражении того или иного алгоритма оптимизации. Теперь можно переходить непосредственно к оценке результатов.

3.4.2. Оценка результатов

Для каждого датасета были посчитаны средние для оптимального значения и номера итерации. Также, для каждого датасета было рассчитано количество значащих цифр (определяется средним значением дисперсии по всем запускам для датасета).

Полный список результатов классической Байесовской оптимизации представлен в приложении Б. Полный список результатов расширенной Байесовской оптимизации представлен в приложении В.

Теперь сравним результаты классической и расширенной Байесовской оптимизации для датасета из части 3.3. На рисунке 7 розовой линией представлен результат работы расширенной Байесовской оптимизации, то есть предложенного алгоритма. Можно отметить, что оптимальный результат получен значительно раньше и по абсолютному значению также побеждает классическую Байесовскую оптимизацию. На наборе данных *car* (рисунок 8) диапазон значений целевой функции гораздо шире, однако результат также побеждает как по абсолютному значению, так и по номеру итерации его достижения. На данном примере можно наглядно показать влияние начальной расстановки и последующей корректировки гиперпараметров. Мы видим, что начальное значение целевой функции для нашего алгоритма располагается значительно ниже результата крассического алгоритма, это говорит о влиянии начальной

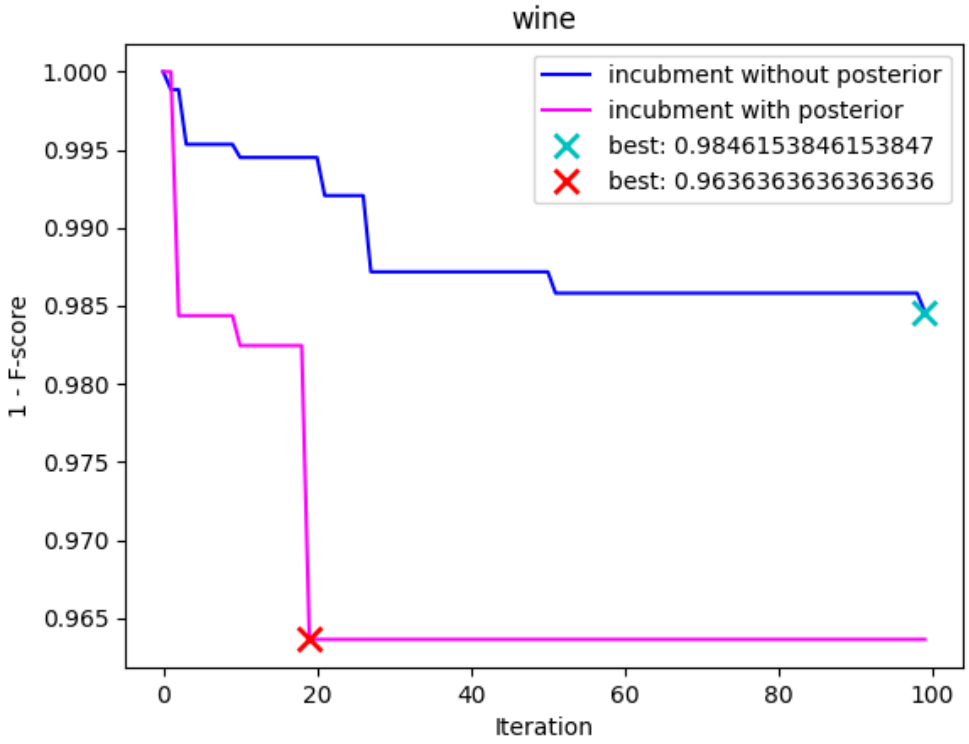

Рисунок 7 – Результат работы расширенной Байесовской оптимизации на наборе данных *wine*.

расстановки из подобных задач. Дальнейшее улучшение уже является результатом корректировки на основе достоверности.

Однако не на всех датасетах получены подобные результаты, например, для датасета *desharnais* (рисунок 9) начальная расстановка сыграла отрицательною роль, а непосредственно корректировка по достоверности позволила получить оптимальный результат.

Выводы по главе 3

По результатам видно, что предложенный алгоритм работает эффективнее классической Байесовской оптимизации.

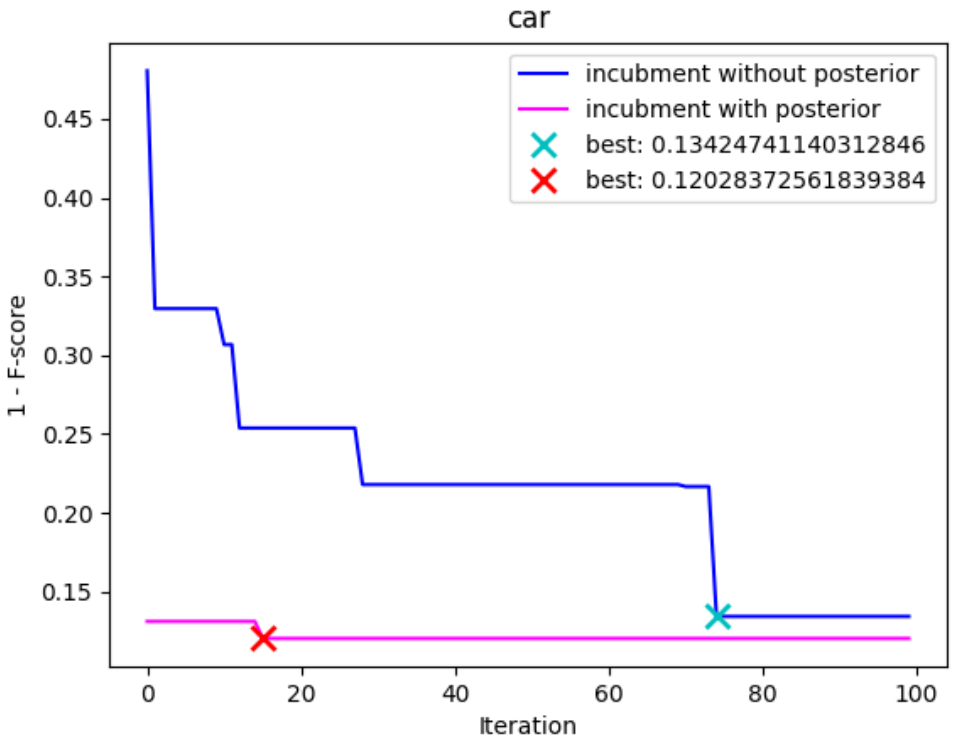

Рисунок 8 – Результат работы расширенной Байесовской оптимизации на наборе данных *car*.

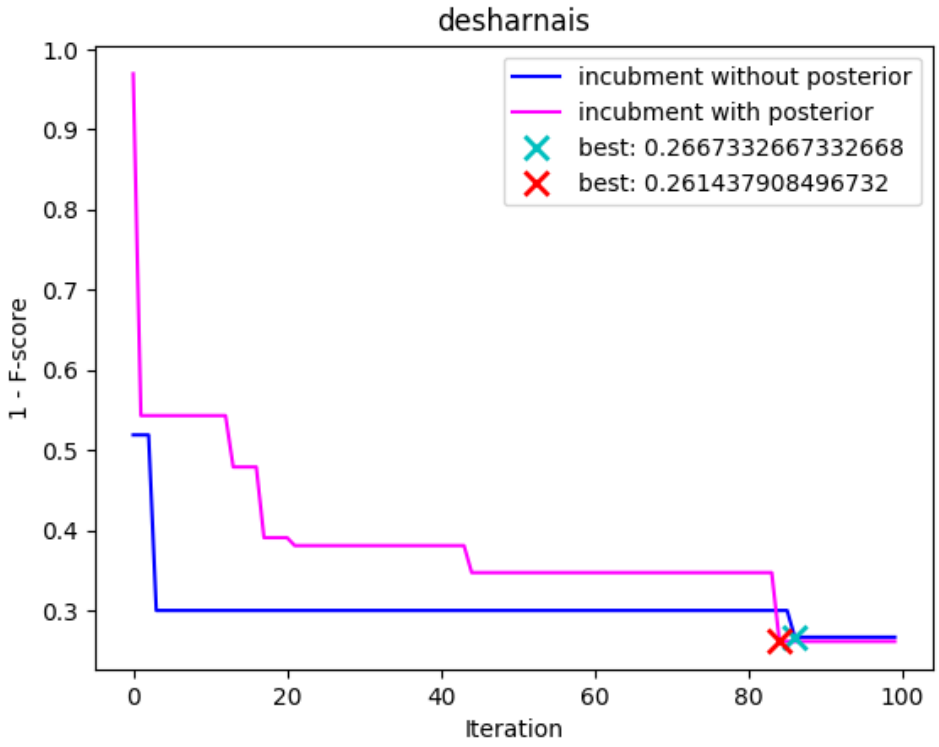

Рисунок 9 – Результат работы расширенной Байесовской оптимизации на наборе данных *desharnais*.

ЗАКЛЮЧЕНИЕ

В ходе исследования были рассмотрены существующие решения для проблемы оптимизации гиперпараметров и предложено новое эффективное решение, которое показало свою эффективность на статистических экспериментах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Algorithms for Hyper-Parameter Optimization / J. S. Bergstra [et al.] // Advances in Neural Information Processing Systems 24 / ed. by J. Shawe-Taylor [et al.]. — Curran Associates, Inc., 2011. — P. 2546–2554. — URL: http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

2 Efficient and Robust Automated Machine Learning / M. Feurer [et al.] // Advances in Neural Information Processing Systems 28 / ed. by C. Cortes [et al.]. — Curran Associates, Inc., 2015. — P. 2962–2970. — URL: http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf.

3 *Falkner S.*, *Klein A.*, *Hutter F.* BOHB: Robust and Efficient Hyperparameter Optimization at Scale // Proceedings of the 35th International Conference on Machine Learning (ICML 2018). — 07/2018. — P. 1436–1445.

4 *Feurer M.*, *Hutter F.* Hyperparameter Optimization // AutoML: Methods, Sytems, Challenges / ed. by F. Hutter, L. Kotthoff, J. Vanschoren. — Springer, 05/2019. — Chap. 1. P. 3–33.

5 *Hutter F.*, *Hoos H. H.*, *Leyton-Brown K.* Sequential Model-Based Optimization for General Algorithm Configuration (extended version) : tech. rep. / University of British Columbia, Department of Computer Science. — 2010. — TR-2010–10. — Available online: http://www.cs.ubc.ca/~hutter/papers/10-TR-SMAC.pdf.

6 *Kiss O.* Bayesian Optimization for machine learning algorithms in the context of Higgs searches at the CMS experiment. — 2019. — arXiv: 1911.02501 [physics.data-an].

7 *Lindauer M.*, *Hutter F.* Warmstarting of Model-based Algorithm Configuration. — 2017. — arXiv: 1709.04636 [cs.AI].

8 OpenML: Networked Science in Machine Learning / J. Vanschoren [et al.] // SIGKDD Explorations. — New York, NY, USA, 2013. — Vol. 15, no. 2. — P. 49–60. — DOI: 10.1145/2641190.2641198. — URL: http://doi.acm.org/10.1145/2641190.2641198.

9 RoBO: A Flexible and Robust Bayesian Optimization Framework in Python / A. Klein [et al.] // NIPS 2017 Bayesian Optimization Workshop. — 12/2017.

ПРИЛОЖЕНИЕ А. ИНФОРМАЦИЯ ОБ ОТОБРАННЫХ ДАТАСЕТАХ

Имя датасета	**Число объектов**	**Число признаков**	**Число классов**	**ID на OpenML**
abalone	4177	9	3	1557
artificial-characters	10218	8	10	1459
balance-scale	625	5	3	11
breast-tissue	106	10	6	1465
car	1728	7	4	40975
cardiotocography	2126	36	10	1466
cmc	1473	10	3	23
cnae-9	1080	857	9	1468
collins	1000	24	30	40971
covertype	581012	55	7	150
desharnais	81	13	3	1117
diggle-table-a2	310	9	9	694
ecoli	336	8	8	39
energy-efficiency	768	10	37	1472
eye-movements	10936	28	3	1044
fabert	8237	801	7	41164
fars	100968	30	8	40672
Fashion-MNIST	70000	785	10	40996
gas-drift	13910	129	6	1476
gas-drift-different-concentrations	13910	130	6	1477
gina-prior2	3468	785	10	1041
glass	214	10	6	41
har	10299	562	6	1478
hayes-roth	160	5	3	329
heart-long-beach	200	14	5	1512
heart-switzerland	123	13	5	1513
helena	65196	28	100	41169
Indian-pines	9144	221	8	41972
iris	150	5	3	61

jannis	83733	55	4	41168
JapaneseVowels	9961	15	9	375
jungle-chess-2pcs-endgame-panther-elephant	4704	47	3	41000
jungle-chess-2pcs-raw-endgame-complete	44819	7	3	41027
leaf	340	16	30	1482
LED-display-domain-7digit	500	8	10	40496
mfeat-factors	2000	217	10	12
mfeat-fourier	2000	77	10	14
mfeat-karhunen	2000	65	10	16
mfeat-morphological	2000	7	10	18
mfeat-pixel	2000	241	10	40979
microaggregation2	20000	21	5	41671
nursery	12960	9	5	26
page-blocks	5473	11	5	30
pokerhand	829201	11	10	155
PopularKids	478	11	3	1100
prnn-fglass	214	10	6	952
prnn-viruses	61	19	4	480
rmftsa-sleepdata	1024	3	4	679
robot-failures-lp1	88	91	4	1516
robot-failures-lp5	164	91	5	1520
satimage	6430	37	6	182
seeds	210	8	3	1499
segment	2310	20	7	40984
seismic-bumps	210	8	3	1500
semeion	1593	257	10	1501
shuttle	58000	10	7	40685
spectrometer	531	103	48	313
steel-plates-fault	1941	28	7	40982

synthetic-control	600	62	6	377
tae	151	6	3	48
tamilnadu-electricity	45781	4	20	40985
teachingAssistant	151	7	3	1115
thyroid-allbp	2800	27	5	40474
thyroid-allhyper	2800	27	5	40475
user-knowledge	403	6	5	1508
vehicle	846	19	4	54
vertebra-column	310	7	3	1523
volcanoes-a1	3252	4	5	1527
volcanoes-a3	1521	4	5	1529
volcanoes-a4	1515	4	5	1530
volcanoes-d1	8753	4	5	1538
wall-robot-navigation	5456	5	4	1526
waveform-5000	5000	41	3	60
wine	178	14	3	187
wine-quality-white	4898	12	7	40498
zoo	101	17	7	62

ПРИЛОЖЕНИЕ Б. РЕЗУЛЬТАТЫ КЛАССИЧЕСКОЙ БАЙЕСОВСКОЙ ОПТИМИЗАЦИИ

Имя датасета	Лучшее значение (1 - F-score)	Номер итерации
page-blocks	0.13516414692083642	84
robot-failures-lp1	0.08618397125750067	82
mfeat-fourier	0.16586653632657422	97
jungle-chess-2pcs-raw-endgame-complete	0.2831540971429782	99
heart-switzerland	0.6538168757168756	98
gas-drift-different-concentrations	0.014135671941996442	92
wall-robot-navigation	0.02080579413576409	83
jungle-chess-2pcs-endgame-panther-elephant	0.0004813765790142255	88
leaf	0.40293986047115576	95
PopularKids	0.0755458190128694	85
mfeat-karhunen	0.02322287779210741	99
diggle-table-a2	0.9795551003350139	97
rmftsa-sleepdata	0.9080866235490108	95
semeion	0.06514854990131118	87
desharnais	0.094592058505102	86
teachingAssistant	0.4417724517376909	98
collins	0.8756704066611304	97
volcanoes-a3	0.7748012053245856	97
artificial-characters	0.3559097202747393	97
volcanoes-a4	0.7954900451834821	99
glass	0.5438046693293679	85
nursery	0.06674407462192931	81
shuttle	0.02125073667637095	98
segment	0.018440943963987422	97
heart-long-beach	0.6881087163579751	91
vertebra-column	0.1430715573316498	98

cnae-9	0.04608401806475787	98
jannis	0.9999873749141746	87
wine-quality-white	0.7684229624815606	97
vehicle	0.34810891893008283	96
ecoli	0.13981443473353988	95
eye-movements	0.5454590857253053	94
seeds	0.057802300267854614	96
car	0.16487930442466345	85
fabert	0.8563974922818955	13
breast-tissue	0.6818280793280793	93
thyroid-allbp	0.5500200435481564	99
gas-drift	0.01229587283888477	99
mfeat-factors	0.06790745344168528	94
volcanoes-d1	0.7979166787265425	96
har	0.01990235753262858	96
satimage	0.11962597103528783	86
Fashion-MNIST	0.1841999445039933	70
seismic-bumps	0.02381313808909159	66
pokerhand	0.30047170829842273	93
helena	0.9781497717820775	98
thyroid-allhyper	0.5189136491165705	98
wine	0.974506165505316	99
balance-scale	0.053141906070482484	96
microaggregation2	0.5716839504480571	88
steel-plates-fault	0.7624399867662701	87
tae	0.45732035806702226	86
mfeat-pixel	0.558084964753526	95
gina-prior2	0.07109914430206611	97
synthetic-control	0.0	14
cmc	0.4518832129599252	92
energy-efficiency	0.8229788731752743	92
iris	0.0	5
fars	0.4919820642867753	87

abalone	0.615400258824198	95
prnn-viruses	0.03733333333333333	93
Indian-pines	0.4921257947810759	96
covertype	0.44623970623986625	89
JapaneseVowels	0.5106516915536152	96
user-knowledge	0.12889846264159402	95
spectrometer	0.9661251360456881	92
hayes-roth	0.1456313707054801	97
robot-failures-lp5	0.2881688745907933	91
prnn-fglass	0.39034465586128364	84
waveform-5000	0.13219632609315485	89
zoo	0.0	48
cardiotocography	0.12124778732718564	96
mfeat-morphological	0.5584251769846496	98
volcanoes-a1	0.7917649985130863	98
tamilnadu-electricity	0.04787838195460763	98
LED-display-domain-7digit	0.22008194071239123	79

ПРИЛОЖЕНИЕ В. РЕЗУЛЬТАТЫ РАСШИРЕННОЙ БАЙЕСОВСКОЙ ОПТИМИЗАЦИИ

Имя датасета	**Лучшее значение (1 - F-score) (КБО)**	**Лучшее значение (1 - F-score) (ПБО)**	**Номер ите-рации (КБО)**	**Номер ите-рации (ПБО)**
car	0.13424741140312846	0.12028372561839384	74	15
wine	0.9846153846153847	0.9636363636363636	99	19
cmc	0.4217043857598791	0.4608957535597932	12	89
zoo	0.0	0.0	12	11
nursery	0.04052981019439761	0.05928043944850669	61	63
abalone	0.6296727096012757	0.629497094309903	95	12
cardiotocography	0.06492996280208396	0.06644365589389312	96	57
desharnais	0.2667332667332668	0.261437908496732	86	84
glass	0.5231009070294785	0.45202303143479605	68	89
segment	0.02531199490574576	0.03319329454434661	62	22