МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Гуртовая Ольга Владимировна

Методы онлайн оптимизации квадратичной функции потерь, основанные на использовании случайных признаков Фурье

Специальность 1.2.2. — «Математическое моделирование, численные методы и комплексы программ»

Диссертация

на соискание учёной степени кандидата

физико-математических наук

Научный руководитель:

доктор физико-математических наук,

доцент Рохлин Д. Б.

Ростов-на-Дону — 2025

Содержание

Введение

Актуальность темы. Современные задачи в таких областях, как обработка финансовых временных рядов [1], анализ потоковых данных в интернете вещей [32], адаптивные системы рекомендаций [56], робототехника и оптимальное управление [92], требуют построения математических моделей, способных адаптироваться к изменяющимся условиям в реальном времени. Ключевая сложность заключается в том, что часто бывает неизвестна сама природа данных, а также конкретные механизмы и факторы, вызывающие их изменение, которые могут иметь состязательный (adversarial) характер. В связи с этим актуальна задача математического моделирования и анализа таких процессов. В этом контексте онлайн оптимизация выделяется как одно из наиболее перспективных направлений. В отличие от классической (офлайн) оптимизации, где всё множество данных доступно заранее, онлайн алгоритмы обрабатывают информацию последовательно, постоянно адаптируя свои решения на основе как новых данных, так и предыдущего опыта. Это позволяет моделям динамически перестраиваться и поддерживать высокую эффективность работы без необходимости полного переобучения на всех накопленных данных.

Основным инструментом, исследуемым в диссертации, является метод случайных признаков Фурье, первоначально предложенный в работах [68, 69]. Исторически этот метод возник как вычислительно эффективная альтернатива ядерным методам, позволяющая преодолеть их высокую вычислительную сложность. Ранние подходы в онлайн оптимизации с ядрами были сосредоточены на стратегиях снижения сложности за счёт ограничения числа опорных векторов, как, например, в алгоритмах Projectron [62] и Forgetron [25].

Метод случайных признаков Фурье предлагает принципиально иное решение: вместо работы в бесконечномерном пространстве, порождаемом ядром, он строит явное низкоразмерное отображение данных в пространство признаков, где модель становится линейной относительно своих коэффициентов. Аналогично преобразованию Фурье, ядро (мера схожести признаков) аппроксимируется с помощью набора простых тригонометрических функций. Сначала генерируется набор случайных векторов-частот ω_i из распределения, связанного

с ядром преобразованием Фурье, и случайных сдвигов фазы b_i, равномерно распределённых на отрезке $[0, 2\pi]$. Затем для каждой входной точки данных (вектора признаков) x вычисляется её проекция на случайные направления ω_i, к которой прибавляются сдвиги b_i. Новыми признаками будут $\cos(\langle \omega_i, x \rangle + b_i)$.

Теоретические оценки аппроксимации, полученные в работах [68, 69] и последующее развитие этих оценок, например в работе [75], создало прочный теоретический фундамент для практического применения метода. Данный подход можно рассматривать как построение моделей типа «чёрного ящика», способных аппроксимировать широкий класс зависимостей без необходимости точного знания внутренней структуры системы.

Разработанные в работе численные методы онлайн оптимизации, основанные на этой идее, демонстрируют свою универсальность для построения математических моделей в самых разных предметных областях. Процесс моделирования можно описать следующим образом: данные поступают последовательно (онлайн), на каждом шаге по имеющимся признакам $x_t \in \mathbb{R}^d$ (входные параметры системы) строится их образ в пространстве случайных признаков Фурье $\Phi(x_t) \in \mathbb{R}^m$, и на его основе строится прогноз целевой переменной $\hat{y}_t = \langle w_t, \Phi(x_t) \rangle$, где $w_t \in \mathbb{R}^m$ — вектор весов модели. Затем становится известно истинное значение целевой переменной $y_t \in \mathbb{R}$, и вычисляется квадратичная функция потерь

$$l_t(w_t) = (\hat{y}_t - y_t)^2 = (\langle w_t, \Phi(x_t) \rangle - y_t)^2,$$

характеризующая точность модели. Задача состоит в построении такой последовательности векторов весов w_t, которая минимизирует суммарные потери за всё время работы. Эффективность алгоритма оценивается величиной сожаления (regret)

$$R_T(u) = \sum_{t=1}^{T} l_t(w_t) - \sum_{t=1}^{T} l_t(u),$$

представляющего собой разность между накопленными потерями нашего алгоритма и потерями «эксперта» u. Этот «эксперт» может выбирать свою стра-

тегию ретроспективно, обладая полным знанием всех данных. Таким образом, $R_T(u)$ измеряет, насколько мы «сожалеем», что не следовали стратегии u.

Проиллюстрируем широту применения данного подхода в разных задачах математического моделирования, рассмотренных в диссертации.

Моделирование стационарных марковских процессов (глава 1), которые являются фундаментальным инструментом для описания систем, развивающихся во времени. Задача состояла в том, чтобы, наблюдая только текущее состояние системы (входные признаки), построить прогноз не просто следующего значения, а некоторой сложной функции от её будущего состояния через несколько шагов (целевая переменная). В качестве конкретного примера для проверки теоретических выводов была использована модель векторной авторегрессии (VAR). Такая модель позволяет описывать взаимосвязанную динамику нескольких временных рядов (например, курсов валют и цен на акции), где поведение каждого ряда в настоящем зависит от его собственных прошлых значений и прошлых значений других рядов.

Моделирование и анализ сложных физических, экономических и социальных явлений на основе имеющихся табличных данных (глава 2). В частности:

- Моделирование аэроакустического шума, генерируемого при обтекании профиля крыла. Цель: установить нелинейную зависимость между набором входных признаков (частота, угол атаки, толщина профиля и др.) и целевой переменной — уровнем звукового давления.

- Моделирование рынка недвижимости. Цель: установить сложную, многофакторную зависимость между входными признаками, описывающими объект и его местоположение (площадь, количество комнат, характеристики района и т.д.), и целевой переменной — стоимостью дома.

- Моделирование социально-экономического поведения на примере данных об оставлении чаевых. Цель: описать зависимость между набором входных факторов (общая сумма счёта, день недели, пол плательщика и др.) и целевой переменной — размером оставленных чаевых.

Как будет показано в третьей главе диссертации, разработанные для таких задач численные методы онлайн оптимизации обладают достаточной общностью, позволяющей применять их к принципиально иному классу проблем. Это демонстрируется на примере применения методов онлайн оптимизации для решения задачи моделирования внешнего воздействия в рамках задачи вариационного исчисления с квадратичным функционалом качества. Современные подходы к решению таких задач в онлайн режиме подробно рассматриваются в обзорных работах [51], где особое внимание уделялось вопросам применения оптимального управления к динамическим системам. Применение методов онлайн оптимизации к вариационным задачам открывает новые возможности в таких областях, как адаптивное управление динамическими системами [24], обработка нестационарных сигналов [87] и оптимальное планирование в условиях неопределённости [5].

Предложенный метод решения вариационных задач с неизвестным внешним воздействием сочетает технику тригонометрической аппроксимации с методами онлайн оптимизации, что позволяет эффективно решать задачи оптимального управления в условиях неполной информации. Данный подход развивает идею аппроксимации в пространстве Фурье, но, в отличие от стохастического подхода на основе случайных признаков, использует классическую тригонометрическую аппроксимацию в задачах поиска оптимальной траектории. Теоретическая значимость подтверждается полученными оценками ошибок аппроксимации и границами сожаления, а практическая ценность — возможностью применения в системах реального времени, требующих быстрой адаптации к изменяющимся внешним условиям.

Теория выпуклой онлайн оптимизации, составляющая основу данного исследования, имеет глубокие теоретические корни и широкую практическую значимость. Историческое развитие этого направления восходит к фундаментальным работам [39] и [11] по теории стратегий минимизации сожаления, заложившим основы для последующего формализма предсказания с экспертами. Значительный прорыв был достигнут в работах [55] и [97], где были разработаны первые практически применимые алгоритмы агрегирования прогнозов.

Современный этап развития теории онлайн оптимизации опирается на ре-

зультаты, изложенные в классических монографиях [17, 83]. Дальнейшее развитие и систематизация этого направления представлены в современных трудах [42, 63]. Важными достижениями в этой области стали разработка адаптивных алгоритмов оптимизации, таких как AdaGrad [28], и исследование методов ядерного последовательного обучения [48].

Теоретические основы для анализа алгоритмов онлайн оптимизации были заложены в работе [2], где были получены первые строгие оценки обобщающей способности онлайн алгоритмов для зависимых данных, что позволило перенести многие результаты из классического н.о.р. случая на более сложные сценарии. Особый интерес представляют задачи регрессионного анализа в условиях марковской зависимости данных, для решения которых в работах [3] и [103] были разработаны специализированные алгоритмы.

Также не менее важны разработки в области многоядерного построения моделей [37], которые предоставляют мощные инструменты для комбинирования информации из различных источников, что позволяет создавать более гибкие и точные модели. В контексте возрастающей сложности моделей и данных актуальным становится и автоматизированное построение моделей (AutoML), которое стремится автоматизировать процесс выбора моделей, настройки гиперпараметров и создания признаков. Эти направления — многоядерное обучение и AutoML — отражают две ключевые тенденции современного машинного обучения: необходимость эффективной интеграции разнородных данных и автоматизации сложных этапов анализа.

Цели работы. Целью диссертационной работы является разработка математических моделей и численных методов онлайн оптимизации, основанных на использовании случайных признаков Фурье для сведения задач непараметрической регрессии к конечномерному случаю. В качестве варианта данного подхода исследуется его детерминированный аналог — аппроксимация тригонометрическими полиномами — для решения задач вариационного исчисления. Ключевой задачей для всех рассматриваемых методов является вывод оценок сожаления.

Для достижения указанной цели необходимо решить следующие задачи.

1. Разработать методику применения численного метода Вовка–Азури–Вармута (VAW) к случайным признакам Фурье для моделирования данных с марковской зависимостью с теоретическим обоснованием и экспериментальной проверкой на моделях AR(1) и VAR(1).

2. Разработать новый численный метод VAW^2, основанный на многоядерном подходе и построении ансамбля моделей, для моделирования различных регрессионных зависимостей. Установить для него оценку сожаления. Провести сравнительный анализ предложенного численного метода с известными из литературы родственными алгоритмами на ряде эталонных наборов данных.

3. Разработать трёхуровневый численный метод S-VAW^2, объединяющий иерархическое агрегирование и стратегии масштабирования данных. Провести сравнительный анализ с ведущим AutoML-фреймворком.

4. Сформулировать и решить задачу моделирования внешнего воздействия динамической системы с квадратичным функционалом качества, включая её сведение к конечномерной задаче и вывод границ статического и динамического сожалений.

5. Реализовать комплекс программ на Python для численного моделирования и сравнительного анализа предложенных онлайн-алгоритмов.

Объект исследования — алгоритмы онлайн оптимизации в гильбертовых пространствах для задач, где данные поступают последовательно и могут иметь как стохастическую, так и состязательную (adversarial) природу.

Предмет исследования — методология сведения бесконечномерных задач онлайн оптимизации к конечномерным и анализ качества соответствующих алгоритмов. В частности, исследуются:

- применение случайных признаков Фурье для аппроксимации ядерных методов в задачах онлайн регрессии;

- применение аппроксимации тригонометрическими полиномами в задачах вариационного исчисления с неизвестным внешним воздействием;

- теоретические оценки границ сожаления как ключевая метрика качества разработанных подходов.

Методы исследования. Работа опирается на методы функционального анализа (в частности, теорию гильбертовых пространств с воспроизводящим ядром), теории вероятностей (включая теорию марковских цепей), методы выпуклого анализа (включая субградиентные методы онлайн оптимизации).

Научная новизна исследования. Результаты, выносимые на защиту, являются новыми. Автором совместно с научным руководителем проводилась постановка задач, обсуждались полученные основные результаты и формулировки выводов.

Теоретическое значение исследования заключается в развитии теории онлайн оптимизации для решения задач в гильбертовых пространствах. В работе установлены новые теоретические оценки сожаления для предложенного класса многоядерных алгоритмов, а также для алгоритмов, решающих задачи непараметрической регрессии с марковскими данными и вариационного исчисления с неизвестным внешним воздействием.

Практическая значимость исследования определяется разработкой и экспериментальной апробацией эффективных численных методов VAW^2 и S-VAW^2, работающих в онлайн режиме. Данные алгоритмы реализуют гибкий многоядерный подход к онлайн регрессии на основе вычислительно эффективной техники случайных признаков Фурье. Проведённые эксперименты на широком наборе реальных данных показали, что предложенный подход демонстрирует результаты, сопоставимые с ведущим AutoML-фреймворком AutoGluon-Tabular в задачах регрессии. Все разработанные алгоритмы были реализованы в виде программного модуля на языке Python, что подтверждает их практическую применимость и создаёт основу для их дальнейшего использования в прикладных системах.

Основные положения, выносимые на защиту.
В области математического моделирования:

1. Разработан подход к моделированию условных математических ожиданий для стационарных марковских процессов по одной наблюдаемой тра-

ектории, основанный на аппроксимации случайными признаками Фурье.

2. Предложен метод математического моделирования нелинейных регрессионных зависимостей, когда данные обладают состязательным свойством, на основе иерархических ансамблей моделей со случайными признаками Фурье.

3. Сформулирована и исследована задача математического моделирования внешнего воздействия неизвестной природы в вариационных задачах с квадратичным функционалом качества в режиме онлайн.

В области численных методов:

4. Разработан численный метод для решения задач онлайн регрессии на марковских данных, основанный на комбинации алгоритма Вовка–Азури–Вармута (VAW) со случайными признаками Фурье. Для данного метода получены теоретические оценки сожаления для квадратичной функции потерь, которые обосновывают его эффективность в условиях зависимых данных. Экспериментально подтверждена конкурентоспособность предложенного подхода на моделях авторегрессии первого порядка и векторной авторегрессии.

5. Предложен новый двухуровневый численный метод многоядерного обучения VAW^2, обладающий значительно более низкой вычислительной сложностью (порядка $O(Nm^2)$ на итерацию) по сравнению с прямым применением алгоритма VAW к конкатенированным векторам признаков ($O(N^2m^2)$). Для метода VAW^2 установлена оценка сожаления порядка $O(T^{1/2}\ln T)$, что подтверждает его теоретическую эффективность. Здесь N — число ядер, m — количество случайных признаков для каждого ядра; T — длина выборки. Вычислительные эксперименты показали, что алгоритм VAW^2 превосходит в качестве предсказаний известные из литературы родственные численные методы на ряде наборов данных.

6. Разработан трёхуровневый численный метод (S-VAW^2), интегрирующий иерархическое агрегирование и различные стратегии масштабирования данных. Экспериментально показано, что данный метод является вычислительно более эффективной альтернативой AutoML-системе AutoGluonTabular для решения задач регрессии на широком наборе эталонных данных.

7. Для задачи моделирования внешнего воздействия динамической системы с квадратичным функционалом качества теоретически обосновано сведение исходной бесконечномерной задачи к конечномерной и выведены границы статического и динамического сожалений. Проделаны вычислительные эксперименты, подтвердившие теоретические результаты.

В области программного обеспечения:

8. Разработан комплекс программ на языке Python, позволяющий проводить численное моделирование и сравнительный анализ предложенных онлайн алгоритмов, основанных на использовании случайных признаков Фурье: алгоритм VAW; двухуровневый алгоритм VAW^2; трёхуровневый алгоритм S-VAW^2. На основе данного комплекса проведены вычислительные эксперименты, подтвердившие теоретические выводы и продемонстрировавшие эффективность разработанных методов.

Степень достоверности результатов. Достоверность результатов, полученных в диссертации, обеспечивается строгостью приведённых доказательств, а также имеющимися публикациями в рецензируемых изданиях и выступлениями на конференциях. Все численные эксперименты, проводимые в рамках диссертационной работы, находятся в открытом доступе.

Апробация результатов. Результаты настоящего исследования были представлены на следующих конференциях.

- Всероссийская научно-практическая конференция «Математика, информатика, компьютерные науки, математическое моделирование, образование (МИКМО-2024)» (Симферополь, 2024);

- XIX Владикавказская молодёжная математическая школа (Владикавказ, 2024);
- Санкт-Петербургская молодёжная конференция по теории вероятностей и математической физике (Санкт-Петербург, 2024);
- Международная научная конференция «Порядковый анализ и смежные вопросы математического моделирования, XVIII: Теория операторов и дифференциальные уравнения» (РСО-А, Дзинага, 2025);
- Международная научная конференция «Современные методы и проблемы теории операторов и гармонического анализа и их приложения - 2025 (ОТНА-2025)» (Ростов-на-Дону, 2025).

Публикации и личный вклад автора. Основные результаты диссертационного исследования изложены в 7 научных публикациях, из них 4 в сборниках трудов конференции. Статья [71] опубликована в журнале, входящем в международную базу данных Scopus; статья [73] — в журнале, входящем в базы данных Scopus и Web of Science; получено свидетельство о государственной регистрации программы для ЭВМ [113]. Тезисы докладов [114], [109], [110], [111] опубликованы в материалах конференций.

Статьи [73], [71] опубликованы в соавторстве с научным руководителем. Д.Б. Рохлину принадлежат постановки задачи, указание методов исследования и общее руководство. Автору диссертации принадлежат доказательства теорем и численные эксперименты.

Основные положения, выносимые на защиту, являются личным вкладом автора.

Кроме того, статья [112] принята к публикации в журнале, входящем в перечень ВАК, а препринт находящейся на рецензии статьи [72], написанной в соавторстве с руководителем, размещён в ArXiv.

Структура и объём диссертации. Диссертационная работа состоит из введения, трёх глав, заключения, приложений и списка литературы, содержащего 115 наименований. Полный объём диссертации составляет 142 страницы (в том числе приложений 21 стр.).

Первая глава диссертации посвящена задаче оценивания условных математических ожиданий в моделях с марковской зависимостью и разработке соответствующего численного метода на основе алгоритма Вовка–Азури–Вармута и случайных признаков Фурье. Рассматривается задача аппроксимации условного математического ожидания $h(x) = \mathsf{E}(g(X_0, \ldots, X_s)|X_0 = x)$ по единственной траектории марковского стационарного процесса X_t, где g — некоторая функция. Задача сводится к задаче регрессии с признаками X_t и целевыми переменными $g(X_t, \ldots, X_{t+s})$. Для решения этой задачи рассматривается класс гипотез, сгенерированный m случайными признаками Фурье с равномерно ограниченными весами w. Для поиска весов используется алгоритм онлайн оптимизации Вовка-Азури-Вармута (VAW).

В параграфах 1.1 и 1.2 вводятся постановка задачи и общий алгоритм Вовка–Азури–Вармута (VAW) для нелинейных признаков и приводится его анализ для процессов, обладающих свойством геометрического β-перемешивания. Показано, что при ограничениях на l^∞-норму вектора весов ($w \in [-c, c]^m$) оценка ожидаемых квадратичных потерь имеет порядок $\mathcal{O}\left(\frac{c^2 m^3}{n}\right)$, что демонстрирует полиномиальную зависимость от числа признаков m.

Ключевым подходом главы, изложенным в параграфе 1.3, является применение техники случайных признаков Фурье для преодоления этого недостатка. Для случая, когда истинная регрессионная зависимость f принадлежит гильбертову пространству с воспроизводящим ядром (RKHS: Reproducing kernel Hilbert space) $H(\phi, p)$, выводится новая оценка сожаления порядка $\mathcal{O}\left(\frac{\|f\|_p^2}{\sqrt{n}}\right)$. Эта оценка не зависит полиномиально от размерности m, а определяется гладкостью целевой функции (через её норму $\|f\|_p$), что является значительным теоретическим улучшением и делает подход практически применимым.

В параграфе 1.4 результаты апробируются на модели векторной авторегрессии (VAR), для которой установлено выполнение всех необходимых теоретических условий.

В параграфе 1.5 подводятся итоги и формулируются выводы по всем результатам, полученным в главе 1.

Вторая глава диссертации посвящена разработке и анализу вычислительно эффективных методов многоядерной онлайн оптимизации. В ней рассмотре-

на задача регрессии с квадратичной функцией потерь в пространствах RKHS, как и в первой главе. Однако, в отличие от первой главы, данные здесь обладают состязательным, а не марковским свойством.

В параграфе 2.1 ставится задача построения ансамбля моделей на основе многоядерного подхода для решения проблемы выбора оптимального ядра в задачах нелинейной регрессии.

В параграфе 2.2 рассматривается базовый подход, при котором алгоритм VAW применяется к объединённому (конкатенированному) вектору признаков всех ядер. Для этого подхода выводится оценка сожаления, однако отмечается его главный недостаток — высокая вычислительная сложность, делающая его неприменимым для большого числа ядер.

Для преодоления этого недостатка в главе разработан новый двухуровневый численный метод VAW^2. Этот алгоритм обладает не только сильными теоретическими гарантиями (для него установлена оценка сожаления, сопоставимая с базовым подходом), но и существенно более низкой вычислительной сложностью, что является его ключевым практическим преимуществом. Также рассматриваются его модификации с использованием усечённых прогнозов и мета-алгоритма EWA.

В параграфе 2.3 представлены результаты численных экспериментов, подтверждающие теоретические выводы и демонстрирующие превосходство численного метода VAW^2 над существующими аналогами. Также представляется новый численный метод S-VAW^2, объединяющий метод VAW^2 со стратегиями предварительной обработки данных.

В параграфе 2.4 подводятся итоги и формулируются выводы по результатам, представленным во второй главе.

Третья глава диссертации посвящена применению методов онлайн оптимизации для решения задачи моделирования внешнего воздействия в рамках задачи вариационного исчисления с квадратичным функционалом качества.

В параграфе 3.1 ставится задача поиска оптимальной траектории в онлайн режиме, где на каждом шаге игроку противостоит «соперник», выбирающий непредсказуемое внешнее воздействие. Для решения этой бесконечномерной

задачи предложен метод аппроксимации искомого решения в ортонормированном базисе из тригонометрических полиномов, что позволяет свести её к стандартной конечномерной задаче онлайн оптимизации.

Ключевым теоретическим результатом главы, изложенным в параграфе 3.2, является доказательство корректности этого сведения. Показано, что основные параметры результирующей конечномерной задачи (диаметр множества допустимых решений, параметр сильной выпуклости, константы Липшица и гладкости) являются равномерно ограниченными вне зависимости от размерности аппроксимации. Это гарантирует применимость и устойчивость стандартных онлайн алгоритмов для решения исходной задачи.

В параграфе 3.3 выводятся оценки статического и динамического сожалений для ряда онлайн алгоритмов в различных сценариях поведения внешнего воздействия и приводятся результаты численных экспериментов, подтверждающие теоретические выводы.

В параграфе 3.4 подводятся итоги и формулируются выводы по результатам, представленным в третьей главе.

Благодарности. Автор искренне благодарит своего научного руководителя Дмитрия Борисовича Рохлина за наставничество и поддержку.

Диссертационная работа выполнена при поддержке Регионального научно-образовательного математического центра ЮФУ, соглашение Минобрнауки России № 075-02-2025-1720.

1 Применение численного метода Вовка-Азури-Вармута и случайных признаков Фурье в задаче регрессии с марковскими данными

Данная глава посвящена задаче аппроксимации условного математического ожидания $h(x) = \mathsf{E}(g(X_0, \ldots, X_s)|X_0 = x)$ для стационарного марковского процесса X_t по единственной наблюдаемой траектории. Проблема сводится к задаче регрессии, где X_t — признаки, а $g(X_t, \ldots, X_{t+s})$ — значение целевой функции, и включает в себя аппроксимацию неизвестной функции f_*, задающей траекторию динамической системы

$$X_{t+1} = f_*(X_t) + \xi_{t+1},$$

где ξ_t являются независимыми и одинаково распределёнными случайными величинами. Подобные задачи исследовались в работах, включая работу [106]. Более того, рассматриваемая проблематика тесно связана с современными направлениями исследований, такими как: аппроксимация операторов динамики (в частности, переходного оператора Маркова [65], оператора Купмана [58, 4] и стохастических генераторов [53]), глубокое обучение для марковских процессов (включая нейросетевые оценки переходных плотностей [19] и регуляризованные RNN для нелинейных систем [76]), непараметрические методы (такие как ядерные оценки для неэргодических процессов [90] и адаптивные алгоритмы в L^2_π [10]), а также приложения в управлении (например, идентификация для MPC [104] и обратные задачи в RL [45]).

В отличие от подхода [106], в данной главе используется класс функций, построенный на основе случайных признаков Фурье с ограниченными коэффициентами [68, 69, 70]. Для нахождения коэффициентов линейной регрессии применяется численный метод онлайн оптимизации Vovk-Azoury-Warmuth (VAW) [98, 6].

Известно, что для н.о.р. данных и квадратичной функции потерь оценка ошибки составляет $O(n^{-1/2})$ при использовании $\widetilde{O}(\sqrt{n})$ случайных признаков

[74, 15]. В этой главе будет показано, что аналогичный результат справедлив и для зависимых данных. Данный результат основан на подходе работы [2], где было доказано, что стабильный в некотором смысле алгоритм с небольшим сожалением обеспечивает малую оценку ошибки при выполнении условия перемешивания.

Для аппроксимации условного математического ожидания $h(x)$ рассматривается класс линейных комбинаций признаков с равномерно ограниченными коэффициентами. Применение численного метода VAW к данному классу функций в сочетании с результатами работы [2] позволяет получить верхнюю границу ошибки аппроксимации (теорема 1). На основе идей [70] и результатов, касающихся гильбертовых пространств с воспроизводящим ядром (RKHS), установлено, что функция $h(x)$ может быть аппроксимирована с произвольной точностью линейной комбинацией достаточного количества случайных признаков Фурье (теорема 2), при этом ожидаемая ошибка для алгоритма VAW с $n^{1/2}$ признаками имеет порядок $\widetilde{O}(n^{-1/2})$. В качестве математической модели рассматривается стохастический процесс, описываемый векторной авторегрессией (VAR). Разрабатывается численный метод для оценки параметров этой модели в онлайн режиме.

1.1 Постановка задачи

Пусть $P(x, B)$ — переходное ядро в $\mathbb{R}^d$, обладающее единственным инвариантным распределением π:

$$\int_{\mathbb{R}^d} \pi(dx)P(x, B) = \pi(B), \quad \forall B \in \mathscr{B}(\mathbb{R}^d),$$

где $\mathscr{B}(\mathbb{R}^d)$ является борелевской σ-алгеброй пространства $\mathbb{R}^d$ [108]. Рассмотрим однородный стационарный марковский процесс $(X_t)_{t=0}^{\infty}$, определённый на

вероятностном пространстве $(\Omega, \mathscr{F}, \mathsf{P})$:

$$\mathsf{P}(X_{t+1} \in B | \mathscr{F}_t) = P(X_t, B), \quad \mathscr{F}_t = \sigma(X_0, \ldots, X_t).$$

Здесь $\mathscr{F}_t$ — естественная фильтрация процесса, то есть σ-алгебра, порожденная наблюдениями $X_0, \ldots, X_t$. Напомним, что процесс называется однородным, если переходное ядро $P(x, B)$ не зависит от времени t, и стационарным, если начальное распределение совпадает с инвариантным распределением π:

$$\mathsf{P}(X_0 \in B) = \pi(B).$$

В этом случае из марковского свойства следует, что

$$\mathsf{P}(X_t \in B) = \pi(B)$$

для всех $t \geq 0$, то есть распределение процесса не меняется со временем.

Положим $Z_t = (X_t, \ldots, X_{t+s})$, где $s \geq 1$. Процесс $(Z_t)_{t=0}^{\infty}$ является марковским относительно своей естественной фильтрации $(\mathscr{H}_t)_{t=0}^{\infty}$, где $\mathscr{H}_t = \sigma(Z_0, \ldots, Z_t)$. Также заметим, что стационарность процесса $(X_t)_{t=0}^{\infty}$ влечет за собой стационарность процесса $(Z_t)_{t=0}^{\infty}$, так как его распределение определяется конечномерными распределениями $(X_t, \ldots, X_{t+s})$, которые не зависят от t. Инвариантная мера Π процесса $(Z_t)_{t=0}^{\infty}$ совпадает с совместным распределением вектора $(X_0, \ldots, X_s)$:

$$\Pi(B_0 \times \cdots \times B_s) = \mathsf{P}(X_0 \in B_0, \ldots, X_s \in B_s).$$

Пусть функция $g\colon \mathbb{R}^{(s+1)d} \to \mathbb{R}$, где $s \geq 1$, является борелевской. Цель исследования в данной главе — аппроксимация функции

$$h(x) = \mathsf{E}[g(X_0, X_1, \ldots, X_s) \mid X_0 = x], \quad x \in \mathbb{R}^d, \tag{1}$$

используя единственную наблюдаемую траекторию $(X_0, \ldots, X_n)$, $n \gg s$.

Для обеспечения сходимости и устойчивости оценок предполагается, что функция g равномерно ограничена, то есть существует константа $\overline{g} > 0$ такая,

что

$$|g(x_0, \ldots, x_s)| \leq \overline{g} \quad \text{для всех } (x_0, \ldots, x_s) \in \mathbb{R}^{(s+1)d}.$$

В качестве примера рассмотрим случай, когда функция g является индикаторной функцией множества $B \in \mathscr{B}(\mathbb{R}^d)$:

$$g(x_0, x_1, \ldots, x_s) = \mathbb{1}_B(x_s) = \begin{cases} 1, & x_s \in B, \\ 0, & x_s \notin B. \end{cases}$$

Тогда выражение (1) принимает вид:

$$h(x) = \mathsf{P}(X_s \in B \mid X_0 = x),$$

что соответствует вероятности перехода процесса X из состояния x в множество B за s шагов.

Задача аппроксимации функции $h(x)$ может быть сформулирована в рамках регрессионного подхода, где состояния процесса X_t выступают в роли признаков, а соответствующие значения функции $g(X_t, \ldots, X_{t+s})$ используются в качестве целевых переменных. Качество построенной модели $\hat{h}(x)$, аппроксимирующей искомую функцию, оценивается посредством ожидаемых квадратичных потерь, называемых также риском. Риск оценки $\hat{h}(x)$ относительно инвариантного распределения π определяется следующим выражением:

$$\mathcal{R}(\hat{h}) = \mathsf{E}_{X \sim \pi}\left[(h(X) - \hat{h}(X))^2\right] = \int_{\mathbb{R}^d} (h(x) - \hat{h}(x))^2 \pi(dx). \tag{2}$$

Задача заключается в нахождении такой оценки $\hat{h}$, которая минимизирует данный функционал риска, используя единственную траекторию наблюдаемого процесса.

1.2 Алгоритм Вовка-Азури-Вармута на нелинейных признаках

Для решения поставленной задачи предлагается использовать численный метод онлайн оптимизации, известный как алгоритм Вовка-Азури-Вармута (Vovk-Azoury-Warmuth, VAW) [64]. В отличие от традиционных подходов, этот метод предполагает последовательное взаимодействие алгоритма со средой, что особенно эффективно для моделирования процессов с нестационарными данными.

Рассматривается последовательность обучающих пар $(x_t, y_t)_{t=1}^T$, где x_t — вектор признаков на шаге t, а y_t — соответствующее целевое значение. Пусть $w_t \in W$ — это вектор весов, который алгоритм выбирает на каждом шаге t для формирования предсказания. В качестве меры ошибки используется квадратичная функция потерь $l_t(w_t) = (\langle x_t, w_t \rangle - y_t)^2$.

На каждом шаге t алгоритм:

1. Получает входной вектор x_t
2. Формирует предсказание $\langle x_t, w_t \rangle$
3. Получает истинное значение y_t
4. Вычисляет квадратичные потери $l_t(w_t)$

Эффективность численного метода оценивается через величину сожаления (regret), которая сравнивает накопленные потери метода с потерями некоторого произвольного предсказателя:

$$R_T(w) = \sum_{t=1}^{T} l_t(w_t) - \sum_{t=1}^{T} l_t(w), \quad \forall w. \tag{3}$$

Алгоритм Vovk-Azoury-Warmuth (VAW) обеспечивает оптимальные асимптотические оценки сожаления для квадратичных потерь. Согласно теореме 3.1 в работе [64], сожаление для алгоритма VAW имеет порядок $O\left(\sqrt{d \log T}\right)$, где d — размерность пространства признаков. Данная оценка является неулучшаемой.

Для работы алгоритма VAW необходимо, чтобы на каждом шаге t вектор признаков x_t был известен до определения вектора весов w_t [64]. В этом случае вектор весов w_t вычисляется следующим образом:

$$w_t = \operatorname*{argmin}_w \left(\frac{1}{2} \sum_{j=1}^{t-1} (\langle w, x_j \rangle - y_j)^2 + \frac{1}{2} \langle w, x_t \rangle^2 + \frac{\lambda}{2} \|w\|_2^2 \right). \tag{4}$$

Здесь $\frac{1}{2} \sum_{j=1}^{t-1} (\langle w, x_j \rangle - y_j)^2$ — кумулятивные квадратичные потери до момента времени $t-1$. Член $\frac{1}{2} \langle w, x_t \rangle^2$ можно интерпретировать как штраф за предсказание на текущем шаге, сделанное до наблюдения истинного значения y_t. Регуляризационный член $\frac{\lambda}{2} \|w\|_2^2$ с параметром $\lambda > 0$ обеспечивает устойчивость и единственность решения, предотвращая переобучение. В работе [64] была установлена следующая оценка сожаления для алгоритма VAW.

Теорема 1.1. *Предположим, что $\|x_t\|_2 \leq R$ и $|y_t| \leq Y$ для всех $t = 1, \ldots, T$. Тогда для алгоритма, определённого выражением (4), выполняется следующая оценка:*

$$\sum_{t=1}^{T} (y_t - \langle w_t, x_t \rangle)^2 - \inf_{w \in \mathbb{R}^d} \sum_{t=1}^{T} (y_t - \langle w, x_t \rangle)^2 \leq \lambda \|w^*\|^2 + dR^2 \ln \left(1 + \frac{TR^2}{\lambda d} \right), \tag{5}$$

где d — размерность пространства признаков.

Рассмотрим параметрический класс линейных функций

$$\mathcal{H} = \{ \langle \Phi(x), w \rangle : |w_i| \leq c,\ i = 1, \ldots, m \}$$

с нелинейными признаками $\Phi(x) = (\phi_1(x), \ldots, \phi_m(x))$. Будем предполагать, что $|\phi_i(x)| \leq 1$.

Для оценки качества аппроксимации функции $h(x)$ введем разность ожидаемых квадратичных потерь между двумя произвольными векторами весов v

и w как:

$$\mathcal{E}(v,w) = \frac{1}{2}\int \left(\langle \Phi(x), v\rangle - h(x)\right)^2 \pi(dx) - \frac{1}{2}\int \left(\langle \Phi(x), w\rangle - h(x)\right)^2 \pi(dx).$$

Пусть вектор весов $\widehat{w}_n$ получен с помощью некоторого алгоритма по единственной наблюдаемой траектории $(X_0, \dots X_n)$, а w — произвольный элемент из множества $\mathcal{W} = [-c, c]^m$. Цель состоит в получении верхней границы для оценки ошибки $\mathsf{E}\mathcal{E}(\widehat{w}_n, w)$ для любого, и, следовательно, наилучшего, вектора коэффициентов w. Рассматриваемая проблема тесно связана с задачей выпуклого агрегирования при $c = 1$, так как в этом случае множество $\mathcal{H}$ содержит выпуклые комбинации признаков $\pm\phi_i$. Для независимых и одинаково распределённых данных исследование [94] показало, что нижняя граница ошибки оценки имеет порядок m/n, где m обозначает количество признаков, а n — число примеров. Эта оценка достижима посредством минимизации эмпирического риска при использовании квадратичной функции потерь, как продемонстрировано в работе [52].

Приведенные результаты основаны на подходе, представленном в работе [2]. В связи с этим будут использованы аналогичные обозначения. Пусть $F(w, Z_t)$ — функция потерь, определённая на расширенном векторе состояния $Z_t = (X_t, Y_t)$, где X_t — текущее состояние процесса, а $Y_t = g(Z_t)$ — соответствующее значение целевой функции:

$$F(w, Z_t) = \frac{1}{2}\big(\langle \Phi(X_t), w\rangle - Y_t\big)^2.$$

$$f(w) = \mathsf{E}F(w, Z_0) = \int F(w, z)\Pi(dz).$$

Данное выражение допускает следующее преобразование:

$$f(v) = \frac{1}{2}\Big(\mathsf{E}\langle\Phi(X_0), v\rangle^2 - 2\mathsf{E}\big[\langle\Phi(X_0), v\rangle h(X_0)\big] + \mathsf{E}g^2(X_0, \ldots, X_s)\Big)$$

$$= \frac{1}{2}\mathsf{E}\Big(\big(\langle\Phi(X_0), v\rangle - h(X_0)\big)^2 + g^2(Z_0) - h^2(X_0)\Big),$$

где последнее равенство следует из определения $h(x) = \mathsf{E}(g(Z_0)|X_0 = x)$ и тождества:

$$\mathsf{E}\big[\langle\Phi(X_0), v\rangle Y_0\big] = \mathsf{E}\Big[\mathsf{E}\big(\langle\Phi(X_0), v\rangle g(Z_0)\big|X_0\big)\Big]$$

$$= \mathsf{E}\Big[\langle\Phi(X_0), v\rangle \mathsf{E}\big(g(Z_0)|X_0\big)\Big] = \mathsf{E}\big[\langle\Phi(X_0), v\rangle h(X_0)\big].$$

Таким образом

$$\mathcal{E}(v, w) = f(v) - f(w).$$

По наблюдаемой траектории $X_0, \ldots, X_{t+s}$ построим последовательность весов w_t с помощью алгоритма VAW (4). Учитывая, что теперь рассматриваются нелинейные признаки $\Phi(X)$, на каждом шаге t вектор весов w_t вычисляется как решение следующей задачи минимизации с ограничениями:

$$w_t = \operatorname*{argmin}_{w\in\mathcal{W}} \left(\frac{1}{2}\sum_{j=1}^{t-1}(\langle w, \Phi(X_j)\rangle - Y_j)^2 + \frac{1}{2}\langle w, \Phi(X_t)\rangle^2 + \frac{\epsilon}{2}\|w\|_2^2\right). \qquad (6)$$

Эта формула может быть переписана в альтернативных формах, например:

$$w_t = \operatorname*{argmin}_{w\in\mathcal{W}} \left(-\sum_{j=1}^{t-1} Y_j\langle w, \Phi(X_j)\rangle + \frac{1}{2}\sum_{j=1}^{t-1}\langle w, \Phi(X_j)\rangle^2 + \frac{1}{2}\langle w, \Phi(X_t)\rangle^2 + \frac{\epsilon}{2}\|w\|_2^2\right)$$

$$= \operatorname*{argmin}_{w\in\mathcal{W}} \left(\sum_{j=1}^{t-1}\langle\nabla_w F(w_j, Z_j), w\rangle + \frac{1}{2}\sum_{j=1}^{t-1}\langle w_j - w, \Phi(X_j)\rangle^2\right.$$

$$\left. + \frac{1}{2}\langle w, (\Phi(X_t)\Phi(X_t)^T + \epsilon I)w\rangle\right).$$

Для случая, когда $\mathcal{W} = \mathbb{R}^m$, решение имеет явный вид:

$$w_t = \underset{w \in \mathbb{R}^m}{\operatorname{argmin}} \left(-\sum_{j=1}^{t-1} Y_j \langle w, \Phi(X_j) \rangle + \frac{1}{2} \langle S_t w, w \rangle \right) = S_t^{-1} \left(\sum_{j=1}^{t-1} Y_j \Phi(X_j) \right), \quad (7)$$

где $S_t = \epsilon I + \sum_{j=1}^{t} \Phi(X_j)\Phi(X_j)^T$. Для эффективного вычисления S_t^{-1} на каждом шаге t можно использовать формулу Шермана-Моррисона:

$$S_t^{-1} = S_{t-1}^{-1} - \frac{S_{t-1}^{-1}\Phi(X_t)\Phi(X_t)^T S_{t-1}^{-1}}{1 + \Phi(X_t)^T S_{t-1}^{-1}\Phi(X_t)}. \quad (8)$$

Введем функцию $\ell(a, y) = (a - y)^2/2$. Тогда $F(w, z) = \ell(\langle w, \Phi(x) \rangle, y)$. Ввиду ограниченности весов и признаков ($|\langle w, \Phi(x) \rangle| \leq cm$ для $w \in \mathcal{W}$ и $|y| \leq \overline{g}$) заметим, что функция ℓ обладает свойством Липшица с константой $L = \overline{g} + cm$:

$$|\ell(a, y) - \ell(b, y)| = \left| y - \frac{a + b}{2} \right| \cdot |a - b| \leq L|a - b|.$$

Отсюда вытекает, что:

$$|F(w, z) - F(v, z)| \leq L|\langle w - v, \Phi(x) \rangle| \leq L\sqrt{m}\|w - v\|,$$

$$F(w, z) \leq |\langle w, \Phi(x) \rangle|^2 + y^2 \leq \overline{F} = c^2 m^2 + \overline{g}^2.$$

Таким образом, получаем следующее выражение для сожаления:

$$\mathfrak{R}_n(w) = \sum_{t=1}^{n} (F(w_t, Z_t) - F(w, Z_t)).$$

Также нам понадобится оценка сожаления, полученную в исследовании [2, предложение 3], где верхняя граница r для нормы признаков соответствует $\sqrt{m}$

в наших обозначениях:

$$\begin{aligned}\mathfrak{R}_n(w) &\le \frac{9L^2m}{2}\ln\left(1+\frac{mn}{\epsilon}\right)+\frac{\epsilon}{2}\|w\|^2 \\ &= \frac{9(\overline{g}+cm)^2m}{2}\ln\left(1+\frac{mn}{\epsilon}\right)+\frac{\epsilon}{2}\|w\|^2. \end{aligned} \tag{9}$$

В отличие от сценария с н.о.р. данными, преобразование последовательности оценок w_t в единую агрегированную оценку $\widehat{w}_n$ для получения оценки ожидаемого сожаления (процедура, известная как «online-to-batch conversion» [16]) не является тривиальной задачей. Пример, приведённый в работе [2], показывает, что малого сожаления недостаточно в общем случае. Дополнительно требуется свойство устойчивости численного метода. Ключевым результатом, полученным в работе [2], является следующее свойство устойчивости алгоритма VAW с ограничениями (6):

$$\begin{aligned}\sum_{t=1}^{n-\tau}(F(w_t,z_{t+\tau})-F(w_{t+\tau},z_{t+\tau})) &\le 7L^2\tau m\ln\left(1+\frac{mn}{\varepsilon}\right) \\ &= 7(\overline{g}+cm)^2\tau m\ln\left(1+\frac{mn}{\varepsilon}\right). \end{aligned} \tag{10}$$

Эта формула соответствует формуле (26) в [2].

Дадим определение коэффициента β-перемешивания для σ-алгебр $\mathscr{A}$ и $\mathscr{B}$:

$$\beta(\mathscr{A},\mathscr{B}) = \mathsf{E}\left[\sup_{B\in\mathscr{B}}|\mathsf{P}(B|\mathscr{A})-\mathsf{P}(B)|\right].$$

Пусть P_n — переходное ядро за n шагов, задаваемое соотношением $P_n(x,B) = \mathsf{P}(X_n \in B|X_0 = x)$. Коэффициент β-перемешивания для стационарного марковского процесса X определяется следующим образом:

$$\beta_X(n) = \beta(\sigma(X_0),\sigma(X_k,k\ge n)) = \beta(\sigma(X_0),\sigma(X_n)),$$

где последнее равенство следует из работы [26, предложение F.3.2]. Также из исследований [27, 26] известно, что $\beta_X(n)$ может быть выражен через расстоя-

ние по вариации

$$\beta_X(n) = \int_{\mathbb{R}^d} \pi(dx) d_{\mathrm{TV}}(\pi, P_n(\cdot; x)),$$

где

$$d_{\mathrm{TV}}(\mathsf{P}, \widetilde{\mathsf{P}}) = \sup_{A \in \mathscr{B}(\mathbb{R}^d)} |\mathsf{P}(A) - \widetilde{\mathsf{P}}(A)| = \frac{1}{2} \mathsf{E}_{\mathsf{Q}} |\xi - \widetilde{\xi}| \tag{11}$$

— расстояние по вариации. Здесь ξ и $\widetilde{\xi}$ — плотности мер P и $\widetilde{\mathsf{P}}$ относительно доминирующей меры Q, например $\mathsf{Q} = (\mathsf{P} + \widetilde{\mathsf{P}})/2$ [85]. Из определения следует, что для процесса с расширенным состоянием $Z_t = (X_t, \ldots, X_{t+s})$ коэффициент перемешивания связан с исходным процессом X соотношением

$$\beta_Z(n) = \beta_X(n - s), \quad Z_t = (X_t, \ldots, X_{t+s}),$$

что также обсуждается в работе [58, лемма 3.6.5].

Теорема 1.2. *Пусть последовательность векторов весов w_t генерируется с помощью алгоритма VAW с ограничениями (6), а $\overline{w}_n = \sum_{t=1}^n w_t/n$. Тогда для любых $w \in [-c, c]^m$ и $\tau \in \mathbb{N}$ справедливо, что*

$$\mathsf{E}\mathcal{E}(\overline{w}_n, w) \le 2cLm\beta_X(\tau - s + 1) + \left(\frac{9}{2} + 7\tau\right) \frac{L^2 m}{n} \ln\left(1 + \frac{mn}{\epsilon}\right) + \frac{2\tau \overline{F}}{n} + \frac{\epsilon}{2n}\|w\|^2, \tag{12}$$

где $L = \overline{g} + cm$ и $\overline{F} = c^2 m^2 + \overline{g}^2$. В частности, если процесс $(X_t)_{t=0}^{\infty}$ является геометрически β-перемешивающим, то есть $\beta_X(n) \le \beta_X(0) e^{-\gamma n}$, то

$$\mathsf{E}\mathcal{E}_n(\widehat{w}_n, w) = \widetilde{O}\left(\frac{c^2 m^3 + \overline{g}^2 m}{n}(s + 1/\gamma)\right). \tag{13}$$

Обозначение $\widetilde{O}$ скрывает абсолютные константы, параметр ϵ и полилогарифмические множители.

Доказательство. По неравенству Йенсена

$$\mathsf{E}\mathcal{E}(\overline{w}_n, w) = \mathsf{E}f(\overline{w}_n) - f(w) \le \frac{1}{n}\sum_{t=1}^{n}(\mathsf{E}f(w_t) - f(w)). \tag{14}$$

Представим $f(w_t) - f(w)$ в виде суммы

$$\sum_{t=1}^{n}(f(w_t) - f(w)) = S_1 + S_2 + S_3 + S_4, \tag{15}$$

где

$$\begin{aligned}
S_1 &= \sum_{t=1}^{n}(f(w_t) - F(w_t, Z_{t+\tau}) + F(w, Z_{t+\tau}) - f(w)),\\
S_2 &= \sum_{t=1}^{n}(F(w_t, Z_{t+\tau}) - F(w_t, Z_t)),\\
S_3 &= \sum_{t=1}^{n}(F(w_t, Z_t) - F(w, Z_t)),\\
S_4 &= \sum_{t=1}^{n}(F(w, Z_t) - F(w, Z_{t+\tau})).
\end{aligned}$$

Математическое ожидание слагаемого S_1 оценивается с использованием свойства перемешивания процесса $(Z_t)_{t=0}^{\infty}$. Поскольку w_t измерим относительно $\sigma(Z_0, \ldots, Z_{t-1}) = \mathscr{H}_{t-1}$, то

$$\mathsf{E}F(w_t, Z_{t+\tau}) = \mathsf{E}\mathsf{E}(F(w_t, Z_{t+\tau})|\mathscr{H}_{t-1}) = \mathsf{E}\int_{\mathbb{R}^d} F(w_t, z)\, P_{\tau+1}(Z_{t-1}, dz).$$

Отсюда следует, что

$$\begin{aligned}
\mathsf{E}(f(w_t) - F(w_t, Z_{t+\tau})) &= \mathsf{E}\int_{\mathbb{R}^d} F(w_t, z)\Pi(dz) - \mathsf{E}\int_{\mathbb{R}^d} F(w_t, z)\, P_{\tau+1}(Z_{t-1}, dz)\\
&= \mathsf{E}\int_{\mathbb{R}^d} F(w_t, z)(\rho(z) - \rho_{\tau+1}(z; Z_{t-1}))\mu_t(dz),\\
\mathsf{E}(F(w, Z_{t+\tau}) - f(w)) &= \mathsf{E}\int_{\mathbb{R}^d} F(w, z)(\rho_{\tau+1}(z; Z_{t-1}) - \rho(z))\mu_t(dz),
\end{aligned}$$

где ρ и $\rho_{\tau+1}(\cdot; Z_{t-1})$ — плотности мер Π и $P_{\tau+1}(Z_{t-1}, \cdot)$ относительно вероятност-

ной меры $\mu_t = (\Pi + P_{\tau+1}(Z_{t-1}, \cdot))/2$. Используя свойство Липшица, получаем, что

$$\begin{aligned}
&\mathsf{E}(F(w, Z_{t+\tau}) - f(w) + f(w_t) - F(w_t, Z_{t+\tau})) \\
&\leq \mathsf{E}\int_{\mathbb{R}^d}(F(w,z) - F(w_t,z))(\rho_{\tau+1}(z; Z_{t-1}) - \rho(z))\mu_t(dz) \\
&\leq L\sqrt{m}\mathsf{E}\left(\|w - w_t\|\int_{\mathbb{R}^d}|\rho_{\tau+1}(z; Z_{t-1}) - \rho(z)|\mu_t(dz)\right) \\
&\leq 2cLm\mathsf{E}d_{\mathrm{TV}}(\mathsf{P}_{\tau+1}(\cdot; Z_{t-1}), \Pi) = 2cLm\int_{\mathbb{R}^d}\Pi(dz)d_{\mathrm{TV}}(\mathsf{P}_{\tau+1}(\cdot; z), \Pi) \\
&= 2cLm\beta_Z(\tau+1) = 2cLm\beta_X(\tau - s + 1).
\end{aligned}$$

Оценка для S_2 следует из свойства устойчивости (10):

$$\begin{aligned}
\sum_{t=1}^{n}(F(w_t, Z_{t+\tau}) - F(w_t, Z_t)) &= \sum_{t=1}^{n-\tau}((F(w_t, Z_{t+\tau}) - F(w_{t+\tau}, Z_{t+\tau})) \\
&\quad + \sum_{t=n-\tau+1}^{n}F(w_t, Z_{t+\tau}) - \sum_{t=1}^{\tau}F(w_t, Z_t) \\
&\leq \sum_{t=1}^{n-\tau}((F(w_t, Z_{t+\tau}) - F(w_{t+\tau}, Z_{t+\tau})) + \tau\overline{F} \\
&\leq 7L^2\tau m\ln\left(1 + \frac{mn}{\varepsilon}\right) + \tau\overline{F}.
\end{aligned}$$

Слагаемое S_3 является сожалением ($\mathfrak{R}_n(w)$), а слагаемое S_4 оценивается элементарным образом:

$$S_4 = \sum_{t=1}^{n}(F(w, Z_t) - F(w, Z_{t+\tau})) = \sum_{t=1}^{\tau}F(w, Z_t) - \sum_{t=n+1}^{n+\tau}F(w, Z_t) \leq \tau\overline{F}.$$

Применив математическое ожидание в (15), получаем, что

$$\begin{aligned}
\sum_{t=1}^{n}(\mathsf{E}f(w_t) - f(w)) \leq 2cLmn\beta_X(\tau - s + 1) &+ 7L^2\tau m\ln\left(1 + \frac{mn}{\varepsilon}\right) \\
&+ 2\tau\overline{F} + \mathsf{E}\mathfrak{R}_n(w).
\end{aligned}$$

Из оценок (9) и (14) следует искомое неравенство (12).

Теперь предположим, что процесс $(X_t)_{t=0}^{\infty}$ является геометрически β-перемешивающим: $\beta_X(n) \le \beta_X(0)e^{-\gamma n}$. Положим $\tau = \gamma^{-1}\ln n + s - 1$. Из (12) вытекает, что

$$\begin{aligned}\mathsf{E}\mathcal{E}_n(\widehat{w}_n, w) &= O\left(\frac{cLm\beta_X(0)}{n} + \left(\frac{\ln n}{\gamma} + s\right)\left[\frac{L^2 m}{n}\ln\left(1 + \frac{mn}{\epsilon}\right) + \frac{\overline{F}}{n}\right] + \frac{\epsilon}{n}c^2 m\right) \\ &= \widetilde{O}\left(\frac{c^2m^3 + \overline{g}^2 m}{n}(s + 1/\gamma)\right). \qquad \square\end{aligned}$$

Замечание 1.1. Для алгоритма VAW без ограничений (7) сожаление допускает следующую оценку [64, теорема 7.24]:

$$\mathfrak{R}_n(w) \le \frac{\overline{g}^2 m}{2}\ln\left(1 + \frac{n}{\epsilon}\right) + \frac{\epsilon}{2}\|w\|^2.$$

Более того, если X_t являются н.о.р., по стандартной процедуре «online-to-batch conversion» [64, теорема 3.1]

$$\mathsf{E}\mathcal{E}(\overline{w}_n, w) = \mathsf{E}f(\overline{w}_n) - f(w) \le \frac{\mathsf{E}\mathfrak{R}_n(w)}{n} = \widetilde{O}\left(\frac{(\overline{g}^2 + c^2)m}{n}\right) \qquad (16)$$

для $w \in [-c, c]^m$. Оценка (13) не воспроизводит оценку (16). Это указывает на то, что член c^2m^3 в (13) может быть улучшен.

Пример 1.1. Пусть $(X_t)_{t=0}^{\infty}$ — стационарное пространственно-неоднородное случайное блуждание по множеству целых чисел $\mathbb{Z}$. Предположим, что $\mathsf{P}(X_{t+1} = j | X_t = i) = 0$ для $|j - i| > 1$, и процесс является геометрически β-перемешивающим. Чтобы оценить $\mathsf{P}(X_{t+s} \in A | X_t = i)$, где $A = \{j \in \mathbb{Z} : |j| \le k\}$, положим $g(x_s) = I_{\{x_s \in A\}}$. Заметим, что

$$h(x) = \mathsf{E}(g(X_s)|X_0 = x) = 0$$

для $|x| > k + s$. Если признаки $\phi_i(x) = I_{\{x=i\}}$ для $|i| \le k + s$, то функцию $h(x)$

можно представить в виде:

$$h(x) = \sum_{|i| \le k+s} w_i^* \phi_i(x),$$

с некоторыми весами $w_i^* \in [-1, 1]$. Таким образом, для точного представления функции h достаточно $m = 2(s+k)+1 = 2s+|A|$ признаков ϕ_i. Из оценки (13) и определения $\mathcal{E}(v, w)$ следует, что

$$\begin{aligned} \mathsf{E}\mathcal{E}(\overline{w}_n, w) &= \frac{1}{2}\mathsf{E}\sum_{i\in\mathbb{Z}} \left(\langle \Phi(i), \overline{w}_n\rangle - h(i)\right)^2 \pi(i) \\ &= \frac{1}{2}\mathsf{E}\sum_{i\in\mathbb{Z}} \left(\overline{w}_{n,i} - \mathsf{P}(X_s \in A | X_0 = i)\right)^2 \pi(i) \\ &= \widetilde{O}\left(\frac{(2s+|A|)^3}{n}(s+1/\gamma)\right). \end{aligned}$$

1.3 Случайные признаки Фурье

Пусть $\phi : \mathbb{R}^d \times \Theta \to [-1, 1]$ — непрерывная функция, а Θ — замкнутое подмножество конечномерного пространства. Следуя концепции, представленной в работе [69], введём функциональное пространство

$$\mathcal{H}(\phi, p) = \left\{ x \mapsto f(x) = \int \alpha(\theta)\phi(x;\theta)\, d\theta : \int \frac{\alpha^2(\theta)}{p(\theta)}\, d\theta < \infty \right\} \qquad (17)$$

где $\alpha(\theta)$ — некоторая интегрируемая функция, а $p(\theta)$ — фиксированная положительная весовая функция на Θ. Пусть функции $f(x)$ и $g(x)$ принадлежат пространству $\mathcal{H}$. Их скалярное произведение задаётся формулой

$$\langle f, g\rangle_{\mathcal{H}} = \int \frac{\alpha(\theta)\beta(\theta)}{p(\theta)} d\theta,$$

где $f(x) = \int \alpha(\theta)\phi(x;\theta)d\theta$, $g(x) = \int \beta(\theta)\phi(x;\theta)d\theta$. Это скалярное произведение порождает норму $\|f\|_{\mathcal{H}}^2 = \int \frac{\alpha^2(\theta)}{p(\theta)} d\theta$ и делает $\mathcal{H}(\phi, p)$ гильбертовым пространством.

Ключевым свойством этого пространства является наличие воспроизводя-

щего ядра. В работе [69, предложение 4.1] доказано, что $\mathcal{H}(\phi, p)$ является RKHS с воспроизводящим ядром

$$k(x, y) = \int p(\theta)\phi(x; \theta)\phi(y; \theta)\, d\theta. \tag{18}$$

Фундаментальное свойство воспроизводящего ядра заключается в том, что для любой функции $f \in \mathcal{H}$ и любого $x \in \mathbb{R}^d$ выполняется тождество:

$$\begin{aligned}\langle f, k(x, \cdot)\rangle_{\mathcal{H}} &= \left\langle \int \alpha(\theta)\phi(\cdot; \theta)\, d\theta, \int p(\theta)\phi(x; \theta)\phi(\cdot; \theta)\, d\theta \right\rangle_{\mathcal{H}} \\ &= \int \frac{\alpha(\theta)(p(\theta)\phi(x; \theta))}{p(\theta)}\, d\theta = \int \alpha(\theta)\phi(x; \theta)\, d\theta = f(x).\end{aligned}$$

Это свойство позволяет «воспроизводить» значение функции в любой точке x через скалярное произведение с ядром.

Понятие универсального ядра играет большую роль в теории RKHS, поскольку оно гарантирует, что соответствующее пространство функций является достаточно богатым для аппроксимации любого элемента из широкого класса функций. Ядро k называется универсальным, если соответствующее RKHS $\mathcal{H}_K$ плотно в $L^2(\mathbb{R}^d; \rho)$ для любой вероятностной меры ρ на борелевской σ-алгебре $\mathscr{B}(\mathbb{R}^d)$ [14]. Напомним также, что ядро называется трансляционно-инвариантным, если его значение зависит только от разности аргументов $k(x, y) = \kappa(x - y)$, причём функция κ является положительно определённой, то есть для любого конечного набора точек $x_1, \ldots, x_n \in \mathbb{R}^d$ и любых действительных чисел $c_1, \ldots, c_n \in \mathbb{R}$ выполняется неравенство:

$$\sum_{i,j=1}^{n} c_i c_j \kappa(x_i - x_j) \geq 0.$$

Согласно теореме Бохнера, любая непрерывная положительно определённая функция $\kappa(x)$ допускает представление в виде преобразования Фурье некоторой неотрицательной σ-аддитивной меры Λ на $\mathscr{B}(\mathbb{R}^d)$:

$$\kappa(x) = \int_{\mathbb{R}^d} e^{i\langle x, y\rangle}\, \Lambda(dy). \tag{19}$$

Данное представление связывает свойства ядра в пространственной области с его свойствами в частотной области. Частным случаем трансляционно-инвариантных ядер являются радиальные ядра, где функция

$$\kappa(x) = \int_{[0,\infty)} e^{-t\|x\|^2}\, \nu(dt) \tag{20}$$

для некоторой неотрицательной σ-аддитивной меры ν на $\mathscr{B}([0,\infty))$.

Для трансляционно-инвариантных ядер $k(x,y) = k(x-y)$ универсальность соответствующего гильбертова пространства с воспроизводящим ядром $\mathcal{H}_k$ эквивалентна тому, что спектральная мера Λ является строго положительной на всём пространстве $\mathbb{R}^d$. В случае радиальных ядер $K(x,y) = k(\|x-y\|)$ универсальность $\mathcal{H}_k$ достигается тогда и только тогда, когда соответствующая радиальная спектральная мера ν не является дельта-мерой в нуле.

Гауссовское ядро $k(x,y) = \exp(-\sigma\|x-y\|^2)$ с параметром $\sigma > 0$ удовлетворяет любому из следующих критериев универсальности:

1. Его спектральная мера (преобразование Фурье) строго положительна всюду в $\mathbb{R}^d$

2. Соответствующая радиальная мера ν имеет ненулевую массу вне точки 0 [88].

Следуя подходу [69], положим $\|f\|_p = \sup_\theta |\alpha(\theta)/p(\theta)|$ и рассмотрим подпространство

$$\widetilde{\mathcal{H}}(\phi,p) = \left\{ x \mapsto f(x) = \int_\Theta \alpha(\theta)\phi(x;\theta)d\theta : \|f\|_p < \infty \right\}$$

пространства $\mathcal{H}(\phi;p)$. Очевидно, что

$$\|f\|^2_{\mathcal{H}(\phi,p)} = \int \frac{\alpha^2(\theta)}{p(\theta)}\, d\theta \le \|f\|_p^2.$$

Следующий результат содержится в работе [69].

Лемма 1.1. *Пусть* $\phi(x;\theta) = \cos(\langle\omega, x\rangle + b)$, *где* $\theta = (\omega, b)$, *причём* $p(\omega, b) = p_1(\omega)p_2(b)$, *где*

$$p_1(\omega) = \frac{1}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{\|\omega\|^2}{2\sigma^2}\right), \quad p_2(b) = \begin{cases} 1/(2\pi), & b \in (-\pi, \pi), \\ 0, & \textit{в противном случае}. \end{cases}$$

Тогда $\mathcal{H}(\phi, p)$ *является RKHS с гауссовским ядром*

$$k(x, y) = \frac{1}{2}\exp\left(-\frac{\sigma^2}{2}\|x - y\|_2^2\right),$$

и его подпространство $\widetilde{\mathcal{H}}(\phi, p)$ *плотно в* $L^2(\mathbb{R}^d; \mu)$.

Доказательство. Действительно, $\mathcal{H}(\phi, p)$ является RKHS с ядром (18). Покажем, что это ядро является гауссовским:

$$\begin{aligned} k(x, y) &= \int_{\mathbb{R}^d}\int_{-\pi}^{\pi} p_1(\omega)p_2(b)\cos(\langle\omega, x\rangle + b)\cos(\langle\omega, y\rangle + b)\,db\,d\omega \\ &= \int_{\mathbb{R}^d} p_1(\omega)\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}\cos(\langle\omega, x\rangle + b)\cos(\langle\omega, y\rangle + b)\,db\right]d\omega. \end{aligned}$$

Воспользовавшись формулой произведения косинусов, получаем:

$$\frac{1}{2\pi}\int_{-\pi}^{\pi}\cos(\langle\omega, x\rangle + b)\cos(\langle\omega, y\rangle + b)\,db = \frac{1}{2}\cos(\langle\omega, x - y\rangle).$$

Тогда

$$k(x, y) = \frac{1}{2}\int_{\mathbb{R}^d} p_1(\omega)\cos\langle\omega, x - y\rangle\,d\omega.$$

Это выражение является преобразованием Фурье гауссовской плотности $p_1(\omega)$. Известно, что преобразование Фурье гауссовской функции также является гауссовской функцией. Тогда

$$\int_{\mathbb{R}^d}\exp\left(-\frac{\|\omega\|^2}{2\sigma^2}\right)e^{i\langle\omega, v\rangle}d\omega = (\sqrt{2\pi}\sigma)^d\exp\left(-\frac{\sigma^2}{2}\|v\|^2\right).$$

В нашем случае $v = x - y$. Таким образом

$$k(x,y) = \frac{1}{2}\frac{1}{(\sqrt{2\pi}\sigma)^d}\left((\sqrt{2\pi}\sigma)^d \exp\left(-\frac{\sigma^2}{2}\|x-y\|_2^2\right)\right)$$
$$= \frac{1}{2}\exp\left(-\frac{\sigma^2}{2}\|x-y\|_2^2\right).$$

Следовательно, ядро $k(x,y)$ является гауссовским. Также из предыдущих замечаний вытекает, что пространство $\mathcal{H}(\phi,p)$ плотно в $L^2(\mathbb{R}^d;\mu)$.

В случае, когда $\alpha(\theta) = p(\theta)\sum_{i=1}^N c_i\phi(x_i;\theta)$ получаем:

$$x \mapsto f(x) = \int \alpha(\theta)\phi(x;\theta)d\theta = \sum_{i=1}^N c_i k(x_i,x) \in \widetilde{\mathcal{H}}(\phi;p),$$

и множество таких функций плотно в $\mathcal{H}(\phi;p)$ в силу основных свойств пространства RKHS. Наконец, можно заключить, что вложение $\mathcal{H}(\phi;p) \subset L^2(\mathbb{R}^d;\mu)$ непрерывно:

$$\|f\|_{L^2(\mathbb{R}^d;\mu)} \le \|f\|_\infty \le \int |\alpha(\theta)|\,d\theta = \int \frac{|\alpha(\theta)|}{p^{1/2}(\theta)}p^{1/2}(\theta)\,d\theta$$
$$\le \left(\int \frac{\alpha^2(\theta)}{p(\theta)}\,d\theta\right)^{1/2} = \|f\|_{\mathcal{H}(\phi,p)}. \qquad \square$$

Рассмотрим случайное множество

$$\widehat{\mathcal{H}}_m(\phi;f) = \left\{x \mapsto \sum_{k=1}^m w_k\phi(x,\theta_k) : |w_k| \le \frac{\|f\|_p}{m}\right\},$$

где $\theta_1,\dots,\theta_m$ — н.о.р. случайные величины с плотностью вероятности p. Покажем, что для любой функции $f \in \widetilde{\mathcal{H}}(\phi,p)$ существует такая случайная функция $\widehat{f}_m \in \widehat{\mathcal{H}}_m(\phi;f)$, что

$$\mathsf{E}(\widehat{f}_m(x) - f(x))^2 \le \frac{\|f\|_p^2}{m}, \quad x \in \mathbb{R}^d.$$

Любой функции $f(x) = \int_\Theta \alpha(\theta)\phi(x;\theta)d\theta$ можно поставить в соответствие слу-

чайную функцию вида $\widehat{f}_m(x)$ вида

$$\widehat{f}_m(x) = \frac{1}{m}\sum_{k=1}^{m}\frac{\alpha(\theta_k)}{p(\theta_k)}\phi(x;\theta_k).$$

По построению, каждый член $\frac{\alpha(\theta_k)}{p(\theta_k)}\phi(x;\theta_k)$ ограничен по модулю, так как $|\phi(x;\theta_k)| \le 1$ и $\sup_\theta |\alpha(\theta)/p(\theta)| = \|f\|_p$. Следовательно, $|w_k| = \frac{1}{m}\left|\frac{\alpha(\theta_k)}{p(\theta_k)}\right| \le \frac{\|f\|_p}{m}$, что удовлетворяет условию для $\widehat{f}_m \in \widehat{\mathcal{H}}_m(\phi; f)$. Тогда

$$\begin{aligned}\mathsf{E}\widehat{f}_m(x) = \mathsf{E}\left[\frac{1}{m}\sum_{k=1}^{m}\frac{\alpha(\theta_k)}{p(\theta_k)}\phi(x;\theta_k)\right] &= \frac{1}{m}\sum_{k=1}^{m}\int_\Theta \frac{\alpha(\theta)}{p(\theta)}\phi(x;\theta)p(\theta)\,d\theta \\ &= \int_\Theta \alpha(\theta)\phi(x;\theta)\,d\theta = f(x).\end{aligned}$$

Теперь вычислим математическое ожидание квадрата разности:

$$\begin{aligned}&\mathsf{E}\left(\frac{1}{m}\sum_{k=1}^{m}\frac{\alpha(\theta_k)}{p(\theta_k)}\phi(x;\theta_k) - f(x)\right)^2 = \frac{1}{m}\mathsf{E}\left(\frac{\alpha(\theta_1)}{p(\theta_1)}\phi(x;\theta_1) - f(x)\right)^2 \\ &\le \frac{1}{m}\mathsf{E}\left(\frac{\alpha(\theta_1)}{p(\theta_1)}\phi(x;\theta_1)\right)^2 = \frac{1}{m}\int\frac{\alpha^2(\theta)}{p(\theta)}\phi^2(x;\theta)\,d\theta \le \frac{1}{m}\int\frac{\alpha^2(\theta)}{p(\theta)}\,d\theta \le \frac{\|f\|_p^2}{m}.\end{aligned} \tag{21}$$

Заметим, что данное утверждение является более простой версией леммы 1 из работы [70].

Применение случайных признаков позволяет переформулировать задачу нелинейной регрессии в терминах линейной регрессии по коэффициентам случайных признаков с сохранением свойства выпуклости функции потерь, что открывает путь для использования эффективных онлайн алгоритмов.

Теорема 1.3. *Рассмотрим вектор случайных признаков* $\Phi(x) = (\phi_1(x), \ldots, \phi_m(x))$*, где* $\phi_i(x) = \phi(x;\theta_i)$*, и построим* $\overline{w}_n \in \mathcal{W} = [-c, c]^m$*,* $c = \|f\|_p/m$ *с помощью алгоритма VAW с ограничениями, как в теореме 1.2. Тогда для* $m \propto \sqrt{n}$

$$\mathsf{E}\frac{1}{2}\int(\langle\Phi(x), \overline{w}_n\rangle - h(x))^2\pi(dx) \le \widetilde{O}\left(\frac{\|f\|_p^2 + \overline{g}^2}{\sqrt{n}}(s + 1/\gamma)\right)$$

$$+ \int (f(x) - h(x))^2 \pi(dx) \qquad (22)$$

для любой $f \in \widetilde{\mathcal{H}}(\phi, p)$.

Доказательство. Пусть $f \in \widetilde{\mathcal{H}}(\phi, p)$. Оценим $\mathsf{E}\frac{1}{2}\int(\langle\Phi(x), \overline{w}_n\rangle - h(x))^2\pi(dx)$, используя функции $f(x)$ и $\widehat{f}_m(x)$:

$$\mathsf{E}\frac{1}{2}\int(\langle\Phi(x), \overline{w}_n\rangle - h(x))^2\pi(dx) \le \mathsf{E}\int(\langle\Phi(x), \overline{w}_n\rangle - f(x))^2\pi(dx)$$
$$+ \int (f(x) - h(x))^2\pi(dx) = 2\mathsf{E}\int(\langle\Phi(x), \overline{w}_n\rangle - \widehat{f}_m(x))^2\pi(dx)$$
$$+ 2\mathsf{E}\int(\widehat{f}_m(x) - f(x))^2\pi(dx) + \int(f(x) - h(x))^2\pi(dx).$$

Для оценки первого и второго членов в правой части применим теорему 1.2 с ограничением $c = \|f\|_p/m$ и воспользуемся неравенством (21). Тогда для $m \propto \sqrt{n}$ получаем, что

$$\mathsf{E}\frac{1}{2}\int(\langle\Phi(x), \overline{w}_n\rangle - h(x))^2\pi(dx) \le \widetilde{O}\left(\frac{(\|f\|_p^2 + \overline{g}^2)m}{n}(s + 1/\gamma)\right)$$
$$+ 2\frac{\|f\|_p^2}{m} + \int(f(x) - h(x))^2\pi(dx). \qquad \square$$

Последний член в (22) представляет собой ошибку аппроксимации. Он характеризует, насколько хорошо истинная функция $h(x)$ может быть представлена функциями из пространства $\widetilde{\mathcal{H}}(\phi, p)$ в смысле $L^2(\pi)$-расстояния. Эта ошибка не зависит от объёма выборки n и количества признаков m.

1.4 Пример: векторная авторегрессия

В этом разделе рассматривается применение теоретических концепций к широко используемой стохастической модели — векторной авторегрессии (VAR) первого порядка. Эта модель представляет собой важный инструмент для анализа многомерных временных рядов, где текущее состояние системы линейно зависит от предыдущего состояния и аддитивного шума [57]. Современные ис-

следования, такие как работы [59] по байесовским VAR и [21] по разреженным VAR-моделям, демонстрируют актуальность этого подхода в эпоху больших данных.

Рассмотрим динамику d-мерного вектора состояния X_t, описываемую линейным стохастическим разностным уравнением:

$$X_t = \kappa Q X_{t-1} + R\xi_t, \quad X_0 = x. \tag{23}$$

Здесь $X_t \in \mathbb{R}^d$ — вектор состояния системы в момент времени t, а $X_0 = x$ — начальное состояние. Случайные возмущения $\xi_t \sim N(0, I_d)$ являются независимыми и одинаково распределёнными d-мерными стандартными нормальными векторами, где I_d — единичная матрица размера $d \times d$. Параметр $\kappa \in (0, 1)$ представляет собой скалярный коэффициент, гарантирующий стационарность процесса. Матрицы Q и R являются квадратными матрицами размера $d \times d$. Матрица Q также является ортогональной, то есть $Q^T Q = QQ^T = I_d$, где Q^T — транспонированная матрица Q.

Последние исследования, включая работу [78] по структурным VAR-моделям для временных сетевых данных, показали важность корректного выбора структуры матриц Q и R при моделировании сложных систем. Особый интерес представляет случай, когда матрица Q обладает разреженной структурой [23], что характерно для многих приложений в нейронауках и экономике.

Для понимания поведения системы (23) необходимо найти явное выражение для X_t. Применяя рекурсивное разложение, получаем:

$$X_t = \kappa^t Q^t x + \sum_{j=1}^{t} \kappa^{t-j} Q^{t-j} R\xi_j.$$

Первый член, $\kappa^t Q^t x$, описывает влияние начального состояния x, которое экспоненциально затухает из-за множителя κ^t при $\kappa \in (0, 1)$. Второй член представляет собой сумму взвешенных случайных возмущений. Поскольку ξ_j являются нормально распределёнными случайными векторами, и сумма линейных преобразований независимых нормальных случайных величин также является нормальной, X_t будет нормально распределённым вектором. Более того, мож-

но переписать сумму в виде произведения матрицы и вектора, что упрощает вычисление ковариационной матрицы:

$$X_t = \kappa^t Q^t x + (\kappa^{t-1}Q^{t-1}R, \kappa^{t-2}Q^{t-2}R, \ldots, R) \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_t \end{pmatrix}.$$

Из этой формы следует, что $X_t \sim N(\mu_t, \Sigma_t)$, где $\mu_t = \kappa^t Q^t x$ и ковариационная матрица

$$\Sigma_t = (\kappa^{t-1}Q^{t-1}R, \ldots, R)(R^T(Q^{t-1})^T, \ldots, R^T)^T = \sum_{j=0}^{t-1} \kappa^{2j} Q^j R R^T (Q^j)^T.$$

Для анализа асимптотического поведения X_t при $t \to \infty$ рассмотрим предел его характеристической функции $\varphi_t(u)$, которая для нормального распределения $N(\mu, \Sigma)$ имеет вид:

$$\lim_{t\to\infty} \varphi_t(u) = \lim_{t\to\infty} e^{i\langle \kappa^t Q^t x, u\rangle - \langle \Sigma_t u, u\rangle/2}.$$

Поскольку $\kappa \in (0, 1)$ и матрица Q ортогональна, член $\kappa^t Q^t x$ стремится к нулевому вектору при $t \to \infty$. Одновременно, ковариационная матрица Σ_t сходится к стационарной ковариационной матрице Σ, определяемой как ряд:

$$\Sigma = \sum_{j=0}^{\infty} \kappa^{2j} Q^j R R^T (Q^j)^T.$$

Этот ряд сходится, так как $\kappa \in (0, 1)$, обеспечивая геометрическое убывание членов. Таким образом, предел характеристической функции равен $e^{-\langle \Sigma u, u\rangle/2}$, что соответствует характеристической функции нормального распределения $N(0, \Sigma)$. Отсюда следует, что X_t слабо сходится к нормальному распределению $N(0, \Sigma)$.

Если начальное состояние X_0 распределено в соответсвии с этим стационарным распределением $\pi = N(0, \Sigma)$, то процесс X_t является строго стационарным. Для проверки этого свойства убедимся, что матрица ковариации Σ

удовлетворяет следующему уравнению:

$$\Sigma = RR^T + \kappa^2 Q\Sigma Q^T. \tag{24}$$

Если $X_{t-1} \sim N(0, \Sigma)$, то для X_t выполняется, что

$$\begin{aligned}
\varphi_{X_t}(u) &= \mathsf{E} e^{i\langle u, \kappa Q X_{t-1} + R\xi_t\rangle} = \mathsf{E} e^{i\langle u, \kappa Q X_{t-1}\rangle} \mathsf{E} e^{i\langle u, R\xi_t\rangle} \\
&= \varphi_{X_{t-1}}(\kappa Q u) e^{-\langle RR^T u, u\rangle/2} \\
&= e^{-\langle \Sigma(\kappa Q u), \kappa Q u\rangle/2 - \langle RR^T u, u\rangle/2} \\
&= e^{-\kappa^2 \langle Q^T \Sigma Q u, u\rangle/2 - \langle RR^T u, u\rangle/2} \\
&= e^{-\langle (\kappa^2 Q^T \Sigma Q + RR^T) u, u\rangle/2}.
\end{aligned}$$

Применяя выражение (24) для Σ, получаем:

$$\varphi_{X_t}(u) = e^{-\langle \Sigma u, u\rangle/2}.$$

Таким образом, $X_t \sim N(0, \Sigma)$, что подтверждает стационарность процесса.

Марковская цепь, описываемая уравнением (23) на $\mathbb{R}^d$, является λ_d-неприводимой, где λ_d обозначает меру Лебега на $\mathbb{R}^d$. Неприводимость означает, что из любого состояния x можно достичь любого борелевского множества положительной меры за конечное число шагов. Это свойство выполняется благодаря наличию шумового члена $R\xi_t$, который является нормально распределённым и, следовательно, имеет плотность, строго положительную на всём $\mathbb{R}^d$.

Из работы [93, предложение 4.3] известно, что любое компактное множество $C \subset \mathbb{R}^d$ является «малым» (small set). Это означает, что существует $\delta > 0$ такое, что для любого $x \in C$ и любого борелевского множества $B \in \mathscr{B}(\mathbb{R}^d)$ вероятность перехода из x в B удовлетворяет неравенству:

$$P(x, B) \geq \delta \lambda_d(B), \quad x \in C,\ B \in \mathscr{B}(\mathbb{R}^d).$$

Это свойство указывает на то, что из любого состояния в компактном множестве цепь Маркова имеет ненулевую вероятность «покрыть» значительную часть пространства, что является ключевым для установления эргодичности.

Для доказательства геометрической эргодичности воспользуемся условием дрейфа с функцией Ляпунова $V(x) = 1 + \|x\|$. Возьмем $C = B_r(0) = \{x : \|x\| \leq r\}$ — шар радиуса r с центром в начале координат. Для любого $\varepsilon > 0$ и достаточно большого r покажем, что

$$\begin{aligned}
\mathsf{E}(V(X_t)|X_{t-1} = x) &= 1 + \mathsf{E}\|\kappa Q x + R\xi_t\| \\
&\leq 1 + \mathsf{E}\|\kappa Q x\| + \mathsf{E}\|R\xi_t\| \\
&= 1 + \kappa\|Qx\| + \mathsf{E}\|R\xi_t\| \\
&= 1 + \kappa\|x\| + \mathsf{E}\|R\xi_t\| \\
&\leq (\kappa + \varepsilon)(1 + \|x\|) = (\kappa + \varepsilon)V(x) \quad \text{для } \|x\| \geq r.
\end{aligned}$$

Последнее неравенство выполняется для достаточно больших значений $\|x\| \geq r$ за счёт выбора ε и использования определения функции $V(x) = 1+\|x\|$. Таким образом, процесс (23) удовлетворяет условию дрейфа

$$\mathsf{E}(V(X_t)|X_{t-1} = x) \leq \lambda V(x) + b\mathbb{1}_C(x),$$

где $\lambda \in (0, 1)$ и $b > 0$ — некоторые константы. В совокупности, данное условие и свойство «малых» множеств обеспечивают геометрическую эргодичность процесса X_t [93, предложение 3.13]. Согласно этому свойству распределение X_t сходится к своему стационарному распределению π по метрике полной вариации с экспоненциальной скоростью

$$d_{\mathrm{TV}}(P_n(x, \cdot), \pi) \leq M_x e^{-\gamma n},$$

где M_x — константа, зависящая от начального состояния x, а $\gamma > 0$ — положительная константа. Следствием геометрической эргодичности также является свойство геометрического β-перемешивания, выражаемое как $\beta_X(n) \leq Be^{-\gamma n}$ [61, теорема 2.1], [12, теорема 3.7].

Рассмотрим функцию $g(x_0, \ldots, x_s) = g(x_s) = e^{-\beta\|x_s\|^2/2}$, которая зависит только от состояния системы в момент времени s. Цель — вычислить условное математическое ожидание этой функции при условии начального состоя-

ния $X_0 = x$:

$$h(x) = \mathsf{E}(g(X_s)|X_0 = x) = \mathsf{E}e^{-\beta\|Y\|^2/2}, \quad \text{где } Y \sim N(\kappa^s Q^s x, \Sigma_s).$$

Поскольку Y является нормально распределённым вектором, для нахождения $h(x)$ требуется вычислить интеграл от плотности нормального распределения, умноженной на экспоненциальный член:

$$h(x) = \int_{\mathbb{R}^d} e^{-\beta\|y\|^2/2} \frac{1}{(2\pi)^{d/2}\sqrt{\det \Sigma_s}} \exp\left(-\frac{1}{2}\langle \Sigma_s^{-1}(y - \kappa^s Q^s x), y - \kappa^s Q^s x\rangle\right) dy.$$

Раскроем степень экспоненты

$$\langle \Sigma_s^{-1}(y - \mu_s), y - \mu_s\rangle = \langle \Sigma_s^{-1} y, y\rangle - 2\langle \Sigma_s^{-1} y, \mu_s\rangle + \langle \Sigma_s^{-1}\mu_s, \mu_s\rangle,$$

где $\mu_s = \kappa^s Q^s x$. Тогда

$$h(x) = \frac{e^{-\frac{1}{2}\langle \Sigma_s^{-1}\mu_s, \mu_s\rangle}}{(2\pi)^{d/2}\sqrt{\det \Sigma_s}} \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}\langle (\beta I_d + \Sigma_s^{-1})y, y\rangle + \langle \Sigma_s^{-1}\mu_s, y\rangle\right) dy.$$

Для дальнейших вычислений воспользуемся следующей формулой:

$$\int_{\mathbb{R}^d} e^{-\frac{1}{2}\langle Az, z\rangle + \langle b, z\rangle}\, dz = \frac{(2\pi)^{d/2}}{\sqrt{\det A}} e^{\frac{1}{2}\langle A^{-1}b, b\rangle}.$$

Подставляя $A = \beta I_d + \Sigma_s^{-1}$ и $b = \Sigma_s^{-1}\mu_s$, получаем, что

$$h(x) = \frac{\exp\left(-\frac{1}{2}\langle \Sigma_s^{-1}\mu_s, \mu_s\rangle\right)}{\sqrt{\det \Sigma_s}\sqrt{\det(\beta I_d + \Sigma_s^{-1})}} \exp\left(\frac{1}{2}\langle (\beta I_d + \Sigma_s^{-1})^{-1}\Sigma_s^{-1}\mu_s, \Sigma_s^{-1}\mu_s\rangle\right). \quad (25)$$

Рассмотрим член в экспоненте, содержащий выражение $(\beta I_d + \Sigma_s^{-1})^{-1}$:

$$\begin{aligned}(\beta I_d + \Sigma_s^{-1})^{-1} &= (\Sigma_s^{-1}(\beta\Sigma_s + I_d))^{-1} \\ &= (I_d + \beta\Sigma_s)^{-1}\Sigma_s.\end{aligned}$$

Тогда для выражения $\langle (\beta I_d + \Sigma_s^{-1})^{-1}\Sigma_s^{-1}\mu_s, \Sigma_s^{-1}\mu_s\rangle$ получаем, что

$$\langle (I_d + \beta\Sigma_s)^{-1}\Sigma_s\Sigma_s^{-1}\mu_s, \Sigma_s^{-1}\mu_s\rangle$$

$$= \langle (I_d + \beta\Sigma_s)^{-1}\mu_s, \Sigma_s^{-1}\mu_s \rangle$$
$$= \langle \Sigma_s^{-1}(I_d + \beta\Sigma_s)^{-1}\mu_s, \mu_s \rangle$$
$$= \langle (\Sigma_s^{-1} - \beta\Sigma_s^{-1}(I_d + \beta\Sigma_s)^{-1})\mu_s, \mu_s \rangle.$$

Теперь упростим выражение с определителем, используя его свойства:

$$\begin{aligned}\sqrt{\det(\beta I_d + \Sigma_s^{-1})} &= \sqrt{\det(\Sigma_s^{-1}(\beta\Sigma_s + I_d))} \\ &= \sqrt{\det(\Sigma_s^{-1})\det(I_d + \beta\Sigma_s)} \\ &= \frac{1}{\sqrt{\det(\Sigma_s)}}\sqrt{\det(I_d + \beta\Sigma_s)}.\end{aligned}$$

Тогда $\frac{\sqrt{\det\Sigma_s}}{\sqrt{\det(\beta I_d + \Sigma_s^{-1})}} = \sqrt{\det(I_d + \beta\Sigma_s)}$. Из подстановки в (25) вышеприведенных упрощений вытекает, что

$$h(x) = \frac{1}{\sqrt{\det(I_d + \beta\Sigma_s)}} \exp\left(-\frac{\kappa^{2s}}{2}\beta\langle (I_d + \beta\Sigma_s)^{-1}Q^s x, Q^s x\rangle\right). \qquad (26)$$

Определим функцию

$$\alpha(w, b) = \frac{1}{\pi\sqrt{\beta}\kappa^s}\frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\left\langle S^{-1}w, w\right\rangle\right)\cos b,$$

где $(w, b) = \theta \in \Theta = \mathbb{R}^d \times (-\pi, \pi)$, а матрица S задается как:

$$S = \beta\kappa^{2s}\left(Q^s(I_d + \beta\Sigma_s)(Q^s)^T\right)^{-1}.$$

Функция $h(x)$ может быть представлена в следующем виде:

$$h(x) = \int_\Theta \alpha(\theta)\phi(x;\theta)\, d\theta,$$

где функция $\phi(x;\theta)$ определена в лемме 1.1. Для вычисления данного интеграла используется известная формула для характеристической функции многомер-

ного нормального распределения:

$$\frac{1}{(2\pi)^{d/2}\sqrt{\det(S)}}\int e^{-\langle S^{-1}y,y\rangle/2}e^{i\langle y,x\rangle}\,dy = e^{-\langle Sx,x\rangle/2}.$$

Применяя эту формулу к рассматриваемому интегралу, получаем:

$$\begin{aligned}\int_\Theta \alpha(\theta)\phi(x;\theta)\,d\theta &= \int_{\mathbb{R}^d}\int_{-\pi}^{\pi}\alpha(\omega,b)\cos(\langle\omega,x\rangle+b)d\omega db\\ &= \frac{1}{\sqrt{\beta}\kappa^s}\frac{1}{(2\pi)^{d/2}}\int_{\mathbb{R}^d}\exp\left(-\frac{1}{2}\left\langle S^{-1}\omega,\omega\right\rangle\right)\cos\langle\omega,x\rangle\,d\omega\\ &= \frac{1}{\sqrt{\beta}\kappa^s}\sqrt{\det(S)}e^{-\langle Sx,x\rangle/2} = h(x),\end{aligned}$$

где последнее равенство соответствует формуле преобразования Фурье. Это равенство выполняется, так как

$$\frac{\det(S)}{\beta\kappa^{2s}} = \det\left[(Q^s(I_d+\beta\Sigma_s)Q^s)^{-1}\right] = \det\left(Q^s(I_d+\beta\Sigma_s)^{-1}Q^s\right) = \frac{1}{\det\left(I_d+\beta\Sigma_s\right)}.$$

Отношение $|\alpha|/p$, где p — плотность, введённая в лемме 1, при $\sigma^2 \geq \beta$ может быть оценено следующим образом:

$$\frac{|\alpha|}{p} = \frac{2}{\sqrt{\beta}\kappa^s}e^{-\langle S^{-1}\omega,\omega\rangle/2}|\cos b|\sigma^d e^{\langle\omega,\omega\rangle/(2\sigma^2)} \leq \frac{2\sigma^d}{\sqrt{\beta}\kappa^s}$$

Это неравенство справедливо, так как

$$\langle S^{-1}\omega,\omega\rangle = \frac{1}{\beta\kappa^{2s}}\langle(I_d+\beta\Sigma_s)Q^s\omega,Q^s\omega\rangle \geq \frac{1}{\beta}\|\omega\|^2 \geq \frac{1}{\sigma^2}\|\omega\|^2.$$

Из этого следует, что $h \in \widetilde{\mathcal{H}}(\phi,p)$, и его норма в этом пространстве ограничена

$$\|h\|_p \leq \frac{2\sigma^d}{\sqrt{\beta}\kappa^s}. \tag{27}$$

Подстановка $f = h$ в (22) в сочетании с неравенством (27) даёт следующую

оценку ожидаемой ошибки:

$$\mathsf{E}\frac{1}{2}\int(\langle\Phi(x),\overline{w}_n\rangle - h(x))^2\pi(dx) \leq \widetilde{O}\left(\frac{\|h\|_p^2+\overline{g}^2}{\sqrt{n}}(s+1/\gamma)\right) = \widetilde{O}\left(\frac{\sigma^{2d}}{\beta\kappa^{2s}\sqrt{n}}(s+1/\gamma)\right). \tag{28}$$

Эта оценка справедлива при условии, что количество случайных признаков m выбирается пропорционально $\sqrt{n}$ (т.е. $m \propto \sqrt{n}$), а сами признаки $\phi(x,\theta_k)$ генерируются путем выборки θ_k из распределения, описанного в лемме 1.1. Кроме того, к этим признакам применяется алгоритм VAW с ограничениями на веса $|w_k| \leq c = \frac{\|h\|_p}{m}$. Исходя из неравенства (27), можем установить $c = \sigma^d/(m\sqrt{\beta}\kappa^s)$.

Заметим, что член κ^{-2s} в оценке (28) экспоненциально растет с увеличением горизонта предсказания s. Отсюда следует, что ошибка аппроксимации может значительно увеличиваться для прогнозов на более длительные временные интервалы, что является естественным ограничением для такого рода задач.

1.4.1 Вычислительные эксперименты

Для эмпирической проверки теоретических выводов были проведены вычислительные эксперименты с использованием комплекса программ на языке Python для модели векторной авторегрессии в одномерном ($d = 1$) и двумерном ($d = 2$) пространствах. Для $d = 1$ была рассмотрена простая авторегрессионная модель:

$$X_t = \kappa X_{t-1} + \xi_t.$$

Для $d = 2$ использовалась модель с матрицей вращения Q:

$$\begin{pmatrix} X_{t,1} \\ X_{t,2} \end{pmatrix} = \kappa \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X_{t,1} \\ X_{t,2} \end{pmatrix} + \begin{pmatrix} \xi_{t,1} \\ \xi_{t,2} \end{pmatrix}.$$

Параметры процессов были выбраны следующим образом: $\kappa = 0.99$, что соответствует медленно затухающему процессу, $\beta = 0.1$ и $\sigma = 1$, и $\theta = 0.3$ для двумерного случая. Число случайных признаков m составляло 80 в одномерном

		$d=1$		$d=2$	
n	Алгоритм	$s=5$	$s=15$	$s=5$	$s=15$
10000	VAW	$2.9\cdot10^{-4}$	$7.9\cdot10^{-4}$	$9.5\cdot10^{-3}$	$1.2\cdot10^{-2}$
	MLP	$6.8\cdot10^{-4}$	$2.5\cdot10^{-3}$	$5.8\cdot10^{-3}$	$1.1\cdot10^{-2}$
50000	VAW	$1.5\cdot10^{-4}$	$4.8\cdot10^{-4}$	$8.3\cdot10^{-3}$	$8.1\cdot10^{-3}$
	MLP	$8.7\cdot10^{-4}$	$1.1\cdot10^{-3}$	$1.2\cdot10^{-3}$	$1.4\cdot10^{-3}$
100000	VAW	$8.0\cdot10^{-5}$	$2.8\cdot10^{-4}$	$7.8\cdot10^{-3}$	$7.1\cdot10^{-3}$
	MLP	$8.1\cdot10^{-4}$	$9.3\cdot10^{-4}$	$4.1\cdot10^{-4}$	$9.2\cdot10^{-4}$

Таблица 1: Среднеквадратичная ошибка, усредненная по 3 экспериментам

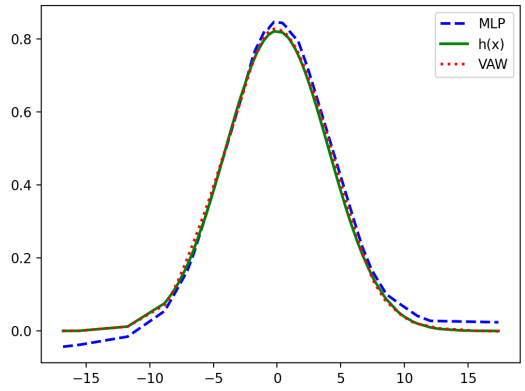
случае и 400 в двумерном.

В экспериментах сравнивались численный метод VAW с использованием случайных признаков Фурье и нейронная сеть типа многослойный перцептрон (MLPRegressor) из библиотеки `scikit-learn`. MLPRegressor использовался с параметрами по умолчанию, включающими один скрытый слой со 100 нейронами. Существенное отличие заключается в том, что алгоритм VAW выполняет всего один проход по данным, в то время как MLPRegressor требует многократных проходов (эпох) для сходимости.

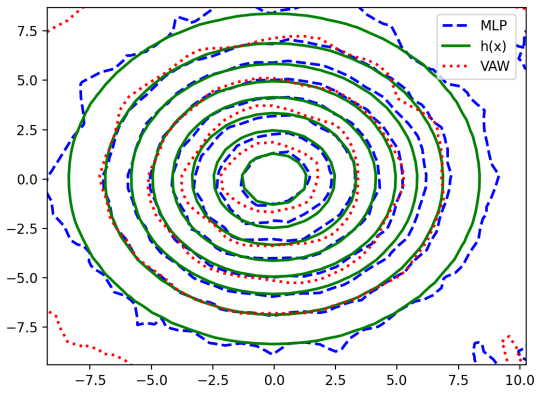
После этапа обучения вычислялась среднеквадратичная ошибка (MSE) на наборе из 1000 тестовых точек x, которые были сгенерированы из стационарного распределения $N(0,\Sigma)$. Для сравнения с истинным значением $h(x)$ использовалось явное аналитическое выражение (26).

Анализ результатов численных экспериментов, представленных в таблице 1, показывает, что с увеличением объёма данных алгоритм VAW демонстрирует сопоставимую или меньшую MSE по сравнению с нейронной сетью MLPRegressor, использующей параметры по умолчанию. При этом VAW требует значительно меньше вычислительных ресурсов, так как выполняется лишь один проход по данным. На рис. 1 представлены график (для $d=1$) и линии уровня (для $d=2$) функции $h(x)$, восстановленной алгоритмами VAW и MLPRegressor при объёме выборки $n=100000$, шаге$s=5$ и количестве признаков $m=80$ (для $d=1$) и $m=400$ (для $d=2$).

Комплекс программ для воспроизведения экспериментов находится в от-



(a) График h и его аппроксимаций ($d = 1$)



(b) Линии уровня h и их аппроксимации ($d = 2$)

Рис. 1: Визуализация функции h

крытом доступе на платформе GitHub [1]. В приложении А.1. приводится листинг программы с реализацией алгоритма VAW с использованием случайных признаков Фурье.

1.5 Заключение к главе 1

В этой главе было проведено исследование применения численного метода Вовка-Азури-Вармута (VAW) в сочетании со случайными признаками Фурье для моделирования динамических процессов с марковской зависимостью. На основе теоретических результатов из работы [2], посвящённых обобщающей способности онлайн алгоритмов на зависимых данных, получены строгие верхние оценки ожидаемой квадратичной ошибки. Эти оценки обеспечивают теоретические гарантии эффективности предложенного метода в условиях временной зависимости данных.

Рассматривалась стандартная постановка задачи обучения с учителем с липшицевой функцией потерь и n независимыми одинаково распределёнными примерами. Согласно фундаментальному результату [70, теорема 1], для достижения ошибки порядка $O(n^{-1/2})$ требуется $O(n)$ случайных признаков. Показано, что в случае квадратичной функции потерь достаточно $\widetilde{O}(\sqrt{n})$ признаков

[1] **Gurtovaya, O. V.** Vovk-Azoury-Warmuth algorithm and random Fourier features for a regression problem with Markovian data [Электронный ресурс] / O. V. Gurtovaya — 2025. — URL: `https://github.com/O-Gurt/VAW_Markov` (дата обращения: 21.08.2025).

для аналогичной точности [74, 15], даже для более широкого класса гипотез. В настоящем исследовании продемонстрировано, что аналогичный результат справедлив и для зависимых данных. Был использован подход из работы [2], где доказано, что алгоритм, обладающий устойчивостью и малым сожалением, обеспечивает низкую ошибку оценивания относительно инвариантного распределения при выполнении условий перемешивания.

Вычислительные эксперименты подтвердили, что при достаточно больших объёмах обучающих данных модель на основе случайных признаков Фурье, обученная с помощью VAW, конкурирует по точности с нейронными сетями, обученными традиционными методами. Стоит отметить, что при обучении алгоритм VAW проходит по данным всего один раз, что существенно снижает вычислительные затраты по сравнению с итерационными подходами, требующими многократных эпох обучения.

Дополнительное применение оптимизационных техник, таких как многопроходное обучение с ранней остановкой, может улучшить аппроксимацию целевой функции, избегая переобучения и усиливая обобщающую способность.

Глава 1 основана на материалах, опубликованных в работе [73].

2 Реализация двух- и трёхуровневого численных методов Вовка-Азури-Вармута с использованием многоядерного подхода

2.1 Переход от одноядерного к многоядерному подходу в онлайн оптимизации

Ядерные методы [81, 43] предоставляют мощный инструмент для анализа сложных нелинейных зависимостей, значительно расширяя возможности традиционных линейных моделей. Эти методы обладают высокой выразительной способностью, что подтверждается их свойством универсальности [88]. Более того, они позволяют эффективно применять аппарат выпуклого анализа, гарантируя достижение глобально оптимальных решений.

Однако, несмотря на эти преимущества, одноядерные методы сталкиваются с двумя серьёзными недостатками. Первый и наиболее очевидный — высокая вычислительная сложность. Она растет пропорционально T^3, где T — число обучающих примеров в стандартной задаче пакетного обучения с учителем. Это делает их трудноприменимыми для работы с большими объёмами данных. В онлайн оптимизации, где данные поступают последовательно, эта проблема лишь усугубляется: при применении алгоритма градиентного спуска в RKHS [49] каждая итерация неизбежно приводит к усложнению линейной комбинации ядер из-за постоянного добавления новых опорных векторов в формируемый словарь. Данное явление получило название «проклятие кернелизации» [100], поскольку оно значительно увеличивает вычислительные затраты и требования к памяти по мере обучения модели.

Для борьбы с этим «проклятием» разработано множество техник, условно разделяющихся на две большие категории [44]: стратегии поддержания бюджета и стратегии функциональной аппроксимации. Первые направлены на ограничение размера словаря опорных векторов путём удаления наименее значимых векторов, их проекции в низкоразмерные пространства или объединения схожих векторов, что позволяет сохранить обучаемость модели при сокращении

вычислительных ресурсов. Вторые стремятся аппроксимировать исходное ядро более простыми функциями, что позволяет избежать хранения всех опорных векторов.

Примечательно, что аналогичные адаптивные численные методы фильтрации на основе ядер также нашли широкое применение в сфере обработки сигналов [86, 96]. Например, в задачах устранения искажений сигнала при передаче по каналу или подавления шума в аудио- или видеосигналах, где постоянно поступают новые данные, применение таких алгоритмов позволяет системе непрерывно подстраиваться под изменяющиеся условия, сохраняя при этом приемлемую вычислительную нагрузку.

Второй, не менее критичный недостаток одноядерных методов — это необходимость выбора подходящего ядра. Этот выбор нетривиален и оказывает решающее влияние на качество и точность конечных результатов. Различные типы данных или конкретные задачи могут требовать специфических ядер, и отсутствие априорных знаний часто приводит к трудоёмкому процессу подбора. Неправильно выбранное ядро может значительно ухудшить обучаемость модели, делая её неэффективной. Для решения этой проблемы были предложены многоядерные методы [50, 7]. Согласно обзору [38], эти методы призваны повысить точность и обобщающую способность моделей, поскольку они позволяют комбинировать преимущества различных ядер, которые могут быть оптимальны для разных аспектов или модальностей данных. Они предполагают выбор оптимальной комбинации ядер из обширного, заранее определённого набора, что позволяет модели более гибко адаптироваться к данным и преодолевать ограничения, связанные с выбором единственного ядра.

При многоядерном подходе происходит построение ансамбля математических моделей, где итоговая прогнозирующая функция является взвешенной суммой более простых моделей, каждая из которых порождается своим ядром. Это позволяет строить более гибкие и точные модели данных сложной структуры. Многоядерные подходы активно исследовались в контексте выпуклой онлайн оптимизации [68, 69]. Например, в работах [79] и [84] онлайн градиентный спуск применялся к векторам случайных признаков, соответствующим каждому ядру, для формирования экспертных стратегий. Веса w_t для каждой такой

экспертной стратегии итеративно обновлялись по формуле:

$$w_{t+1} = w_t - \eta_t \nabla L(w_t; x_t, y_t),$$

где w_t — это веса модели на текущем шаге t, η_t — скорость обучения (шаг градиента), а $\nabla L(w_t; x_t, y_t)$ — градиент функции потерь L по весам w_t для текущего обучающего примера x_t с меткой y_t. Этот процесс позволял каждой экспертной стратегии специализироваться на определённом аспекте данных.

В работе [79] численный метод многоядерной регрессии был успешно применён для математического моделирования крупномасштабного онлайн прогнозирования временных рядов. Верхняя оценка сожаления метода составляет $O(\sqrt{T \log p})$, где p — количество ядер в ансамбле. Этот результат гарантирует, что алгоритм работает почти так же хорошо, как и лучшее ядро в ансамбле, что демонстрирует его высокую точность и устойчивость даже на больших объёмах данных. Результаты показали, что комбинация нескольких ядер позволяет более эффективно улавливать сложную динамику временных рядов по сравнению с использованием одного ядра. Аналогично, в работе [84] исследовалось онлайн оптимизация с несколькими ядрами на основе случайных признаков в средах с неизвестной динамикой. В частности, авторы предложили масштабируемую схему Raker (Random Feature-based Multi-kernel Learning) для статических сред и адаптивную схему AdaRaker для динамических сред. Эти алгоритмы позволили моделям значительно лучше адаптироваться к меняющимся условиям и динамике данных, показывая надёжные результаты там, где одноядерные модели были бы менее эффективны. Предсказания этих стратегий затем эффективно объединялись либо с помощью правила обновления экспоненциальных весов, либо напрямую через алгоритм онлайн градиентного спуска, что позволяло получать более надёжные и точные результаты.

Развивая эти подходы, в более поздней работе [35] было представлено онлайн многоядерное моделирование с использованием графов (Graph-aided online multi-kernel learning). В данной работе был разработан алгоритм OMKL-GF, который решает математическую задачу построения модели, дополнительно используя информацию о связях между входными признаками, представленными в виде графа. Это позволяет алгоритму OMKL-GF лучше адаптироваться

к сложным структурам данных, особенно когда взаимосвязи между признаками имеют явное графическое представление. Благодаря учёту такой графической структуры, OMKL-GF способен более эффективно обучаться в онлайн режиме, динамически настраивая веса для различных ядер и улучшая эффективность модели, особенно в условиях, где данные обладают сложными зависимостями.

В данной главе представлен теоретический анализ и обоснование численного метода, основанного на двухуровневом алгоритме Вовка-Азури-Вармута VAW^2 для решения задачи онлайн линейной регрессии с квадратичной функцией потерь. Предлагаемый метод сочетает функциональную аппроксимацию ядер через случайные признаки Фурье с алгоритмом VAW.

Вычислительная схема алгоритма реализует двухуровневую архитектуру. На первом уровне для каждого ядра из заданного ансамбля осуществляется генерация случайных признаков Фурье, на основе которых формируются экспертные стратегии с использованием метода VAW. На втором уровне также применяется алгоритм VAW, но уже для агрегирования полученных экспертных прогнозов.

Теоретический анализ сопровождается серией численных экспериментов, направленных на сравнительную оценку эффективности VAW^2 относительно современных методов онлайн многоядерного моделирования, включая алгоритм Raker [84] и граф-ориентированный метод OMKL-GF [35].

Также предлагается модификация численного метода S-VAW^2 (Scaled Vovk-Azoury-Warmuth), объединяющая стратегию обучения VAW^2 с техниками нормирования данных, что позволяет ему справляться с разнородностью данных при сохранении вычислительной эффективности. Проведен сравнительный анализ алгоритма S-VAW^2 с системой автоматизированного построения моделей AutoGluon-Tabular на наборе эталонных регрессионных задач.

2.2 Двухуровневый численный метод Вовка-Азури-Вармута в контексте многоядерного подхода

В этом разделе также рассматриваются трансляционно-инвариантные ядра $k(x, y) = \kappa(x - y)$, для которых справедливо выполнение теоремы Бохнера

$$\kappa(z) = \int_{\mathbb{R}^d} e^{i\langle \omega, z\rangle} \Lambda(d\omega).$$

Для наших целей достаточно предположить, что Λ абсолютно непрерывна относительно меры Лебега

$$\kappa(z) = \int_{\mathbb{R}^d} e^{i\langle \omega, z\rangle} q(\omega)\, d\omega = \int_{\mathbb{R}^d} q(\omega) \cos\langle \omega, z\rangle\, d\omega,$$

где q — функция плотности вероятности. В частности, для гауссовских ядер

$$k(x, y) = e^{-\|x-y\|_2^2/(2\sigma^2)}, \quad q(\omega) = \left(\frac{\sigma}{\sqrt{2\pi}}\right)^d e^{-\sigma^2\|\omega\|_2^2/2}, \tag{29}$$

для лапласовских ядер

$$k(x, y) = e^{-\|x-y\|_1/\sigma}, \quad q(\omega) = \frac{\sigma^d}{\pi^d} \prod_{j=1}^{d} \frac{1}{1 + \sigma^2\omega_j^2}. \tag{30}$$

Видно, что здесь q является произведением гауссовских распределений и распределений Коши соответственно [68]. Для таких ядер формула (18) верна при $\Theta = \mathbb{R}^d \times [0, 2\pi]$, $\theta = (\omega, b)$, где

$$p(\theta) = q(\omega)r(b), \quad r(b) = 1/(2\pi),$$
$$\phi(x; \theta) = \sqrt{2}\cos(\langle \omega, x\rangle + b).$$

Тогда

$$\int_\Theta p(\theta)\phi(x; \theta)\phi(y; \theta)\, d\theta = \frac{1}{2\pi} \int_0^{2\pi} \int_{\mathbb{R}^d} 2\cos(\langle \omega, x\rangle + b)\cos(\langle \omega, y\rangle + b) q(\omega) d\omega db$$

$$= \int_{\mathbb{R}^d} \cos\langle \omega, x-y \rangle q(\omega) d\omega = \kappa(x-y) = k(x,y).$$

Рассмотрим вектор случайных признаков Фурье:

$$\Phi(x) = (\phi(x,\theta_k))_{k=1}^m = (\sqrt{2}\cos(\langle \omega_k, x\rangle + b_k))_{k=1}^m,$$

сгенерированный из распределений p. Здесь $\omega_k \sim q$, $b_k \sim U(0, 2\pi)$ — независимые и одинаково распределённые случайные величины. Обозначим через E_θ математическое ожидание относительно совместного распределения $\theta_1, \ldots, \theta_m$. Следующий результат показывает, что любой элемент из $\mathcal{H}$, определённый формулой (17), может быть аппроксимирован линейной комбинацией случайных признаков Фурье.

Лемма 2.1. *Для любого $f = \int \gamma(\theta)\phi(\cdot,\theta)\, d\theta \in \mathcal{H}$ положим*

$$\widehat{w} = \frac{1}{m}\left(\frac{\gamma(\theta_1)}{p(\theta_1)}, \ldots, \frac{\gamma(\theta_m)}{p(\theta_m)}\right),$$

где $\theta_i \sim p$ — независимые и одинаково распределённые случайные величины. Тогда

$$\mathsf{E}_\theta(\langle \widehat{w}, \Phi_\theta(x)\rangle - f(x))^2 \le 2\frac{\|f\|_{\mathcal{H}}^2}{m}, \qquad \mathsf{E}_\theta\|\widehat{w}\|_2^2 = \frac{\|f\|_{\mathcal{H}}^2}{m}. \tag{31}$$

Доказательство. Заметим, что стохастическая оценка $\langle \widehat{w}, \Phi_\theta(x)\rangle$ для $f(x)$ является несмещенной:

$$\begin{aligned}
\mathsf{E}_\theta\langle \widehat{w}, \Phi_\theta(x)\rangle &= \frac{1}{m}\sum_{i=1}^m \mathsf{E}_{\theta_i}\left(\frac{\gamma(\theta_i)}{p(\theta_i)}\phi(x,\theta_i)\right) \\
&= \frac{1}{m}\sum_{i=1}^m \int \frac{\gamma(\theta_i)}{p(\theta_i)}\phi(x,\theta_i)p(\theta_i)d\theta_i \\
&= \frac{1}{m}\sum_{i=1}^m \int \gamma(\theta_i)\phi(x,\theta_i)d\theta_i \\
&= \frac{1}{m}\sum_{i=1}^m f(x) = f(x).
\end{aligned} \tag{32}$$

Вычислим дисперсию этой оценки:

$$\mathsf{E}_\theta\left(\frac{1}{m}\sum_{k=1}^{m}\frac{\gamma(\theta_k)}{p(\theta_k)}\phi(x;\theta_k)-f(x)\right)^2=\frac{1}{m}\mathsf{E}_{\theta_1}\left(\frac{\gamma(\theta_1)}{p(\theta_1)}\phi(x;\theta_1)-f(x)\right)^2$$
$$\leq\frac{1}{m}\mathsf{E}_{\theta_1}\left(\frac{\gamma(\theta_1)}{p(\theta_1)}\phi(x;\theta_1)\right)^2=\frac{1}{m}\int\frac{\gamma^2(\theta_1)}{p^2(\theta_1)}\phi^2(x;\theta_1)p(\theta_1)\,d\theta_1$$
$$=\frac{1}{m}\int\frac{\gamma^2(\theta_1)}{p(\theta_1)}\phi^2(x;\theta_1)\,d\theta_1\leq\frac{2}{m}\int\frac{\gamma^2(\theta_1)}{p(\theta_1)}\,d\theta_1\leq 2\frac{\|f\|^2_{\mathcal{H}}}{m}.$$

Из определения оценки $\widehat{w}$ следует, что

$$\mathsf{E}_\theta\|\widehat{w}\|_2^2=\frac{1}{m^2}\sum_{i=1}^{m}\mathsf{E}_{\theta_i}\left(\left(\frac{\gamma(\theta_i)}{p(\theta_i)}\right)^2\right)=\frac{1}{m^2}\sum_{i=1}^{m}\int\frac{\gamma^2(\theta_i)}{p^2(\theta_i)}p(\theta_i)d\theta_i$$
$$=\frac{1}{m}\int\frac{\gamma^2(\theta_1)}{p(\theta_1)}d\theta_1=\frac{\|f\|^2_{\mathcal{H}}}{m}.\qquad\square$$

Рассмотрим произвольную последовательность пар $(x_t, y_t) \in \mathbb{R}^d \times \mathbb{R}$. Предположим, что зависимость между признаками x_t и целевыми значениями y_t может быть описана некоторой функцией $f \in \mathcal{H}$. Тогда можно сформулировать следующую задачу наименьших квадратов:

$$\sum_{t=1}^{T}(y_t-f(x_t))^2\to\min_{f\in\mathcal{H}}. \tag{33}$$

Согласно лемме 2.1, данная задача может быть представлена в виде:

$$\sum_{t=1}^{T}(y_t-\langle w,\Phi_\theta(x_t)\rangle)^2\to\min_{w\in\mathbb{R}^m}.$$

В рамках онлайн оптимизации цель состоит в построении последовательности векторов весов w_t, обеспечивающую «малые» ожидаемые кумулятивные потери:

$$\mathsf{E}_\theta\sum_{t=1}^{T}(y_t-\langle w_t,\Phi_\theta(x_t)\rangle)^2.$$

При этом качество решения оценивается путем сравнения с потерями (33) для произвольной функции $f \in \mathcal{H}$. Каждый вектор w_t формируется на основе

предыстории наблюдений: $w_t = w_t(\Phi_\theta(x_1), \ldots, \Phi_\theta(x_t), y_1, \ldots, y_{t-1})$. Будем считать, что на каждом шаге t признаки x_t известны до того, как необходимо сделать прогноз целевой переменной y_t. Тогда для нахождения последовательности w_t можно воспользоваться численным методом VAW (6).

Введём дополнительное предположение о равномерной ограниченности целевых переменных: $|y_t| \leq Y$. В таком случае, если $\|\Phi_\theta(x_t)\|_2 \leq \rho$ [17, теорема 11.8], [64, теорема 7.34], то сожаление для алгоритма VAW удовлетворяет оценке:

$$R_T(w) \leq \frac{\lambda}{2}\|w\|_2^2 + \frac{mY^2}{2} \ln\left(1 + \frac{\rho^2 T}{\lambda m}\right). \tag{34}$$

В частности, для случая $\rho = \sqrt{2m}$

$$R_T(w) \leq \frac{\lambda}{2}\|w\|_2^2 + \frac{mY^2}{2} \ln\left(1 + \frac{2T}{\lambda}\right). \tag{35}$$

Для обхода проблемы выбора оптимального ядра для конкретной задачи рассмотрим набор из N гильбертовых пространств $\{\mathcal{H}_j\}_{j=1}^N$, каждое из которых ассоциировано со своим воспроизводящим ядром κ_j (RKHS). Введем пространство

$$\mathcal{H} = \mathcal{H}_1 + \cdots + \mathcal{H}_N = \left\{ \sum_{j=1}^N f_j \mid f_j \in \mathcal{H}_j \right\}$$

Норма в этом пространстве задаётся следующим образом:

$$\|f\|_{\mathcal{H}}^2 = \min \left\{ \sum_{j=1}^N \|f_j\|_{\mathcal{H}_j}^2 \mid f = \sum_{j=1}^N f_j \right\}$$

Как показано в монографии [99, предложение 12.27], такое пространство $\mathcal{H}$ само является RKHS с ядром, равным сумме исходных ядер $k(x, x') = \sum_{j=1}^N k_j(x, x')$. Эта конструкция позволяет комбинировать различные типы ядер, сохраняя при этом свойства RKHS.

Лемма 2.2. *Для $f \in \mathcal{H} = \mathcal{H}_1 + \cdots + \mathcal{H}_N$ возьмем $f_j = \int_\Theta \gamma_j(\theta)\phi_j(x;\theta)d\theta \in \mathcal{H}_j$*

так, что

$$f = \sum_{j=1}^{N} f_j, \qquad \|f\|_{\mathcal{H}}^2 = \sum_{j=1}^{N} \|f_j\|_{\mathcal{H}_j}^2.$$

Для каждого ядра k_j определим

$$\widehat{w}_j = \frac{1}{m}\left(\frac{\gamma_j(\theta_{j1})}{p_j(\theta_{j1})}, \ldots, \frac{\gamma_j(\theta_{jm})}{p_j(\theta_{jm})}\right)$$

по аналогии с леммой 2.1. Здесь $\theta_{jk} \sim p_j$, $k = 1, \ldots, m$ являются независимыми и одинаково распределёнными случайными величинами для каждого $j = 1, \ldots, N$. Пусть $|y| \le Y$. Тогда

$$\mathsf{E}_\theta \left(\sum_{j=1}^{N} \langle \widehat{w}_j, \Phi_{\theta_j}(x) \rangle - y \right)^2 \le 2\frac{N}{m}\|f\|_{\mathcal{H}}^2 + \left(f(x) - y\right)^2, \tag{36}$$

где $\Phi_{\theta_j}(x) = (\phi(x, \theta_{jk}))_{k=1}^m$.

Доказательство. Заметим, что оценка $\langle \widehat{w}_j, \Phi_{\theta_j}(x) \rangle$ для значения функции $f_j(x)$ является несмещённой: см. (32). Тогда

$$\begin{aligned}
&\mathsf{E}_\theta \left(\sum_{j=1}^{N} \langle \widehat{w}_j, \Phi_{\theta_j}(x) \rangle - y \right)^2 - \left(\sum_{j=1}^{N} f_j(x) - y \right)^2 \\
&= \mathsf{E}_\theta \left(\sum_{j=1}^{N} \langle \widehat{w}_j, \Phi_{\theta_j}(x) \rangle \right)^2 - \left(\sum_{j=1}^{N} f_j(x) \right)^2 \\
&= \mathsf{E}_\theta \left(\sum_{j=1}^{N} \left(\langle \widehat{w}_j, \Phi_{\theta_j}(x) \rangle - f_j(x) \right) \right)^2 .
\end{aligned} \tag{37}$$

С помощью известного неравенства $(\sum_{i=1}^N a_i)^2 \le N \sum_{i=1}^N a_i^2$ и леммы 2.1 получаем верхнюю границу для последнего выражения:

$$\begin{aligned}
\mathsf{E}_\theta \left(\sum_{j=1}^{N} \left(\langle \widehat{w}_j, \Phi_{\theta_j}(x) \rangle - f_j(x) \right) \right)^2 &\le N \sum_{j=1}^{N} \mathsf{E}_\theta \left(\langle \widehat{w}_j, \Phi_{\theta_j}(x) \rangle - f_j(x) \right)^2 \\
&\le 2\frac{N}{m} \sum_{j=1}^{N} \|f_j\|_{\mathcal{H}_j}^2 = 2\frac{N}{m}\|f\|_{\mathcal{H}}^2.
\end{aligned} \tag{38}$$

Непосредственное применение неравенств (37) и (38) доказывает справедливость исходного утверждения (36). □

Применим численный метод VAW к последовательности данных $(\Phi_\theta(x_t), y_t)$. Здесь $\Phi_\theta(x)$ представляет собой конкатенированный вектор случайных признаков:

$$\Phi_\theta(x) = (\Phi_{\theta_1}(x), \ldots, \Phi_{\theta_N}(x)), \quad \Phi_{\theta_j}(x) = (\phi_j(x, \theta_{jk}))_{k=1}^m. \tag{39}$$

Такой подход позволяет объединить случайные признаки Φ_{θ_j}, ассоциированные с каждым индивидуальным ядром k_j, в единый Nm-мерный вектор. Таким образом, алгоритм VAW обрабатывает информацию, полученную из множества ядер, в унифицированной форме.

Теорема 2.1. *Пусть $w_t = (w_{t,1}, \ldots, w_{t,Nm}) \in \mathbb{R}^{Nm}$ — вектор весов, полученный при помощи алгоритмов VAW с использованием последовательности данных $(\Phi_\theta(x_t), y_t)$. Тогда*

$$\frac{1}{2}\mathsf{E}_\theta \sum_{t=1}^T (\langle w_t, \Phi_\theta(x_t)\rangle - y_t)^2 \le \frac{1}{2}\sum_{t=1}^T (f(x_t) - y_t)^2 + \left(\frac{\lambda}{2} + NT\right)\frac{\|f\|_{\mathcal{H}}^2}{m} + \frac{NmY^2}{2}\ln\left(1 + \frac{2T}{\lambda}\right) \tag{40}$$

Данная оценка справедлива для любой функции $f \in \mathcal{H}$. Более того, при $T \to +\infty$

$$\frac{1}{2}\mathsf{E}_\theta \sum_{t=1}^T (\langle w_t, \Phi_\theta(x_t)\rangle - y_t)^2 \le \frac{1}{2}\inf_{f \in B_R(\mathcal{H})} \sum_{t=1}^T (f(x_t) - y_t)^2 + O\left(N(R^2 + Y^2 \ln T)\sqrt{T}\right), \quad \textit{если} \quad m \propto \sqrt{T}. \tag{41}$$

Доказательство. Пусть $f \in \mathcal{H}$ и $R_T^{\mathrm{VAW}}(w_1, \ldots, w_N)$ — сожаление алгоритма VAW относительно фиксированного вектора $(w_1, \ldots, w_N) \in (\mathbb{R}^m)^N$. Выберем

f_j и $\widehat{w}_j$ в соответствии с условиями леммы 2.2. Тогда

$$\frac{1}{2}\sum_{t=1}^{T}(\langle w_t, \Phi_\theta(x_t)\rangle - y_t)^2 = R_T^{\text{VAW}}(\widehat{w}_1, \ldots, \widehat{w}_N) + \frac{1}{2}\sum_{t=1}^{T}\left(\sum_{j=1}^{N}\langle \widehat{w}_j, \Phi_{\theta_j}(x_t)\rangle - y_t\right)^2.$$

Используя результаты (35) и леммы 2.1, получаем, что

$$\begin{aligned}\mathsf{E}_\theta R_T^{\text{VAW}}(\widehat{w}_1, \ldots, \widehat{w}_N) &\le \frac{\lambda}{2}\sum_{j=1}^{N}\mathsf{E}_\theta\|\widehat{w}_j\|_2^2 + \frac{NmY^2}{2}\ln\left(1+\frac{2T}{\lambda}\right)\\ &\le \frac{\lambda}{2}\frac{\|f\|_{\mathcal{H}}^2}{m} + \frac{NmY^2}{2}\ln\left(1+\frac{2T}{\lambda}\right). \qquad (42)\end{aligned}$$

Согласно лемме 2.2,

$$\frac{1}{2}\sum_{t=1}^{T}\mathsf{E}_\theta\left(\sum_{j=1}^{N}\langle \widehat{w}_j, \Phi_{\theta_j}(x_t)\rangle - y_t\right)^2 \le T\frac{N}{m}\|f\|_{\mathcal{H}}^2 + \frac{1}{2}\sum_{t=1}^{T}\left(f(x_t)-y_t\right)^2.$$

Объединение оценок (42) и последнего неравенства непосредственно доказывает утверждение (40). В свою очередь, соотношение (41) следует из этого результата при асимптотическом анализе. □

В предположении $m \ge d$, анализ показывает, что временная и пространственная сложности предложенного алгоритма составляют $O(N^2m^2)$ на каждую итерацию (см. (7), (8)). Для существенного уменьшения вычислительных затрат предлагается использовать двухуровневую процедуру:

- **На первом уровне** генерируются N m-мерных векторов случайных признаков, специфичных для каждого ядра k_i. Затем к каждой из N последовательностей данных $(\Phi_{\theta_j}(x_t), y_t)$ применяется индивидуальный алгоритм VAW. Эти VAW-алгоритмы выступают в роли экспертов, каждый из которых делает свои предсказания.

- **На втором уровне** предсказания, полученные от этих экспертных алгоритмов, рассматриваются как мнения экспертов. Эти мнения затем комбинируются с помощью мета-алгоритма, который агрегирует их для получения итогового предсказания.

В качестве мета-алгоритма, агрегирующего мнения экспертов, возьмем алгоритм VAW. При условии $m \geq \max\{d, N\}$) общая временная и пространственная сложности на итерацию значительно снижаются до $O(Nm^2)$. Эта оценка полностью определяется сложностью алгоритмов-экспертов, поскольку вычислительный вклад мета-алгоритма имеет меньший порядок. Важно отметить, что обоснование оценок потерь, представленных в теореме 2.2, не требует условия ограниченности выходных значений экспертов $\langle w_{t,j}, \Phi_{\theta_j}(x_t)\rangle$.
Вся процедура описана в алгоритме 1, который называется VAW^2 [72].

Алгоритм 1 Стратегия обучения VAW^2

Вход: набор данных $(x_t, y_t)_{t=1}^T$
Инициализация: семейство ядер $\{\kappa_j\}_{j=1}^N$, размерность признаков m, параметры регуляризации λ, λ'
1. **Для** $j = 1$ **до** N **делать**
2. Сгенерировать $\omega_{jk} \sim q_j$, $b_{jk} \sim U(0, 2\pi)$ для $k = 1, \ldots, m$
3. Определить признаки Фурье: $\Phi_{\theta_j}(x) = \sqrt{2}\,(\cos(\langle \omega_{jk}, x\rangle + b_{jk}))_{k=1}^m$
4. **Конец цикла**
5. **Для** $t = 1$ **до** T **делать**
6. **Для** $j = 1$ **до** N **делать**
7. $w_{t,j} = \text{argmin}_{w_j} \left\{ \frac{\lambda}{2}\|w_j\|^2 + \frac{1}{2}\sum_{i=1}^{t-1}(\langle \Phi_{\theta_j}(x_i), w_j\rangle - y_i)^2 + \frac{1}{2}\langle \Phi_{\theta_j}(x_t), w_j\rangle^2 \right\}$
8. $z_{t,j} = \langle w_{t,j}, \Phi_{\theta_j}(x_t)\rangle$
9. **Конец цикла**
10. $v_t = \text{argmin}_v \left\{ \frac{\lambda'}{2}\|v\|^2 + \frac{1}{2}\sum_{i=1}^{t-1}(\langle z_i, v\rangle - y_i)^2 + \frac{1}{2}\langle z_t, v\rangle^2 \right\}$
11. $\hat{y}_t = \langle v_t, z_t\rangle$
12. **Конец цикла**

Таблица 2: Применение алгоритма VAW^2

Теорема 2.2. *Пусть $w_{t,j} \in \mathbb{R}^m$ представляет собой вектор весов, генерируемый алгоритмами VAW, примененными к последовательности $(\Phi_{\theta_j}(x_t), y_t)$ для каждого эксперта j. Вектор $\alpha_t \in \mathbb{R}^N$ также генерируется алгоритмом VAW,*

но уже примененным к вектору экспертных предсказаний z_t:

$$z_t = (\langle w_{t,1}, \Phi_{\theta_1}(x_t)\rangle, \ldots, \langle w_{t,N}, \Phi_{\theta_N}(x_t)\rangle).$$

Тогда

$$\begin{aligned}\frac{1}{2}\mathsf{E}_\theta \sum_{t=1}^{T}(\langle \alpha_t, z_t\rangle - y_t)^2 \le \frac{1}{2}\sum_{t=1}^{T}(y_t - f_j(x_t))^2 + \frac{\lambda}{2} \\ + \left(\frac{\lambda}{2} + T\right)\frac{\|f_j\|^2_{\mathcal{H}_j}}{m} + \frac{mY^2}{2}\ln\left(1 + \frac{2T}{\lambda}\right) \\ + \frac{NY^2}{2}\ln\left(1 + \frac{NY^2}{\lambda}\left(2T(T+1) + 2mT\ln\left(1 + \frac{2T}{\lambda}\right)\right)\right).\end{aligned} \tag{43}$$

Это справедливо для любой функции $f_j \in \mathcal{H}_j$ и любого $j = 1, \ldots, N$. При $T \to +\infty$

$$\begin{aligned}\frac{1}{2}\mathsf{E}_\theta \sum_{t=1}^{T}(\langle \alpha_t, z_t\rangle - y_t)^2 \le \frac{1}{2}\min_{1\le j\le N}\inf_{f_j \in B_R(\mathcal{H}_j)}\sum_{t=1}^{T}(y_t - f_j(x_t))^2 \\ + O\left((R^2 + Y^2\ln T)\sqrt{T}\right), \quad \textit{если} \quad m \propto \sqrt{T}.\end{aligned} \tag{44}$$

Доказательство. Возьмем функции f_j и векторы весов $\widehat{w}_j$ в соответствии с леммой 2.2. Обозначим через $R_T^{\mathrm{VAW}}(\delta)$ сожаление алгоритма VAW, примененного к последовательности (z_t, y_t), и через $R_T^{\mathrm{VAW}}(\widehat{w}_j)$ — сожаление алгоритма VAW, примененного к $(\Phi_{\theta_j}(x_t), y_t)$. Для любых неотрицательных весов δ_i таких, что $\sum_{i=1}^N \delta_i = 1$, выполняется:

$$\begin{aligned}\frac{1}{2}\sum_{t=1}^{T}(\langle \alpha_t, z_t\rangle - y_t)^2 &= R_T^{\mathrm{VAW}}(\delta) + \frac{1}{2}\sum_{t=1}^{T}(\langle \delta, z_t\rangle - y_t)^2 \\ &\le R_T^{\mathrm{VAW}}(\delta) + \frac{1}{2}\sum_{t=1}^{T}\sum_{j=1}^{N}\delta_j(z_{t,j} - y_t)^2 \\ &= R_T^{\mathrm{VAW}}(\delta) + \sum_{j=1}^{N}\delta_j R_T^{\mathrm{VAW}}(\widehat{w}_j)\end{aligned}$$

$$+\frac{1}{2}\sum_{t=1}^{T}\sum_{j=1}^{N}\delta_j(\langle\widehat{w}_j,\Phi_{\theta_j}(x_t)\rangle-y_t)^2. \tag{45}$$

Оценим первый член, $R_T^{\mathrm{VAW}}(\delta)$. Из общей оценки (34) для сожаления алгоритма VAW следует, что

$$R_T^{\mathrm{VAW}}(\delta)\le\frac{\lambda}{2}\|\delta\|_2^2+\frac{NY^2}{2}\ln\left(1+\frac{Z_T^2T}{\lambda}\right). \tag{46}$$

Это выражение справедливо при условии, если $z_{t,j}=|\langle w_{t,j},\Phi_{\theta_j}(x_t)\rangle|\le Z_T$. Учитывая логарифмическую зависимость от Z_T, для анализа будет достаточно грубой оценки этой величины:

$$z_{t,j}^2\le 2(\langle w_{t,j},\Phi_{\theta_j}(x_t)\rangle-y_t)^2+2y_t^2.$$

Согласно (35)

$$\begin{aligned}\frac{1}{2}(\langle w_{t,j},\Phi_{\theta_j}(x_t)\rangle-y_t)^2&\le\frac{1}{2}\sum_{t=1}^{T}(\langle w_{t,j},\Phi_{\theta_j}(x_t)\rangle-y_t)^2\\&=\frac{1}{2}\sum_{t=1}^{T}(\langle w_j,\Phi_{\theta_j}(x_t)\rangle-y_t)^2+R_T^{\mathrm{VAW}}(w_j)\\&\le\frac{1}{2}\sum_{t=1}^{T}(\langle w_j,\Phi_{\theta_j}(x_t)\rangle-y_t)^2+\frac{\lambda\|w_j\|^2}{2}+\frac{mY^2}{2}\ln\left(1+\frac{2T}{\lambda}\right).\end{aligned}$$

Это справедливо для любого $w_j\in\mathbb{R}^m$. Положим $w_j=0$ в правой части последнего выражения. Следовательно

$$\frac{1}{2}(\langle w_{t,j},\Phi_{\theta_j}(x_t)\rangle-y_t)^2\le\frac{1}{2}TY^2+\frac{mY^2}{2}\ln\left(1+\frac{2T}{\lambda}\right).$$

Отсюда вытекает, что

$$\sum_{j=1}^{N}z_{t,j}^2\le Z_T^2:=2(T+1)NY^2+2mNY^2\ln\left(1+\frac{2T}{\lambda}\right). \tag{47}$$

Комбинируя оценки (46) и (47), получаем оценку для $R_T^{\mathrm{VAW}}(\delta)$:

$$R_T^{\mathrm{VAW}}(\delta) \le \frac{\lambda}{2}\|\delta\|_2^2 + \frac{NY^2}{2}\ln\left(1 + \frac{NY^2}{\lambda}\left(2T(T+1) + 2mT\ln\left(1+\frac{2T}{\lambda}\right)\right)\right). \tag{48}$$

Оценка математического ожидания второго члена в (45) может быть получена из (35) и леммы 2.1:

$$\sum_{j=1}^{n}\delta_j \mathop{\mathsf{E}}_{\theta} R_T^{\mathrm{VAW}}(\widehat{w}_j) \le \frac{\lambda}{2m}\sum_{j=1}^{n}\delta_j\|f_j\|_{\mathcal{H}_j}^2 + \frac{mY^2}{2}\ln\left(1+\frac{2T}{\lambda}\right). \tag{49}$$

Наконец, оценим математическое ожидание последнего члена в (45) с помощью леммы 2.2 (примененной в частном случае при $N = 1$):

$$\frac{1}{2}\sum_{t=1}^{T}\sum_{j=1}^{N}\delta_j \mathsf{E}_\theta(\langle\widehat{w}_j, \Phi_{\theta_j}(x_t)\rangle - y_t)^2 \le \sum_{j=1}^{N}\delta_j\left(\frac{T}{m}\|f_j\|_{\mathcal{H}_j}^2 + \frac{1}{2}\sum_{t=1}^{T}(f_j(x_t)-y_t)^2\right). \tag{50}$$

Для получения неравенства (43) рассматриваются стандартные базисные векторы $\delta = e_j$ из $\mathbb{R}^N$ и объединяются полученные оценки из (45), (48), (49) с (50). Соотношение (44) следует непосредственно из этих результатов. □

Предположим, что верхняя граница Y для значений y_t известна. В этом случае, последний член в выражении (43) может быть уменьшен путем усечения предсказаний экспертов. Вместо использования исходных предсказаний z_t, возьмем их усеченную версию

$$\overline{z}_t = \min(Y, \max(z_t, -Y)), \quad z_{t,j} = \langle w_{t,j}, \Phi_{\theta_j}(x_t)\rangle, \tag{51}$$

где операции max и min применяются к каждой компоненте вектора. Пусть $\overline{R}_T^{\mathrm{VAW}}(\delta)$ обозначает сожаление алгоритма VAW, применённого к последовательности $(\overline{z}_t, y_t)$. Тогда

$$\sum_{t=1}^{T}(\langle\alpha_t, \overline{z}_t\rangle - y_t)^2 = \overline{R}_T^{\mathrm{VAW}}(\delta) + \sum_{t=1}^{T}(\langle\delta, \overline{z}_t\rangle - y_t)^2$$

$$\leq \overline{R}_T^{\mathrm{VAW}}(\delta) + \sum_{t=1}^{T}\sum_{j=1}^{N} \delta_j (z_{t,j} - y_t)^2$$

Это неравенство справедливо для любых неотрицательных δ_i таких, что $\sum_{i=1}^{N} \delta_i = 1$, поскольку $(\overline{z}_{t,j} - y_t)^2 \leq (z_{t,j} - y_t)^2$. Подставим $Z_T = Y$ в выражение (46). Тогда

$$\overline{R}_T^{\mathrm{VAW}}(\delta) \leq \frac{\lambda}{2} + \frac{NY^2}{2} \ln\left(1 + \frac{Y^2 T}{\lambda}\right).$$

Данная оценка сожаления является более точной по сравнению с оценкой (48).

В рамках того же предположения об известности Y, оценки (43) могут быть дополнительно улучшены за счет использования других алгоритмов агрегирования мнений экспертов, отличных от VAW. Напомним, что функция потерь $\ell : [-Y, Y]^2 \to \mathbb{R}$ называется η-экспоненциально вогнутой, если функция $F(z) = e^{-\eta \ell(y,z)}$ является вогнутой для всех $y \in [-Y, Y]$. В частности, квадратичная функция потерь $\ell(y, z) = (y - z)^2$ обладает свойством η-экспоненциальной вогнутости при $\eta \leq 1/(8Y^2)$ [17, раздел 3.3]. Применяя алгоритм экспоненциально взвешенных средних (EWA) с начальными весами $\alpha_{1,j} = 1/N$ и обновлением весов

$$\alpha_{t,j} = \frac{\alpha_{t-1,j} \exp(-\eta(\overline{z}_{t,j} - y_t)^2)}{\sum_{k=1}^{N} \alpha_{t-1,k} \exp(-\eta(\overline{z}_{t,k} - y_t)^2)}, \quad t = 2, \ldots, T \tag{52}$$

где $\eta = 1/(8Y^2)$, получаем следующую оценку для регрессии [17, предложение 3.1]:

$$\overline{R}_{T,j}^{\mathrm{EWA}} := \frac{1}{2}\sum_{t=1}^{T} (\langle \alpha_t, \overline{z}_t \rangle - y_t)^2 - \frac{1}{2}\sum_{t=1}^{T} (\overline{z}_{t,j} - y_t)^2 \leq 4Y^2 \ln N. \tag{53}$$

Соответствующие улучшенные оценки представлены в следующей теореме.

Теорема 2.3. *Предположим, что значение константы Y известно. Пусть $w_{t,j} \in \mathbb{R}^m$ генерируется алгоритмами VAW, применёнными к последовательности $(\Phi_{\theta_j}(x_t), y_t)$. Одновременно $\alpha_t \in \mathbb{R}^N$ генерируется предсказателем EWA, применённым к последовательности $(\overline{z}_t, y_t)$, где $\overline{z}_t$ — вектор усечённых экс-*

пертных предсказаний (51). Тогда

$$\frac{1}{2}\mathsf{E}_\theta \sum_{t=1}^{T}(\langle \alpha_t, \overline{z}_t\rangle - y_t)^2 \le \frac{1}{2}\sum_{t=1}^{T}(f_j(x_t) - y_t)^2 + 4Y^2 \ln N + \left(\frac{\lambda}{2} + T\right)\frac{\|f_j\|^2_{\mathcal{H}_j}}{m} + \frac{mY^2}{2}\ln\left(1 + \frac{2T}{\lambda}\right) \qquad (54)$$

Это справедливо для любой функции $f_j \in \mathcal{F}_j$ и любого $j = 1, \ldots, N$. При этом оценка (44) из теоремы 2.2 сохраняет свою силу.

Доказательство. Для доказательства теоремы воспользуемся оценкой (53). Тогда получаем, что

$$\begin{aligned}\frac{1}{2}\sum_{t=1}^{T}(\langle \alpha_t, \overline{z}_t\rangle - y_t)^2 &= \overline{R}^{\mathrm{EWA}}_{T,j} + \frac{1}{2}\sum_{t=1}^{T}(\overline{z}_{t,j} - y_t)^2 \\ &\le 4Y^2 \ln N + \frac{1}{2}\sum_{t=1}^{T}(\langle w_{t,j}, \Phi_{\theta_j}(x_t)\rangle - y_t)^2 \\ &\le 4Y^2 \ln N + R^{\mathrm{VAW}}_T(\widehat{w}_j) + \frac{1}{2}\sum_{t=1}^{T}(\langle \widehat{w}_j, \Phi_{\theta_j}(x_t)\rangle - y_t)^2.\end{aligned}$$

Утверждение теоремы следует из этого неравенства в сочетании с оценками (49) и (50), применёнными для случая $\delta = e_j$. □

Алгоритм, рассмотренный в теореме 2.3, будем обозначать как VAW-EWA. При $m \ge \max\{d, N\}$ вычислительная сложность VAW-EWA на одну итерацию совпадает с таковой для алгоритма VAW^2 и составляет $O(Nm^2)$.

Хотя оценка (54) показывает незначительное теоретическое улучшение по сравнению с (43), асимптотическая оценка (44) для больших значений T в теореме 2.3 не улучшена. Возникает естественный вопрос о практической значимости улучшений, достигаемых за счет использования предсказателя EWA для усеченных экспертных оценок $\overline{z}_t$ вместо применения VAW непосредственно к исходным оценкам z_t. Согласно вычислительным экспериментам из раздела 2.3, линейные комбинации экспертных предсказаний в VAW^2 могут превосходить по эффективности выпуклые комбинации, используемые в VAW-EWA и

аналогичных мета-алгоритмах.

Алгоритмы, предложенные в работах [79, 84], могут быть классифицированы как OGD-OGD и OGD-EWA. Это связано с тем, что они используют онлайн градиентный спуск (OGD) в качестве экспертных стратегий, а в качестве мета-алгоритмов применяют либо OGD, либо методы типа EWA. Их вычислительная сложность составляет $O(Ndm)$ на итерацию, что существенно ниже. Однако соответствующие оценки регрессии имеют ограниченную применимость, так как квадратичная функция потерь не является глобально липшицевой, а коэффициенты $\widehat{w}$ из леммы 1.1 не ограничены.

2.3 Вычислительные эксперименты

2.3.1 Сравнение численного метода VAW^2 с другими методами многоядерной оптимизации

В данном разделе представлены результаты вычислительного исследования двухуровневого численного метода VAW^2. Методика проведения экспериментов основана на работе [35]. Комплекс программ для воспроизведения экспериментов находится в открытом доступе на платформе GitHub[2]. В приложении А.2. приводится листинг программы с реализацией алгоритма VAW^2.

Эксперименты проводились с использованием $N = 76$ ядер (51 гауссовское ядро и 25 лапласовских ядер, см. (29, 30) со следующими параметрами:

$$\sigma^2 \in \{10^{2i/25-2}\}_{i=0}^{50} \text{ (гауссовские ядра)}, \quad \sigma \in \{10^{i/6-2}\}_{i=0}^{24} \text{ (лапласовские ядра)}.$$

Число m случайных признаков было установлено равным 50, а параметр регуляризации λ равным 1 для алгоритма VAW. Эффективность алгоритмов оценивалась по среднеквадратичной ошибке

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(\widehat{f}_t(x_t) - y_t)^2,$$

[2] **Gurtovaya, O. V.** Random feature-based double Vovk-Azoury-Warmuth algorithm for online multi-kernel learning [Электронный ресурс] / O. V. Gurtovaya — 2025. — URL: `https://github.com/O-Gurt/VAW2` (дата обращения: 13.08.2025).

при этом результаты усреднялись по 5 независимым запускам в отличие от 20 запусков в оригинальном исследовании [35]. Таким образом, представленные результаты немного отличаются.

Для тестирования использовались те же наборы данных, что и в работе [35], включая:

- Реальные данные из UCI Machine Learning Repository[95]
- Искуственные данные, сгенерированные по модели AR(4):

$$x_t = 0.5x_{t-4} - 0.3x_{t-3} + 0.2x_{t-2} + 0.1x_{t-1} + \epsilon_t, \quad y_t = x_{t+1}, \tag{55}$$

где $\epsilon_t \sim \mathcal{N}(0,1)$, начальные условия $x_k = 0$ ($k = -3, ..., 0$), длина ряда $T = 5000$.

Перед проведением экспериментов все данные были нормализованы в соответствии с процедурой:

$$y_i := (y_i - \min_j y_j)/(\max_j y_j - \min_j y_j), \tag{56}$$

$$x_i := x_i / \max_j \|x_j\|_2, \tag{57}$$

что соответствует подходу, использованному в [35]. Данные были использованы без какой-либо другой предварительной обработки. Полный перечень наборов данных представлен в таблице 3.

Алгоритм VAW^2, проанализированный в теореме 2.2, и алгоритм VAW-EWA, проанализированный в теореме 2.3, сравнивались с несколькими другими алгоритмами:

- Raker [84]: алгоритм типа OGD-EWA в данной нотации. Каждый эксперт строит прогноз

  $$f_{i,t} = \langle w_{i,t}, x_t \rangle$$

  где $w_{i,t} \in \mathbb{R}^d$ — весовой вектор, x_t — входные данные. Мета-алгоритм

Название	Размер	Описание данных	Целевая переменная
Airfoil	(1503, 5)	аэродинамические профили при различных скоростях в аэродинамической трубе и углах атаки	масштабированное звуковое давление
Bias	(7750, 21)	измерения и прогнозы температуры вместе с вспомогательными географическими переменными	минимальная температура воздуха на следующий день
Concrete	(1030, 8)	характеристики бетона, такие как количество цемента или воды	прочность на сжатие
Naval	(11934, 15)	характеристики военно-морского судна, описываемые газотурбинной силовой установкой	положение рычага

Таблица 3: Краткие сведение об используемых в экспериментах наборах данных

EWA объединяет прогнозы

$$p_t = \sum_{i=1}^{N} \frac{w_{i,t-1} e^{-\eta(y_t - f_{i,t})^2}}{\sum_j w_{j,t-1} e^{-\eta(y_t - f_{j,t})^2}} f_{i,t}$$

где $\eta > 0$ — параметр обучения, N —число экспертов. Вычислительная сложность: $O(Nd)$ на итерацию.

- OMKL-GF [35]: схема выбора ядра, управляемая данными, где на каждом временном шаге строится двудольный граф обратной связи с весами

$$\pi_{i,t} = \frac{\exp(-\eta L_{i,t-1})}{\sum_{j=1}^{N} \exp(-\eta L_{j,t-1})}$$

где $L_{i,t} = \sum_{s=1}^{t} \ell_s(\kappa_i)$ — кумулятивные потери ядра κ_i. Вычислительная сложность: $O(Nm^2 + |E_t|)$, где $|E_t|$ — число рёбер графа.

- VAW-VAW-Aggr: прогнозы экспертных стратегий VAW

($f_{i,t} = \langle w_{i,t}, \Phi(x_t) \rangle$) объединяются алгоритмом VAW-Aggregating Вовка [17, Раздел 3.5] по формуле:

$$p_t = \sum_{i=1}^{N} \frac{e^{-\eta R_{i,t-1}}}{\sum_j e^{-\eta R_{j,t-1}}} f_{i,t}$$

где $R_{i,t} = \sum_{s=1}^{t} (y_s - f_{i,s})^2$ — суммарные потери, Φ — проекция на случайные признаки. Квадратичная функция потерь η-смешиваема с $\eta = 2$ [17, Раздел 3.6]. Таким образом, используя мета-алгоритм VAW-Aggregating с $\eta = 2$, можно достичь оценки сожаления немного лучше, чем для мета-алгоритма EWA [17, Предложение 3.2]. Вычислительная сложность: $O(Nm^2)$ на итерацию.

- VAW-ML-Prod, VAW-ML-Poly, VAW-BOA: прогнозы экспертных стратегий VAW объединяются онлайн алгоритмами второго порядка. Например, VAW-BOA использует

$$\eta_t = \sqrt{\frac{\log N}{\sum_{s=1}^{t} \|\nabla_s\|^2}},$$

где ∇_s — градиент потерь на шаге s. Эти алгоритмы используют как кумулятивные потери, так и дисперсию потерь для динамической адаптации скорости обучения [30, 101]. Вычислительная сложность: $O(Nm^2 + N \log N)$ на итерацию. Эти алгоритмы реализованы в библиотеке Opera [31].

Алгоритм VAW из теоремы 2.1 не рассматривался из-за его высокой вычислительной и пространственной сложности.

Результаты экспериментов собраны в таблице 4. Следует отметить, что теоретически все эти алгоритмы, кроме VAW^2, требуют знания интервала, содержащего метки, и должны использоваться с усеченными экспертными прогнозами. Поскольку здесь рассматривается $y_t \in [0, 1]$, вместо $y_t \in [-Y, Y]$, усе-

чение выполнялось соответствующим образом:

$$\overline{z}_t = \min(1, \max(z_t, 0)), \quad z_{t,j} = \langle w_{t,j}, \Phi_{\theta_j}(x_t) \rangle.$$

Для алгоритма VAW^2 представлены результаты как для исходных, так и для усеченных экспертных прогнозов: VAW^2(trunc). Однако эти опции дают почти одинаковые результаты. Наименьшие значения MSE показаны жирным шрифтом. Алгоритм VAW^2 показывает наилучший результат по всем наборам реальных данных.

	AR(4)	Airfoil	Bias	Concrete	Naval
Raker	23.24	28.64	12.70	35.29	11.32
OMKL-GF	20.47	24.37	7.05	34.24	4.60
VAW^2	16.56	22.80	**4.09**	**10.96**	**0.29**
VAW^2(trunc)	16.51	**22.78**	**4.09**	10.97	**0.29**
VAW-Aggr	16.40	26.74	5.02	13.57	0.45
VAW-EWA	16.49	27.61	5.41	15.08	0.62
VAW-BOA	**16.34**	26.42	4.98	13.88	0.52
VAW-ML-Poly	**16.34**	26.10	4.96	13.33	0.37
VAW-ML-Prod	**16.34**	26.27	4.97	13.64	0.48

Таблица 4: MSE (увеличено в 10^3 раз) алгоритмов MKL с 76 ядрами.

На рисунке 2 проиллюстрирована динамика изменения MSE для различных алгоритмов на протяжении итераций. Для улучшения читаемости и ясности графика из него были исключены кривые для VAW^2(trunc), VAW-BOA и VAW-ML-Poly. Анализ графика подтверждает, что алгоритм VAW^2 демонстрирует наименьшую траекторию MSE на всех рассмотренных реальных наборах данных, что свидетельствует о его высокоэффективной и стабильной работе на протяжении всего процесса обучения.

Для более глубокого понимания поведения предложенных алгоритмов, на рисунке 3 представлены векторы итоговых весов экспертов α_T, которые были присвоены алгоритмами VAW^2, VAW-EWA и VAW-ML-Prod по завершении обучения. Видно, что для алгоритма ML-Prod характерна разреженность в распределении весов, что означает концентрацию взвешивания на небольшом подмножестве ядер. В отличие от этого, алгоритм EWA распределяет веса более равномерно по всему набору ядер. Алгоритм VAW, в свою очередь, демонстрирует

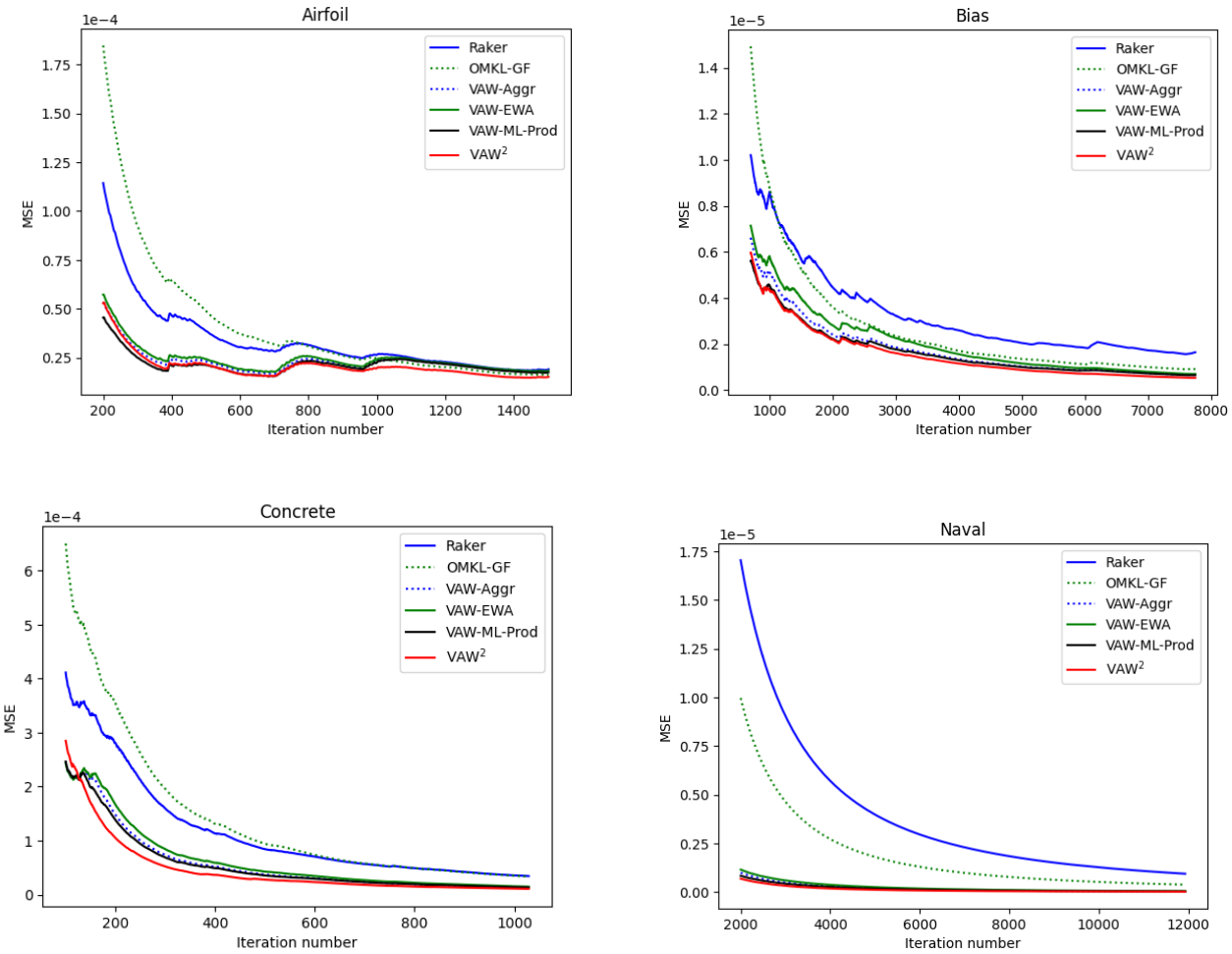


Рис. 2: Итеративное изменение MSE для многоядерных алгоритмов.

отличительную особенность, заключающуюся в существенном использовании отрицательных весов.

2.3.2 Приближение к автоматизированному построению моделей: трёхуровневый алгоритм Вовка-Азури-Вармута

В наше время всё больше возрастает необходимость в автоматизированном построении моделей, что обусловлено стремительным увеличением объёмов данных и сложностью аналитических задач. Современные системы AutoML позволяют существенно сократить время разработки моделей за счёт автоматизации ключевых этапов её построения — от предобработки сырых данных и генерации признаков до выбора оптимальной архитектуры модели и тонкой настройки гиперпараметров.

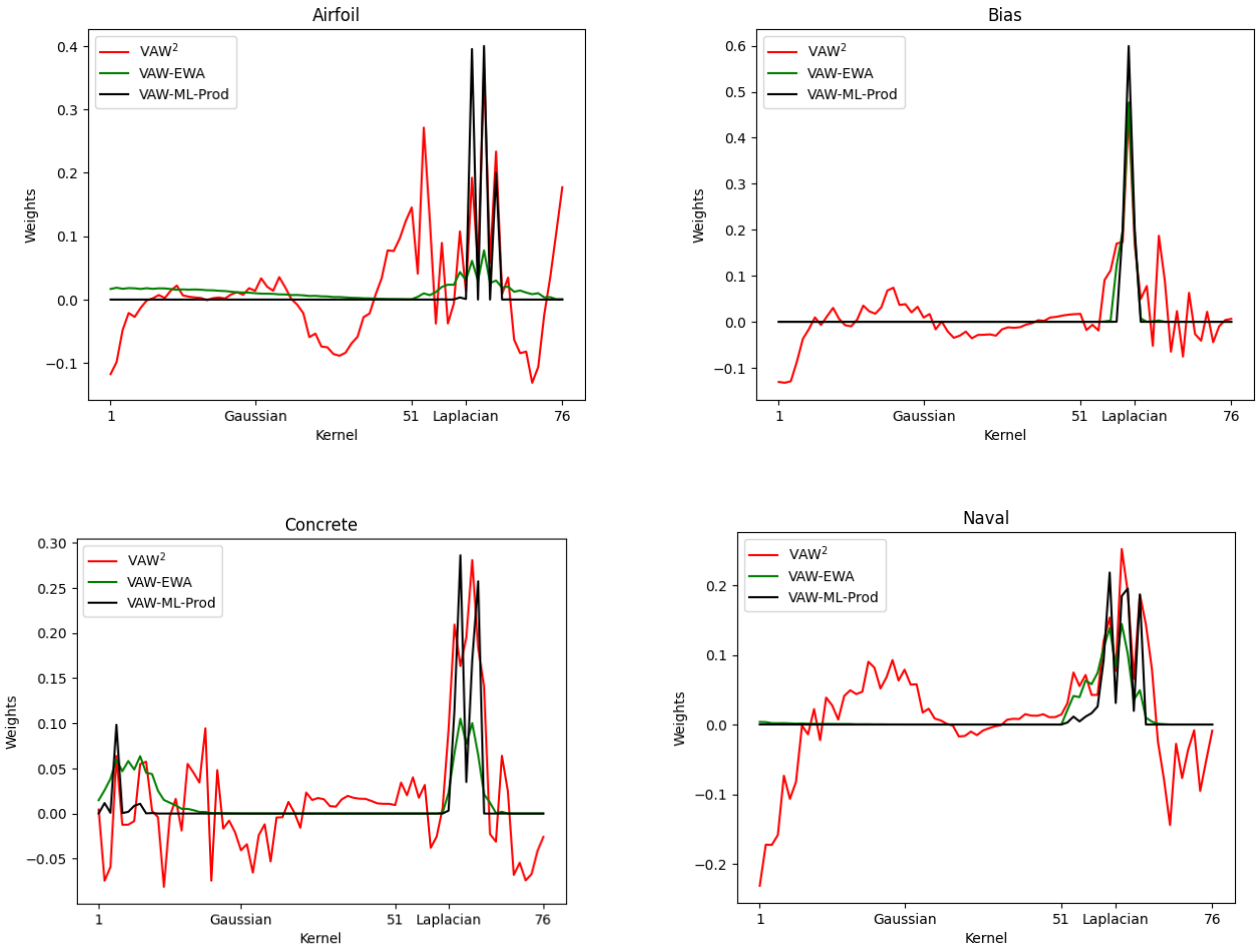


Рис. 3: Итоговые веса алгоритмов VAW^2, VAW-EWA и VAW-ML-Prod.

Особую ценность автоматизированные подходы представляют в условиях ограниченных ресурсов, когда требуется максимально эффективно использовать доступные вычислительные мощности и время работы специалистов. В промышленных приложениях, таких как прогнозирование спроса, автоматическая диагностика медицинских изображений или обработка естественного языка, AutoML-решения демонстрируют сопоставимую с ручными разработками точность при значительном сокращении временных затрат.

Перспективы развития технологии связаны с созданием адаптивных систем, способных автоматически учитывать специфику предметной области и динамически подстраиваться под изменяющиеся условия. Это особенно актуально для задач реального времени, где критически важны как скорость принятия решений, так и устойчивость моделей к дрейфу данных.

Актуальность проблемы автоматизации предварительной обработки дан-

ных будет продемонстрирована на примере численного метода VAW[2]. Для этого был сформирован расширенный набор табличных данных, охватывающий различные предметные области. Детальные характеристики этих наборов данных, включающие их размерность (количество строк и столбцов), наличие категориальных переменных (Кат.пер) и пропущенных значений (NaN), тип целевой переменной и краткое описание, представлены в таблице 5. Важно отметить, что для наборов данных Diamonds и California количество строк было сокращено до 500 и 10000 соответственно с целью оптимизации вычислительных ресурсов.

Название	**Размер**	**Кат.пер**	**NAN**	**Целевая переменная**	**Описание**
Airfoil	(1503, 6)	Нет	Нет	Уровень шума	Данные об акустическом шуме воздушных потоков вокруг авиационных крыльев
Concrete	(1030, 9)	Нет	Нет	Прочность на сжатие	Прочность бетона в зависимости от состава и возраста
Bias	(1000, 12)	Да	Да	Класс ошибки	Данные о систематических ошибках в параметрах систем
Naval	(11934, 16)	Нет	Нет	Состояние двигателя	Параметры работы корабельных двигателей и систем
Mercedes	(4209, 378)	Да	Да	Время тестирования	Время тестирования компонентов автомобилей
House-prices	(1460, 81)	Да	Да	Цена дома	Цены на жильё и характеристики недвижимости
Urban Traffic	(10080, 10)	Да	Нет	Интенсивность трафика	Данные о городском трафике (интенсивность, скорость)

Diamonds	(53940, 10)	Да	Нет	Цена	Характеристики алмазов (огранка, цвет, чистота, размеры) и их цена
Tips	(244, 7)	Да	Нет	Размер чаевых	Статистика чаевых в ресторанах
Boston	(506, 14)	Нет	Нет	Цена жилья	Цены на жильё и характеристики районов Бостона
California	(20640, 9)	Да	Нет	Цена жилья	Данные о стоимости жилья в Калифорнии
Energy Efficiency	(768, 8)	Нет	Нет	Нагрузка на отопление/охлаждение	Данные о характеристиках зданий и их влиянии на потребление энергии
Mtcars	(32, 11)	Да	Нет	Расход топлива	Характеристики автомобилей 1973-74 гг.

Таблица 5: Характеристики наборов данных

Основные источники, из которых были получены наборы данных из таблицы 5:

- Набор данных `Urban Traffic` доступен для загрузки с сайта UCI Machine Learning Repository [95].
- Наборы данных `Mercedes` и `House-prices` являются открытыми и доступны для загрузки на платформе соревнований по машинному обучению Kaggle. [46].
- Наборы данных `Energy Efficiency`, `California` и `Diamonds` встроены в библиотеку Scikit-learn.
- Наборы данных `Tips`, `Boston` и `Mtcars` широко используются в задачах машинного обучения и доступны через библиотеки Seaborn, PyDataset и Statsmodels соответственно.

Масштабирование данных — это ключевой этап предварительной обработки в машинном обучении. Оно представляет собой процесс трансформации признаков, целью которого является приведение их к сопоставимому диапазону значений или унификация статистических характеристик [40]. Эта процедура критически важна для ускорения сходимости алгоритмов, повышения эффективности работы моделей, чувствительных к масштабу признаков, а также для улучшения интерпретируемости полученных результатов. В ходе численных экспериментов были использованы следующие распространенные методы масштабирования:

- `StandardScaler`: Этот метод преобразует данные так, чтобы их среднее значение стало равно нулю, а стандартное отклонение — единице. Масштабированное значение x' для исходного x рассчитывается по формуле:

$$x' = \frac{x - \mu}{\sigma}$$

  где μ представляет среднее значение признака, а σ — его стандартное отклонение.

- `MinMaxScaler`: Применяется для приведения значений признаков к заданному диапазону, обычно $[0, 1]$. Масштабированное значение x' определяется как:

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}.$$

  Здесь x_{min} и x_{max} обозначают минимальное и максимальное значения признака соответственно.

- `RobustScaler`: Отличается устойчивостью к выбросам, поскольку использует медиану и межквартильный размах для масштабирования. Формула для вычисления x' выглядит следующим образом:

$$x' = \frac{x - Q_2}{Q_3 - Q_1},$$

  где Q_2 — это медиана (второй квартиль), Q_1 — первый квартиль (25-й перцентиль), а Q_3 — третий квартиль (75-й перцентиль) признака.

В таблице 6 представлены результаты применения алгоритма VAW^2 к различным наборам данных, где показаны значения MSE как с использованием ранее описанных методов масштабирования, так и без них.

	VAW^2	VAW^{2}StandardScaler	VAW^{2}MinMaxScaler	VAW^{2}RobustScaler
AIRFOIL	82.276	**20.873**	26.724	25.359
CONCRETE	336.328	28.339	**26.051**	27.934
BIAS	19.952	1.024	**0.873**	33.073
NAVAL	0.823	**0.003**	**0.003**	0.008
MERCEDES	**53.006**	640.388	**53.006**	**53.006**
HOUSE-PRICES	3.806×10^{10}	2.978×10^{9}	$\mathbf{1.203 \times 10^{9}}$	3.729×10^{9}
URBAN TRAFFIC	**2.176**	4.409	4.196	5.273
DIAMONDS	12.446×10^{5}	9.622×10^{5}	$\mathbf{9.621 \times 10^{5}}$	10.343×10^{5}
TIPS	1.193	**0.809**	1.191	1.079
BOSTON	19.746	**8.618**	12.420	11.383
FETCH CALIFORNIA	1.075	**0.353**	0.386	0.374
ENERGY EFFICIENCY	2.145	1.282	2.392	**0.851**
MTCARS	**3.071**	136.076	243.298	410.675

Таблица 6: Применение алгоритма VAW^2 с разным типом масштабирования. Жирным шрифтом выделены наименьшие MSE для каждого набора данных.

Результаты, представленные в таблице 6, демонстрируют существенную вариабельность эффективности различных методов масштабирования в зависимости от характеристик набора данных. Экспериментальные данные показывают, что ни один из исследуемых подходов не обладает абсолютным преимуществом на всех рассматриваемых датасетах, что подтверждает сложность задачи выбора оптимальной стратегии масштабирования. На основании этих наблюдений был разработан трёхуровневый численный метод S-VAW^2 (Scaled Vovk-Azoury-Warmuth), объединяющий в себе многоядерное моделирование с использованием случайных признаков Фурье [69] и стратегии предварительной обработки данных. Основная цель S-VAW^2 заключается в существенном повышении стабильности и точности предсказаний путём иерархического ансамблирования результатов, полученных от различных компонентов. Такой подход позволяет алгоритму эффективно справляться с разнородностью данных, обеспечивая при этом высокую эффективность в режиме онлайн.

Методология S-VAW^2 предусматривает применение различных методов масштабирования как для входных признаков $X = \{x_t\}_{t=1}^{T}$, так и для целевых переменных $Y = \{y_t\}_{t=1}^{T}$. Алгоритм инициализируется наборами функций масштабирования для признаков $\{S_1^X, \ldots, S_K^X\}$ и для меток $\{S_1^Y, \ldots, S_L^Y\}$, а также

семейством ядер $\{\kappa_j\}_{j=1}^{N}$, размерностью признаков m и параметрами регуляризации λ, λ'. При поступлении каждого нового элемента данных (x_t, y_t) в последовательном режиме, процесс обработки начинается с инициализации вектора предсказаний $P_t = (P_{t,1}, \ldots, P_{t,C})^\top$, который будет содержать результаты работы $C = KL$ экспертов, каждый из которых соответствует уникальной комбинации функций масштабирования признаков и меток.

Далее, для каждого масштабирующего эксперта осуществляется следующая процедура. Алгоритм циклически проходит по всем K видам масштабирования признаков $(S_1^X, \ldots, S_K^X)$ и для каждого из них по всем L видам масштабирования меток $(S_1^Y, \ldots, S_L^Y)$. На каждой итерации текущий входной признак x_t и метка y_t трансформируются с помощью выбранных функций масштабирования:

$$x'_t = S_i^X(x_t)$$

$$y'_t = S_j^Y(y_t)$$

После этого для масштабированных данных (x'_t, y'_t) применяется двухуровневый алгоритм VAW^2. Результатом его работы является предсказание $P_{t,(i-1)L+j}$. Важно отметить, что если для метки y_t было применено какое-либо масштабирование (то есть $S_j^Y \neq$ None), то полученное предсказание подвергается обратному масштабированию

$$P_{t,(i-1)L+j} = (S_j^Y)^{-1}(P_{t,(i-1)L+j})$$

Это гарантирует, что все предсказания экспертов представлены в исходном масштабе целевой переменной, что является критически важным для сопоставимости и корректного агрегирования.

После того как все C экспертов сделали свои предсказания для текущего объекта, финальное предсказание $\hat{y}_t$ для данного объекта формируется путём повторного применения алгоритма VAW к вектору предсказаний P_t, которые выступают в роли признаков для верхнего уровня VAW, и фактической метке y_t. На этом верхнем уровне агрегирования веса v_t для комбинации предсказаний

экспертов рассчитываются итеративно следующим образом:

$$v_t = \operatorname*{argmin}_v \left\{ \frac{\lambda'}{2}\|v\|^2 + \frac{1}{2}\sum_{i=1}^{t-1}(\langle P_i, v\rangle - y_i)^2 + \frac{1}{2}\langle P_t, v\rangle^2 \right\},$$

где λ' — параметр регуляризации для верхнего уровня агрегирования, P_i — вектор предсказаний экспертов на предыдущем шаге i, а y_i — соответствующая фактическая переменная. Окончательное предсказание $\hat{y}_t$ формируется как скалярное произведение $\langle v_t, P_t\rangle$. Этот верхний уровень агрегирования позволяет динамически взвешивать предсказания экспертов на основе их работы, повышая общую устойчивость и точность модели. Таким образом, иерархический подход S-VAW^2 позволяет эффективно справляться с разнородностью данных и изменениями в их распределении, обеспечивая при этом высокую эффективность в режиме онлайн.

Описанная процедура подробно изложена в алгоритме 2.

Для оценки эффективности предложенного численного метода S-VAW^2 в рамках данного исследования будет проведён сравнительный анализ с платформой автоматизированного построения моделей (AutoML) AutoGluon, разработанной Amazon Web Services (AWS). Особенностью AutoGluon является комплексная автоматизация всего цикла создания ML-моделей: от этапов предварительной обработки данных и отбора информативных признаков до выбора и оптимизации алгоритмов. Сравнительный анализ будет производиться только с модулем AutoGluon-Tabular, разработанным для работы с табличными данными. В работе [29] AutoGluon-Tabular продемонстрировал высокую эффективность и надёжность в сравнении с другими популярными AutoML-платформами, такими как H2O.ai, TPOT, Auto-sklearn и Google Cloud AutoML, на обширном наборе из 50 различных задач классификации и регрессии.

Функциональность AutoGluon-Tabular реализуется через последовательную серию автоматизированных этапов, инициируемых вызовом метода `fit()`.

1. **Автоматический анализ задачи и выбор метрики**: На начальной стадии конвейера система осуществляет детерминированный анализ целевой переменной для категоризации задачи прогнозирования. Это включа-

Алгоритм 2 Стратегия обучения S-VAW^2

Вход: набор данных $(x_t, y_t)_{t=1}^T$
Инициализация: виды масштабирования признаков $\{S_1^X, \ldots, S_K^X\}$, виды масштабирования меток $\{S_1^Y, \ldots, S_L^Y\}$, семейство ядер $\{\kappa_j\}_{j=1}^N$, размерность признаков m, параметры регуляризации λ, λ'
1. **Для** $t = 1$ **до** T **делать**
2. Инициализировать $P_t = (P_{t,1}, \ldots, P_{t,C})^\top$
3. **Для** $i = 1$ **до** K **делать**
4. **Для** $j = 1$ **до** L **делать**
5. $x'_t = S_i^X(x_t)$
6. $y'_t = S_j^Y(y_t)$
7. $P_{t,(i-1)L+j} = \text{VAW}^2(x'_t, y'_t)$ {вычисление предсказаний масштабирующих экспертов}
8. **Если** $S_j^Y \neq$ None, **то**
9. $P_{t,(i-1)L+j} = (S_j^Y)^{-1}(P_{t,(i-1)L+j})$ {выполнениие обратного масштабирования}
10. **Конец цикла**
11. **Конец цикла**
12. $\hat{y}_t = \text{VAW}(P_t, y_t)$
13. **Конец цикла**

Таблица 7: Алгоритм S-VAW^2

ет автоматическое определение, является ли задача бинарной классификацией, многоклассовой классификацией или регрессией. На основе этой классификации динамически выбирается соответствующая метрика оценки. Для задач классификации такой метрикой служит точность (accuracy):

$$Accuracy = \frac{\text{Количество правильных предсказаний}}{\text{Общее количество предсказаний}}$$

тогда как для задач регрессии используется среднеквадратичная ошибка (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

где $\hat{y}_i$ — i-е предсказание, полученное алгоритмом, а y_i — истинное значение i-го наблюдения.

2. **Комплексная предобработка данных**: После анализа целевой перемен-

ной AutoGluon-Tabular выполняет обширный комплекс операций по предобработке данных, адаптируя сырые входные данные для последующего моделирования.

- **Числовые признаки**: Пропущенные значения в числовых признаках заполняются медианными значениями. Распределения, демонстрирующие статистическую асимметрию, подвергаются квантильной нормализации, в то время как все остальные числовые признаки стандартизируются.
- **Категориальные признаки**: Обрабатываются дифференцированно: для признаков с высокой кардинальностью (как правило, с более чем четырьмя уникальными категориями) используются слои встраивания (embedding layers), интегрированные в нейронные сети, тогда как для признаков с небольшим количеством уникальных значений применяется One-Hot кодирование.
- **Текстовые и временные данные**: Система также включает специализированные модули для обработки текстовых и временных данных, преобразуя их в подходящие числовые представления, такие как векторы n-грамм для текста или упорядоченные числовые значения для дат.
- **Исключение признаков**: Столбцы, не поддающиеся однозначной категоризации (например, уникальные идентификаторы, не несущие прогностической ценности), могут быть исключены из дальнейшего анализа.

3. **Автоматизированная настройка гиперпараметров**: Система итеративно исследует пространство гиперпараметров, тестируя различные комбинации их значений для каждой модели. Для поиска оптимальных гиперпараметров используются такие стратегии, как байесовская оптимизация, а также алгоритмы, основанные на ранней остановке и распределении ресурсов, например, Hyperband/ASHA (Asynchronous Successive Halving Algorithm) [54].

4. **Обучение диверсифицированного набора базовых моделей**: AutoGluon-Tabular одновременно обучает диверсифицированный набор базовых моделей, что позволяет захватывать широкий спектр шаблонов в данных, значительно повышая надежность и прогностическую точность. В число обучаемых моделей входят алгоритмы градиентного бустинга, такие как LightGBM [47], CatBoost [67] и XGBoost [20], а также ансамблирующие методы, например, Random Forest [13] и Extra Trees [34]. Особое внимание уделяется табличным нейронным сетям (Tabular Neural Networks), которые специально разработаны для табличных данных и часто включают полносвязные слои с функцией активации ReLU для числовых признаков, линейные ярлыки (shortcut paths) и слои встраивания для категориальных признаков. Эти нейронные сети часто используют «широкую и глубокую» (wide and deep) архитектуру, позволяющую эффективно изучать как простые линейные зависимости, так и сложные нелинейные взаимодействия.

5. **Многослойное ансамблирование (стекинг) и бэггинг**: Для минимизации дисперсии предсказаний каждая базовая модель обучается на множестве бутстрап-выборок из тренировочного набора данных. В частности, AutoGluon реализует k-fold бэггинг, при котором каждая модель тренируется на k различных фолдах данных, а их предсказания впоследствии агрегируются путём усреднения. Процесс многослойного ансамбля инициализируется на первом слое, где обучается обширный набор разнообразных базовых моделей на исходных данных. Для предотвращения утечки данных и переобучения предсказания этих базовых моделей генерируются на out-of-fold (OOF) данных (т.е. на тех подмножествах, которые не были использованы для обучения конкретной модели). Далее, эти out-of-fold предсказания служат в качестве мета-признаков для обучения моделей-стекеров на втором слое. Такой подход позволяет мета-моделям эмпирически определять условия и степень доверия к предсказаниям базовых моделей. Данный иерархический процесс рекурсивно расширен до нескольких слоёв, где выходные данные моделей предыдущего слоя выступают в качестве входных для последующих, формируя сложную многоуровневую структуру ансамбля.

6. **Финальное агрегирование предсказаний**: На финальном этапе AutoGluon агрегирует предсказания всех лучших моделей из разных слоев ансамбля через взвешенное усреднение:

$$\hat{y}_{ensemble}(x) = \sum_{i=1}^{M} w_i \cdot \hat{y}_i(x)$$

Здесь $\hat{y}_{ensemble}(x)$ — итоговое ансамблированное предсказание, M — общее количество моделей в финальном ансамбле, включённых в финальный ансамбль (включая как базовые модели, так и модели из стекинговых слоев); $\hat{y}_i(x)$ — предсказание i-й модели, а w_i — вес i-й модели, удовлетворяющий условиям $\sum_{i=1}^{M} w_i = 1$ и $w_i \geq 0$. Веса w_i определяются путём оптимизации на валидационном наборе данных. AutoGluon использует итеративные алгоритмы, такие как неотрицательная линейная регрессия (NNLS), чтобы найти веса, минимизирующие функцию потерь. Это позволяет присвоить более высокие веса моделям, показывающие меньшие ошибки, повышая общую точность и надёжность ансамбля.

Для алгоритма AutoGluon-Tabular данные подавались в исходном виде, поскольку архитектура данного фреймворка включает встроенные механизмы автоматической предобработки данных. В отличие от этого, для модели S-VAW^2 была выполнена специализированная предобработка данных, включающая следующие этапы:

- Обработка пропущенных значений: Для всех численных признаков, содержащих пропущенные значения, применялся метод `KNNImputer` из библиотеки `sklearn.impute`. Пропущенные значения были замещены средним значением 5 ближайших соседей соответствующего столбца. Данный подход позволяет минимизировать потери данных и обеспечить целостность числовых признаков.

- Обработка категориальных признаков: Все категориальные признаки были преобразованы в числовой формат с использованием метода `OneHotEncoder` из библиотеки `sklearn.preprocessing`. Этот метод создает новые бинарные признаки для каждой уникальной категории в ис-

ходном категориальном столбце, что позволяет алгоритму корректно работать с нечисловыми данными.

В рамках текущей работы при проведении всех экспериментов для нового численного метода S-VAW^2 были сохранены те же параметры, как и для алгоритма VAW^2, обеспечивая тем самым сопоставимость результатов.

Для алгоритма S-VAW^2 были использованы следующие стратегии масштабирования данных:

- Для признаков X использовались следующие методы масштабирования:
  - `StandardScaler`
  - `MinMaxScaler`
  - `RobustScaler`
  - Вариант без масштабирования.
- Для целевых переменных Y использовались:
  - `MinMaxScaler`
  - Отсутствие масштабирования.

Общее количество комбинаций масштабирования, для которых обучался второй уровень S-VAW^2, составило $4 \times 2 = 8$. Предсказания от этих 8 экспертов затем были смешаны на третьем уровне S-VAW^2.

В экспериментах с AutoGluon-Tabular применялась версия 1.3.1 данного фреймворка с использованием параметров по умолчанию, за исключением лимита времени, который был установлен на 3600 секунд (1 час) для каждого набора данных. Это обеспечило адекватное время для поиска и обучения, а также позволило провести сравнение алгоритмов в равных условиях. Комплекс программ для воспроизведения экспериментов находится в открытом доступе на платформе GitHub[3]. В приложении А.3. приводится листинг программы с реализацией алгоритма S-VAW^2.

[3]**Gurtovaya, O. V.** Triple Vovk-Azuri-Warmuth Algorithm as an Automated Machine Learning Approach [Электронный ресурс] / O. V. Gurtovaya — 2025. — URL: `https://github.com/O-Gurt/TVAW` (дата обращения: 13.08.2025).

В таблице 8 представлены результаты применения алгоритмов S-VAW^2 и AG-Tabular к различным задачам регрессии.

	S-VAW^2	AG-TABULAR
AIRFOIL	**13.686**	20.160
CONCRETE	**20.977**	163.799
BIAS	**0.878**	1.177
NAVAL	0.0006	**3.461E-14**
MERCEDES	**48.759**	50.617
HOUSE-PRICES	1.356×10^9	$\mathbf{8.568 \times 10^8}$
URBAN TRAFFIC	**3.781**	21.824
DIAMONDS	8.063×10^5	$\mathbf{4.654 \times 10^5}$
TIPS	**0.803**	1.436
BOSTON	**6.538**	22.914
FETCH CALIFORNIA	**0.318**	0.362
ENERGY EFFICIENCY	**0.415**	1.088
MTCARS	**6.944**	8.083

Таблица 8: Результаты применения алгоритмов S-VAW^2 и AG-Tabular к различным задачам регрессии. Жирным шрифтом выделеные наименьшие MSE для каждого набора данных.

Анализ таблицы 8 показал, что алгоритм S-VAW^2 демонстрирует сопоставимую среднеквадратическую ошибку с одним из ведущих AutoML-фреймворков, AutoGluon-Tabular, превзойдя его на большинстве (10 из 13) исследованных датасетов. Эти результаты подтверждают эффективность многоуровневой ансамблевой архитектуры S-VAW^2. На наборе данных Naval была зафиксирована крайне низкая ошибка для AutoGluon-Tabular, равная 3.461E-14, что, вероятно, обусловлено спецификой данных, не содержащих шума. В целом, полученные выводы свидетельствуют о значительной прогностической способности и адаптивности алгоритма S-VAW^2, подчёркивая его перспективность для решения задач регрессии в условиях онлайн оптимизации.

Глава 2 основана на материалах, опубликованных в работах [72], [112].

2.4 Заключение к главе 2

В рамках данного исследования был представлен и проанализирован новый класс численных методов онлайн оптимизации, направленных на решение за-

дачи математического моделирования сложных регрессионных зависимостей в гильбертовых пространствах с воспроизводящим ядром.

Представленный численный метод VAW^2 является двухуровневым методом онлайн многоядерного моделирования. Он использует алгоритм Вовка-Азури-Вармута как для предсказаний отдельных ядер (на уровне экспертов), так и для их комбинации (на мета-уровне). Алгоритм VAW^2 отличается высокой вычислительной эффективностью по сравнению с прямым применением алгоритма VAW к конкатенированным векторам признаков, что делает его пригодным для практических задач. Для этого алгоритма установлена оценка математического ожидания для сожаления порядка $O(T^{1/2} \ln T)$ относительно искусственной случайности, при условии, что количество случайных признаков масштабируется как $T^{1/2}$.

Дальнейшим развитием этого направления стал трёхуровневый численный метод онлайн оптимизации S-VAW^2. Он был разработан как расширение VAW^2 и предназначен для устойчивого прогнозирования на табличных данных. Особенностью S-VAW^2 является его трёхуровневая иерархическая структура алгоритмов VAW, где на каждом уровне происходит адаптивное смешивание предсказаний. Важным нововведением в S-VAW^2 является интеграция различных стратегий масштабирования входных признаков и целевой переменной. Это позволяет алгоритму формировать более устойчивые и точные предсказания, эффективно адаптироваться к разнообразию данных, а также снижать чувствительность к выбору конкретных методов предобработки данных.

3 Об аппроксимации решения периодической одномерной квадратичной задачи вариационного исчисления с неизвестным внешним воздействием в режиме онлайн

В главах 1 и 2 диссертации была продемонстрирована эффективность методов онлайн оптимизации для решения различных задач регрессии. В частности, в первой главе был разработан алгоритм Вовка-Азури-Вармута, использующий случайные признаки Фурье для аппроксимации условного математического ожидания. Этот подход был развит во второй главе, где была расширена система ядер и применён экспертный подход, в рамках которого алгоритм Вовка-Азури-Вармута также использовался для агрегирования прогнозов. Ключевым методологическим элементом в этих главах стало сведение задач нелинейной регрессии к линейной с сохранением свойства выпуклости функции потерь, что, в свою очередь, позволяет применять эффективные онлайн алгоритмы.

В этой главе рассматривается задача моделирования внешнего воздействия динамической системы с квадратичным функционалом качества. Задача сводится к поиску решения дифференциального уравнения Эйлера, для которого предлагается эффективный численный метод аппроксимации. Исходная бесконечномерная задача сводится к конечномерной, но вместо случайных признаков Фурье аппроксимация выполняется с помощью тригонометрических полиномов. Это позволяет применять алгоритмы онлайн оптимизации для нахождения оптимальной траектории. Для различных сценариев предлагаются соответствующие алгоритмы и выводятся оценки статического и динамического сожалений.

3.1 Аппроксимация элементарной задачи оптимального управления периодическими траекториями

Рассмотрим многошаговую игру между игроком и противником. На шаге n игрок выбирает функцию управления $u_n(t)$, где $t \in [0,1]$ и $\int_0^1 u_n(t)\,dt = 0$, и начальную точку $a_n \in \mathbb{R}$. Затем противник выбирает функцию $h_n(t)$, $t \in [0,1]$, и игрок несёт потери

$$J_n(x_n) = \int_0^1 \left(\frac{1}{2}\dot{x}_n^2(t) + \frac{q^2}{2}x_n^2(t) - h_n(t)x_n(t) \right) dt, \tag{58}$$

где $\dot{x}_n(t) = u_n(t)$ и $x_n(0) = a_n$. Здесь $q > 0$ — некоторая константа. Эквивалентным образом, будем считать, что игрок непосредственно выбирает траекторию x_n с периодическим условием на концах рассматриваемого отрезка $x_n(0) = x_n(1)$. Через $x_1^N = (x_1, \ldots, x_N)$ обозначим последовательность решений, принимаемых игроком с шага 1 по шаг N. Тогда цель состоит в минимизации динамического сожаления

$$R_N(x_1^N, y_1^N) = \sum_{n=1}^{N} J_n(x_n) - \sum_{n=1}^{N} J_n(y_n), \tag{59}$$

вычисленного относительно последовательности сравнения $y_n(t)$, $t \in [0,1]$, с тем же условием периодичности $y_n(0) = y_n(1)$. В случае статического сожаления будет использоваться обозначение $R_N(x_1^N, y)$.

Наихудшим случаем будет являться случай, если в качестве y_n взять z_n, являющееся оптимальным решением задачи (58):

$$z_n = \arg\min\{J_n(x) : x(0) = x(1)\}.$$

Из элементарной теории вариационного исчисления [115] известно, что z_n является решением уравнения Эйлера

$$-\ddot{z} + q^2 z = h_n \tag{60}$$

с граничными условиями

$$z(0) = z(1), \quad \dot{z}(0) = \dot{z}(1). \tag{61}$$

Покажем, что функция z_n, удовлетворяющая уравнениям (60) и (61), является глобальным минимумом функционала J_n в пространстве непрерывно дифференцируемых 1-периодических функций $C_p^1(0,1) = \{x \in C(0,1) : x(0) = x(1)\}$. Возьмем произвольную функцию $x \in C_p^1(0,1)$ и вычислим разность функционалов

$$\begin{aligned} J_n(z_n + x) - J_n(z_n) &= \int_0^1 \left(\frac{1}{2}(\dot{z}_n + \dot{x})^2 + \frac{q^2}{2}(z_n + x)^2 - h_n(z_n + x) \right) dt \\ &\quad - \int_0^1 \left(\frac{1}{2}\dot{z}_n^2 + \frac{q^2}{2} z_n^2 - h_n z_n \right) dt \\ &= \int_0^1 \left(\dot{z}_n \dot{x} + \frac{1}{2}\dot{x}^2 + q^2 z_n x + \frac{q^2}{2} x^2 - h_n x \right) dt. \end{aligned}$$

Применим интегрирование по частям к члену $\int_0^1 \dot{z}_n \dot{x}\, dt$:

$$\int_0^1 \dot{z}_n \dot{x}\, dt = [\dot{z}_n x]_0^1 - \int_0^1 \ddot{z}_n x\, dt.$$

Учитывая условия периодичности на границах для функций z_n и x, получаем, что $[\dot{z}_n x]_0^1 = \dot{z}_n(1)x(1) - \dot{z}_n(0)x(0) = 0$. Таким образом, $\int_0^1 \dot{z}_n \dot{x}\, dt = -\int_0^1 \ddot{z}_n x\, dt$. Подставим этот результат в выражение для разности функционалов

$$\begin{aligned} J_n(z_n + x) - J_n(z_n) &= \int_0^1 \left(-\ddot{z}_n x + q^2 z_n x - h_n x \right) dt + \int_0^1 \left(\frac{1}{2}\dot{x}^2 + \frac{q^2}{2} x^2 \right) dt \\ &= \int_0^1 \left((-\ddot{z}_n + q^2 z_n - h_n) x \right) dt + \int_0^1 \left(\frac{1}{2}\dot{x}^2 + \frac{q^2}{2} x^2 \right) dt. \end{aligned}$$

Поскольку z_n удовлетворяет уравнению Эйлера (60), то выражение в скобках $(-\ddot{z}_n + q^2 z_n - h_n)$ пропадает. Следовательно, первый интеграл исчезает. Отсюда вытекает, что

$$J_n(z_n + x) - J_n(z_n) = \int_0^1 \left(\frac{1}{2}\dot{x}^2 + \frac{q^2}{2} x^2 \right) dt \geq 0.$$

Введём пространства Соболева 1-периодических функций $H_p^s(0,1)$ для $s = 1, 2$, полунормы и нормы которых заданы следующими выражениями:

$$|x|_{H^s} = \sqrt{\int_0^1 (x^{(s)})^2(t)\,dt}$$

и:

$$\|x\|_{H^s} = \sqrt{\sum_{k=0}^{s} |x|_{H^k}^2},$$

где $|x|_{H^0}^2 = \|x\|_{L^2}^2 = \int_0^1 x^2(t)\,dt$.

В дальнейшем под C будут пониматься различные положительные константы.

Пусть $h_n \in L^2(0,1)$. Из монографии [36, теорема 8.12] для решений эллиптических уравнений известно, что

$$\|z_n\|_{H^2} \le C(\|h_n\|_{L^2} + \|z_n\|_{L^2} + |z_n(0)|). \tag{62}$$

Для $\|z_n\|_{H^1}^2$ справедлива следующая оценка:

$$\begin{aligned} \|z_n\|_{H^1}^2 &\le C\int_0^1 (\dot z_n^2 + q^2 z_n^2)\,dt \\ &= C\int_0^1 (-\ddot z_n z_n + q^2 z_n^2)\,dt \\ &= C\int_0^1 h_n z_n\,dt \\ &\le C\|h_n\|_{L^2}\|z_n\|_{L^2}. \end{aligned} \tag{63}$$

Второе равенство в (63) получено интегрированием по частям с использованием периодических граничных условий $z_n(0) = z_n(1)$ и $\dot z_n(0) = \dot z_n(1)$. Третье равенство следует из уравнения Эйлера (60), а последнее неравенство — из неравенства Коши-Буняковского.

Оценим $z_n^2(0)$:

$$z_n^2(0) = \left(z_n(t) - \int_0^1 \dot{z}_n(s)\, ds \right)^2 \leq 2 \left(z_n^2(t) + \left(\int_0^1 \dot{z}_n(s)\, ds \right)^2 \right). \tag{64}$$

Проинтегрировав это неравенство по $t \in [0,1]$ и используя неравенство Коши-Буняковского для интегралов, получаем, что

$$z_n^2(0) \leq 2(\|z_n\|_{L^2}^2 + \|\dot{z}_n\|_{L^2}^2) = 2\|z_n\|_{H^1}^2.$$

Таким образом, $|z_n(0)| \leq \sqrt{2}\|z_n\|_{H^1}$. Из неравенства (63) следует, что $\|z_n\|_{L^2} \leq \|z_n\|_{H^1} \leq C\|h_n\|_{L^2}$.
Комбинируя оценки (62), (63), (64), получаем, что

$$\|z_n\|_{H^2} \leq C_0\|h_n\|_{L^2}, \tag{65}$$

где константа C_0 зависит исключительно от q^2.

Рассмотрим стандартную L^2-ортонормированную систему тригонометрических полиномов, определённых на $[0,1]$: $\psi_0 = 1$,

$$\psi_{2j-1}(t) = \sqrt{2}\sin(2\pi jt), \quad \psi_{2j}(t) = \sqrt{2}\cos(2\pi jt), \quad j \in \mathbb{N}.$$

и положим

$$\hat{x}_j = \int_0^1 x(t)\psi_j(t)\, dt, \quad j \in \mathbb{Z}_+ = \{0, 1, \dots\}.$$

Введём функцию округления $\lceil a \rceil = \min\{m \in \mathbb{Z}_+ : m \geq a\}$. Для любой 1-периодической функции $x \in L^2$ справедливо следующее утверждение [80, теорема D.36]: $x \in H_p^s$ при $s \geq 1$ тогда и только тогда, когда

$$\sum_{j=1}^{\infty} j^{2s} \left[\left(\int_0^1 x \sin(2\pi jt)\, dt \right)^2 + \left(\int_0^1 x \cos(2\pi jt)\, dt \right)^2 \right] = \frac{1}{2} \sum_{j=1}^{\infty} j^{2s} (\hat{x}_{2j-1}^2 + \hat{x}_{2j}^2)$$

$$= \frac{1}{2}\sum_{j=1}^{\infty}\left\lceil\frac{2j-1}{2}\right\rceil^{2s}\hat{x}_{2j-1}^2 + \frac{1}{2}\sum_{j=1}^{\infty}\left\lceil\frac{2j}{2}\right\rceil^{2s}\hat{x}_{2j}^2$$
$$= \frac{1}{2}\sum_{j=1}^{\infty}\left\lceil\frac{j}{2}\right\rceil^{2s}\hat{x}_j^2 < \infty.$$

Более того, для $|x|_{H^s}$ справедливо следующее выражение:

$$|x|_{H^s}^2 = (4\pi^2)^s\sum_{j=1}^{\infty} j^{2s}\left[\left(\int_0^1 x\sin(2\pi jt)\,dt\right)^2 + \left(\int_0^1 x\cos(2\pi jt)\,dt\right)^2\right]$$
$$= (4\pi^2)^s\sum_{j=1}^{\infty} j^{2s}(\hat{x}_{2j-1}^2 + \hat{x}_{2j}^2) = (4\pi^2)^s\sum_{j=1}^{\infty}\left\lceil\frac{j}{2}\right\rceil^{2s}\hat{x}_j^2.$$

Также, согласно тождеству Парсеваля

$$|x|_{H^0}^2 = \|x\|_{L^2}^2 = \left(\int_0^1 x\,dt\right)^2 + \sum_{j=1}^{\infty}\left[\left(\int_0^1 x\sin(2\pi jt)\,dt\right)^2 + \left(\int_0^1 x\cos(2\pi jt)\,dt\right)^2\right]$$
$$= \hat{x}_0^2 + \sum_{j=1}^{\infty}(\hat{x}_{2j-1}^2 + \hat{x}_{2j}^2) = \sum_{j=0}^{\infty}\hat{x}_j^2.$$

Таким образом, норма $\|x\|_{H^2}^2$ может быть выражена как $\sum_{k=0}^{2}|x|_{H^k}^2 = \sum_{k=0}^{2}\sum_{j=0}^{\infty}(4\pi^2)^k\lceil j/2\rceil^{2k}\hat{x}_j^2$. Из этого следует оценка для коэффициентов Фурье:

$$\hat{x}_j^2 \le \frac{\|x\|_{H^2}^2}{\sum_{k=0}^{2}(4\pi^2)^k\lceil j/2\rceil^{2k}}. \tag{66}$$

Пусть $\hat{z}_{n,j} = \int_0^1 z_n\psi_j\,dt$ обозначают коэффициенты Фурье оптимальных решений, определяемых уравнениями (60) и (61). Для $\|h_n\|_{L^2} \le b$ определим величины α_j:

$$\alpha_j = \frac{C_0 b}{\sqrt{\sum_{k=0}^{2}(4\pi^2)^k\lceil j/2\rceil^{2k}}}. \tag{67}$$

Из неравенств (65) и (66) непосредственно следует, что $|\hat{z}_{n,j}| \le \alpha_j$. Введём $\Psi_\alpha = (\sqrt{\alpha_j}\psi_j)_{j=0}^{2m}$. Для удобства будем искать приближённое решение в сле-

дующем виде:

$$w \cdot \Psi_\alpha = \sum_{j=0}^{2m} w_j \sqrt{\alpha_j} \psi_j, \quad w \in \mathcal{W}_{2m} = \prod_{j=0}^{2m} \left[-\sqrt{\alpha_j}, \sqrt{\alpha_j}\right].$$

Важно отметить, что для $j \leq 2m$, коэффициенты $\hat{z}_{n,j}$ могут быть представлены как $\hat{z}_{n,j} = \sqrt{\alpha_j}\hat{w}_{n,j}$ для некоторого $\hat{w}_n \in \mathcal{W}_{2m}$.

Теперь дадим более точное описание рассматриваемой задачи онлайн оптимизации. На каждом шаге $n \in \{1, \ldots, N\}$ данной игры игрок выбирает вектор $w_n = (w_{n,j})_{j=0}^{2m} \in \mathcal{W}_{2m}$, а противник выбирает функцию $h_n \in L^2$ такую, что $\|h_n\|_{L^2} \leq b$. Целью игрока является минимизация динамического сожаления

$$R_N(w_1^N, y_1^N) = \sum_{n=1}^{N} (J_n(w_n \cdot \Psi_\alpha) - J_n(y_n)) \tag{68}$$

относительно последовательности сравнения y_n, удовлетворяющей условиям:

$$y_n \in H_p^2, \quad \|y_n\|_{H^2} \leq C_0 b. \tag{69}$$

Без ограничения общности, будем считать, что

$$y_n = \arg\min \left\{ \int_0^1 \left(\frac{1}{2}\dot{x}^2(t) + \frac{q^2}{2}x^2(t) - g_n(t)x(t) \right) dt : x \in H_p^1 \right\} \tag{70}$$

для некоторой функции $g_n \in L^2$ с условием $\|g_n\|_{L^2} \leq b$. Для краткости примем обозначение $y_n = \mathcal{S}(g_n)$.

3.2 Сведение задачи к конечномерному случаю

В игре, описанной в конце предыдущего раздела, последовательность сравнения y_n выбирается из шара в бесконечномерном пространстве H_p^2. В данном разделе показывается, что задачу можно свести к конечномерному случаю, где применимы стандартные оценки сожаления.

Определим функцию $f_n(w)$ следующим образом:

$$f_n(w) = J_n(w \cdot \Psi_\alpha) = \frac{1}{2}\int_0^1 (w \cdot \dot{\Psi}_\alpha)^2\,dt + \frac{q^2}{2}\int_0^1 (w \cdot \Psi_\alpha)^2\,dt - \int_0^1 w \cdot \Psi_\alpha h_n\,dt$$
$$= \frac{1}{2}\sum_{j=0}^{2m} \alpha_j((2\pi)^2\lceil j/2\rceil^2 + q^2)w_j^2 - \sum_{j=0}^{2m}\sqrt{\alpha_j}\hat{h}_{n,j}w_j, \tag{71}$$

где $\hat{h}_{n,j}$ — коэффициенты Фурье функции h_n. Тогда динамическое сожаление

$$\overline{R}_N(w_1^N, u_1^N) = \sum_{n=1}^{N}(J_n(w_n \cdot \Psi_\alpha) - J_n(u_n \cdot \Psi_\alpha))$$
$$= \sum_{n=1}^{N}(f_n(w_n) - f_n(u_n)), \tag{72}$$

где $u_1^N \in \mathcal{W}_{2m}^N$ является конечномерной последовательностью сравнения.

Лемма 3.1. *Пусть $w_n \in \mathcal{W}_{2m}$ — последовательность векторов, построенная алгоритмом онлайн оптимизации для функций, заданных выражением (71). Пусть y_n удовлетворяет условию (69), и пусть*

$$\hat{v}_n = (\alpha_0^{-1/2}\hat{y}_{n,0}, \ldots, \alpha_{2m}^{-1/2}\hat{y}_{n,2m}).$$

Тогда выражения (68) и (72) связаны соотношением:

$$R_N(w_1^N, y_1^N) \le \overline{R}_N(w_1^N, \hat{v}_1^N) + C\frac{N}{m}b^2. \tag{73}$$

Доказательство. Обозначим $y_{n,2m} = \sum_{j=0}^{2m}\hat{y}_{n,j}\psi_j$. По определению, разность сожалений имеет вид:

$$R_N(w_1^N, y_1^N) - \overline{R}_N(w_1^N, \hat{v}_1^N) = \sum_{n=1}^{N}(J_n(\hat{v}_n \cdot \Psi_\alpha) - J_n(y_n))$$
$$= \sum_{n=1}^{N}(J_n(y_{n,2m}) - J_n(y_n)). \tag{74}$$

Далее, оценим модуль разности функционалов

$$|J_n(y_{n,2m}) - J_n(y_n)| = \left|\frac{1}{2}\int_0^1 (\dot{y}_{n,2m}^2 - \dot{y}_n^2)\,dt + \frac{q^2}{2}\int_0^1 (y_{n,2m}^2 - y_n^2)\,dt \right.$$
$$\left. - \int_0^1 (y_{n,2m} - y_n)h_n\,dt\right| \le \frac{1}{2}\|\dot{y}_{n,2m} - \dot{y}_n\|_{L^2}\|\dot{y}_{n,2m} + \dot{y}_n\|_{L^2}$$
$$+ \frac{q^2}{2}\|y_{n,2m} - y_n\|_{L^2}\|y_{n,2m} + y_n\|_{L^2} + \|y_{n,2m} - y_n\|_{L^2}\|h_n\|_{L^2}$$
$$\le Cb\|y_{n,2m} - y_n\|_{H^1}, \tag{75}$$

поскольку $\|y_{n,2m}\|_{H^1} \le \|y_n\|_{H^1} \le C\|h_n\|_{L^2} \le Cb$. Из монографии [80, теорема 12.31]) известно, что

$$\|y_n - y_{n,2m}\|_{H^r} \le \frac{C}{m^{2-r}}|y_n|_{H^2}, \quad r \in \{0,1\}.$$

Следовательно,

$$|J_n(y_{n,2m}) - J_n(y_n)| \le \frac{C}{m}b^2,$$

и утверждение леммы следует из (74). □

Обозначим через D_{2m} диаметр множества $\mathcal{W}_{2m}$, а через $G_{n,2m}$ и $L_{n,2m}$ — константы Липшица и гладкости функции f_n соответственно:

$$D_{2m} = \max\{\|u - v\|_2 : u, v \in \mathcal{W}_{2m}\} = \sum_{j=0}^{2m}(2\sqrt{\alpha}_j)^2,$$

$$|f_n(u) - f_n(v)| \le G_{n,2m}\|u - v\|_2, \quad |\nabla f_n(u) - \nabla f_n(v)| \le L_{n,2m}\|u - v\|,$$

для любых $u, v \in \mathcal{W}_{2m}$. Заметим, что функции f_n являются сильно выпуклыми с параметром сильной выпуклости [9, пример 5.19]

$$\mu_{2m} = \min_{0\le j\le 2m} \alpha_j((2\pi)^2\lceil j/2\rceil^2 + q^2).$$

Лемма 3.2. *Диаметр $\mathcal{W}_{2m}$, константа Липшица, константа гладкости и параметр сильной выпуклости функции f_n удовлетворяют следующим оценкам,*

равномерным по m и n:

$$D_{2m} \le D := Cb^{1/2}, \quad \mu_{2m} \ge \mu := Cb, \tag{76}$$

$$G_{n,2m} \le G =: Cb^{3/2}, \quad L_{n,2m} \le L =: Cb. \tag{77}$$

Доказательство. Первые две оценки следуют из неравенств

$$\frac{C_1 b}{1+j^2} \le \alpha_j \le \frac{C_2 b}{1+j^2}. \tag{78}$$

Кроме того,

$$\begin{aligned}\left|\frac{\partial f_n}{\partial w_j}\right| &= \left|\alpha_j((2\pi)^2\lceil j/2\rceil^2 + q^2)w_j - \sqrt{\alpha_j}\hat{h}_{n,j}\right| \\ &\le Cb|w_j| + \sqrt{\alpha}_j b \le Cb\sqrt{\alpha}_j, \\ G_{n,2m} &\le \sup_{w\in\mathcal{W}_{2m}} \|\nabla f_n(w)\|_2^2 \le Cb^2 \sum_{j=0}^{2m} \alpha_j \le Cb^3.\end{aligned}$$

Наконец,

$$\left|\frac{\partial f_n}{\partial w_j}(u) - \frac{\partial f_n}{\partial w_j}(v)\right| = \alpha_j((2\pi\lceil j/2\rceil)^2 + q^2)|u_j - v_j| \le Cb|u_j - v_j|.$$

Отсюда следует, что $L_{n,2m} \le Cb$. □

Леммы 3.1 и 3.2 позволяют свести исходную проблему к конечномерному случаю. Исходная постановка задачи подразумевает оценку динамического сожаления $R_N(w_1^N, y_1^N)$, где последовательность сравнения y_n принадлежит бесконечномерному пространству H_p^2. Лемма 3.1 демонстрирует, что для оценки истинного сожаления R_N достаточно оценить конечномерное сожаление $\overline{R}_N(w_1^N, u_1^N)$. Это сожаление вычисляется относительно последовательности сравнения u_n, являющейся элементом конечномерного пространства $\mathcal{W}_{2m}$. Для обеспечения необходимой точности аппроксимации требуется выбор достаточного числа m признаков ψ_j, определяющих размерность конечномерного пространства.

Существует достаточно много численных методов онлайн оптимизации,

предназначенных для минимизации сожаления в конечномерных пространствах. Оценки эффективности этих алгоритмов зависят от таких ключевых параметров математической модели, как диаметр множества допустимых решений D_{2m}, параметр сильной выпуклости μ_{2m}, константа Липшица функции потерь $G_{n,2m}$ и константа гладкости $L_{n,2m}$.

Ключевой аспект, обуславливающий эффективность сведения к конечномерному случаю, заключен в лемме 3.2. Данная лемма устанавливает, что все перечисленные величины (D_{2m}, μ_{2m}, $G_{n,2m}$, $L_{n,2m}$) равномерно ограничены по размерности конечномерного приближения m и по номеру шага итерации n.

3.3 Примеры численных методов

В данном разделе будут рассмотрены различные численные методы онлайн оптимизации для решения сформулированной задачи. Основное внимание будет уделено оценке сожаления этих численных методов в контексте как статических, так и динамических сценариев. Помимо этого, целью данного раздела является демонстрация возможности переноса известных теоретических результатов, полученных для конечномерных задач, на бесконечномерную постановку, что было обосновано в предыдущем разделе.

Рассмотрим алгоритм онлайн градиентного спуска (OGD) [107]:

$$w_{n+1} = \Pi_{\mathcal{W}_{2m}}(w_n - \eta_n \nabla f_n(w_n)), \quad \eta_n > 0. \tag{79}$$

Евклидова проекция $\Pi_{\mathcal{W}_{2m}}$ на гиперкуб $\mathcal{W}_{2m}$ может быть вычислена явно [8, пример 8.10]: $v = \Pi_{\mathcal{W}_{2m}} u$, где компоненты v_i определяются следующим образом:

$$v_i = \begin{cases} \sqrt{\alpha}_i, & u_i \geq \sqrt{\alpha}_i, \\ u_i, & -\sqrt{\alpha}_i < u_i < \sqrt{\alpha}_i, \\ -\sqrt{\alpha}_i & u_i \leq -\sqrt{\alpha}_i. \end{cases}$$

Теорема 3.1. *Пусть последовательность w_n построена с использованием алго-*

ритма OGD (79). В случае использования адаптивных шагов

$$\eta_n = \alpha \frac{D_{2m}}{\sqrt{\sum_{i=1}^{n} \|\nabla f_i(w_i)\|_2^2}}, \quad \alpha = \sqrt{2}/2 \tag{80}$$

статическое сожаление удовлетворяет неравенству:

$$R_N(w_1^N, y) \le Cb^2 \left(\sqrt{N} + \frac{N}{m} \right), \quad \|y\|_{H^2} \le C_0 b.$$

В частности, $R_N(w_1^N, y) = O(b^2 N^{1-\alpha})$ *при* $m \propto N^\alpha$, $\alpha \in (0, 1/2]$. *Для шагов* $\eta_n = 1/(\mu n)$ *справедливо:*

$$R_N(w_1^N, y) \le Cb^2 \left(1 + \ln N + \frac{N}{m} \right), \quad \|y\|_{H^2} \le C_0 b.$$

В частности, $R_N(w_1^N, y) = \widetilde{O}(\overline{b}^2 N^{1-\alpha})$ *при* $m \propto N^\alpha$, $\alpha \in (0, 1]$.

Доказательство. Требуется оценить первый член в выражении (73). Все утверждения непосредственно следуют из следующих двух оценок, полученных с использованием неравенств (76) и (77) из леммы 3.2:

$$\overline{R}_N(w_1^N, u) \le \sqrt{2} D \sqrt{\sum_{n=1}^{N} \|\nabla f_n(w_n)\|_2^2}, \quad u \in \mathcal{W}_{2m}$$

для адаптивных шагов (80) [64, теорема 4.14], и

$$\overline{R}_N(w_1^N, u) \le \frac{G^2}{2\mu}(1 + \ln N), \quad u \in \mathcal{W}_{2m} \tag{81}$$

для шагов $\eta_n = 1/(\mu n)$ [64, следствие 4.9]. □

Для адаптивных шагов (80), впервые предложенных в работе [89], алгоритм не опирается на сильную выпуклость и может гарантировать лишь оценку $O(N^{1/2})$ для R_N. При этом достаточно, чтобы количество m признаков ψ_j имело порядок так же $O(N^{1/2})$. Более быстрая скорость сходимости может быть достигнута с использованием шага $\eta_n = 1/(\mu n)$. В частности, при $m \propto N$ получено сожаление порядка $\tilde{O}(1)$ по отношению к N. Неравенство (81) было

получено в работе [41].

Для динамического сожаления предлагается использовать улучшенный алгоритм Ader [102]. Рассматривается M экспертных алгоритмов OGD с шагами η_i, и их веса инициализируются согласно вероятностному распределению

$$p_{1,i} = \frac{M+1}{M}\frac{1}{i(i+1)}, \quad i = 1, \ldots, M. \tag{82}$$

На каждом шаге результаты экспертных алгоритмов усредняются

$$w_n = \sum_{i=1}^{M} p_{n,i} w_{n,i}, \quad w_{n,i} \in \mathcal{W}. \tag{83}$$

Начальные значения $w_{1,i} \in \mathcal{W}_{2m}$ могут быть произвольными. Затем прогнозы экспертов w_n и веса p_n обновляются по правилам:

$$w_{n+1,i} = \Pi_{\mathcal{W}_{2m}}\left(w_{n,i} - \eta_i \nabla f_n(w_n)\right), \tag{84}$$

$$p_{n+1,i} \propto \exp\left(-\alpha \ell_n(w_{n,i})\right), \quad \ell_n(w) = \langle \nabla f_n(w_n), w - w_n \rangle. \tag{85}$$

Предположим, что последовательность сравнения $y_n = \mathcal{S}(g_n)$ генерируется согласно (70), и обозначим L^2-длину последовательности $(g_n)_{n=1}^N$,

$$P_N(g_1^N) = \sum_{n=2}^{N} \|g_n - g_{n-1}\|_{L^2}. \tag{86}$$

Аналогичное обозначение будет использоваться для длины конечномерной последовательности: $P_N(u_1^N) = \sum_{n=2}^{N} \|u_n - u_{n-1}\|_2$.

Теорема 3.2. *Пусть последовательность w_n построена с использованием улучшенного алгоритма Ader (82) – (85) с параметрами $\alpha = \sqrt{2/(NG^2D^2)}$,*

$$\eta_i = \frac{2^{i-1}D}{G}\frac{\sqrt{7}}{\sqrt{2N}}, \quad i = 1, \ldots, M,$$
$$M = \lceil 2^{-1}\log_2(1 + 4N/7)\rceil + 1.$$

Тогда динамическое сожаление удовлетворяет неравенству:

$$R_N(w_1^N, y_1^N) = \widetilde{O}\left(b^{3/2}\sqrt{N(b + P_N(g_1^N))} + b^2\frac{N}{m}\right), \quad \|g_n\|_{L^2} \le b. \tag{87}$$

В частности, $R_N(w_1^N, y_1^N) = \widetilde{O}\left(b^{3/2}\sqrt{N(b + P_N(g_1^N))}\right)$ *при* $m \propto \sqrt{N}$.

Доказательство. Связь между приращениями $\hat{v}_n$ (см. лемму 1.1) и L^2-приращениями g_n задаётся соотношением:

$$\begin{aligned}\|\hat{v}_n - \hat{v}_{n-1}\|_2^2 &= \sum_{j=0}^{2m}\frac{(\hat{y}_{n,j} - \hat{y}_{n-1,j})^2}{\alpha_j} \le \frac{C}{b}\sum_{j=0}^{2m}(1+j^2)(\hat{y}_{n,j} - \hat{y}_{n-1,j})^2 \\ &\le \frac{C}{b}\|y_{n+1} - y_n\|_{H^1}^2 \le \frac{C}{b}\|g_{n+1} - g_n\|_{L^2}^2,\end{aligned}$$

где в первом неравенстве использовано (78). Таким образом,

$$P_N(\hat{v}_1^N) \le \frac{C}{b^{1/2}}P_N(g_1^N). \tag{88}$$

Для конечномерной задачи сожаление улучшенного алгоритма Ader удовлетворяет оценке

$$\overline{R}_N(w_1^N, \hat{v}_1^N) \le \frac{3G}{4}\sqrt{2N(7D^2 + 4DP_N(\hat{v}_1^N))} + \frac{GD\sqrt{2N}}{2}(1 + 2\ln(k+1)),$$

где

$$k = \frac{1}{2}\left\lfloor \log_2\left(1 + \frac{4P_N}{7D}\right)\right\rfloor + 1,$$

приведённой в работе [102, теорема 4]. Из неравенства (88) и лемм 3.1, 3.2 вытекает оценка (87). □

Теперь предположим, что h_n изменяется «медленно». В этом случае целесообразно использовать оптимистичную версию онлайн градиентного спуска. Оценка сожаления для такой ситуации, связанной с вариацией градиента, впервые была получена в исследовании [22]. Следуя работе [18], определим алго-

ритм:

$$\widetilde{w}_{n+1} = \Pi_{\mathcal{W}_{2m}}(\widetilde{w}_n - \eta_n \nabla f_n(w_n)), \tag{89}$$

$$w_{n+1} = \Pi_{\mathcal{W}_{2m}}(\widetilde{w}_{n+1} - \eta_{n+1} \nabla f_n(w_n)). \tag{90}$$

Также введём обозначения:

$$\Sigma_N^2 = \sum_{n=1}^{N} \|h_n - h_{n-1}\|_{L^2}^2, \quad \Sigma_{\max}^2 = \max_{1 \le n \le N} \|h_n - h_{n-1}\|_{L^2}^2, \quad h_0 := 0.$$

Отметим, что

$$\|\nabla f_n(w) - \nabla f_{n-1}(w)\|_2^2 = \sum_{j=0}^{2m} \alpha_j (\hat{h}_{n,j} - \hat{h}_{n-1,j})^2.$$

Таким образом,

$$\sum_{n=1}^{N} \|\nabla f_n(w) - \nabla f_{n-1}(w)\|_2^2 \le C_2 b \Sigma_N^2, \tag{91}$$

$$\max_{1 \le n \le N} \|\nabla f_n(w) - \nabla f_{n-1}(w)\|_2^2 \le C_2 b \Sigma_{\max}^2. \tag{92}$$

Теорема 3.3. *Пусть последовательность w_n построена с использованием оптимистичного алгоритма градиентного спуска (89), (90) с шагом*

$$\eta_n = \frac{D}{\sqrt{10D^2L^2 + 4G^2 + \sum_{s=1}^{n-1} \|\nabla f_s(w_s) - \nabla f_{s-1}(w_{s-1})\|_2^2}}, \quad \nabla f_0(w_0) := 0.$$

Тогда статическое сожаление удовлетворяет неравенству:

$$R_N(w_1^N, y) \le Cb^2 \left(1 + \frac{N}{m}\right) + Cb\Sigma_N, \quad \|y\|_{H^2} \le C_0 b. \tag{93}$$

Доказательство. Используя (91) и лемму 3.2, перепишем оценку из работы [18, теорема 1] в наших обозначениях:

$$\overline{R}_N(w_1^n, v) \le C\left(D^2L + DG + Db\Sigma_N\right) \le C\left(b^2 + b\Sigma_N\right), \quad v \in \mathcal{W}_{2m}.$$

Теперь утверждение следует из леммы 3.1. □

В общем случае второй член в неравенстве (93) имеет порядок $\sqrt{N}$. Этот порядок может быть улучшен путём учёта сильной выпуклости f_n.

Теорема 3.4. *Пусть последовательность w_n построена с использованием оптимистичного алгоритма градиентного спуска (89), (90) с $\eta_n = 2/(\mu n)$. Тогда*

$$R_N^s(w_1^N, y) \leq Cb^2 \frac{N}{m} + \widetilde{O}\left(b^2 + \Sigma_{\max}^2\right).$$

Доказательство. Из работы [18, теорема 3] следует, что

$$\overline{R}_N(w_1^n, v) = \widetilde{O}\left(b\frac{\Sigma_{\max}^2}{\mu} + \frac{L^2D^2 + G^2}{\mu} + \mu D^2\right), \quad v \in \mathcal{W}_{2m},$$

где также учтено (92). Утверждение непосредственно следует из лемм 3.1, 3.2. □

В случае динамического сожаления оптимистичный онлайн градиентный спуск требует более сложной схемы обновления, чем улучшенный алгоритм Ader, для получения наилучших оценок сожаления. Целесообразно использовать Алгоритм 2 из работы [18], основанный на идеях [105] и [91]. Первым шагом алгоритма является инициализация весов

$$w_1 = \widetilde{w}_1 \in \mathcal{W}; \quad p_{1,i} = \frac{1}{M}, \quad i = 1, \ldots, M.$$

На каждом шаге n усредняются результаты базовых алгоритмов

$$w_n = \sum_{i=1}^{M} p_{n,i} w_{n,i}, \quad w_{n,i} \in \mathcal{W}$$

Результаты базовых алгоритмов обновляются с помощью оптимистичного градиентного спуска

$$\widetilde{w}_{n+1,i} = \Pi_{\mathcal{W}}\left(\widetilde{w}_{n,i} - \eta_i \nabla f_n(w_n)\right),$$
$$w_{n+1,i} = \Pi_{\mathcal{W}}\left(\widetilde{w}_{n+1,i} - \eta_i \nabla f_n(w_n)\right).$$

Веса обновляются с помощью оптимистичного алгоритма Hedge из работы [91]

$$p_{n+1,i} \propto \exp\left(-\varepsilon_n \sum_{j=1}^{n} (\ell_{j,i} + m_{n+1,i})\right)$$

где

$$\begin{aligned}
\ell_{n,i} &= \langle \nabla f_n(w_n), w_{n,i} \rangle + \lambda \|w_{n,i} - w_{n-1,i}\|_2^2, \quad n \geq 2, \\
\ell_{1,i} &= \langle \nabla f_1(w_1), w_{1,i} \rangle, \\
m_{n+1,i} &= \langle M_{n+1}, w_{n+1,i} \rangle + \lambda \|w_{n+1,i} - w_{n,i}\|_2^2, \quad n \geq 1, \\
M_{n+1} &= \nabla f_n(w_n), \quad n \geq 1; \quad M_1 = 0,
\end{aligned}$$

и $\lambda > 0$ является параметром.

Теорема 3.5. *Пусть последовательность w_n построена с использованием описанного алгоритма 2 из работы [18] с параметрами:*

$$\begin{aligned}
\eta_i &= \min\left\{1/(8L), \sqrt{(D^2/(8G^2N))2^{i-1}}\right\}, \quad i = 1, \ldots, M, \\
M &= \lceil 2^{-1} \log_2(G^2N/(8L^2D^2)) \rceil + 1, \\
\varepsilon_n &= \min\left\{1/(8D^2L), \sqrt{(\ln M)/(D^2V_n)}\right\}, \\
V_n &= \sum_{j=1}^{n} \|\nabla f_j(w_j) - \nabla f_{j-1}(w_{j-1})\|_2^2, \quad f_0 := 0.
\end{aligned}$$

Тогда

$$R_N(w_1^N, y_1^N) \leq Cb^2 \frac{N}{m} + \widetilde{O}\left(b^2 + bP_N(g_1^N) + \sqrt{b + P_N(g_1^N)}\Sigma_N b\right). \tag{94}$$

где $y_n = \mathcal{S}(g_n)$ и $P_N(g_1^N)$ определены в (70) и (86) соответственно.

Доказательство. Как и ранее, требуется оценить первый член в правой части (73). Применим неравенство (37) из работы [18], полученное в доказательстве их теоремы 7:

$$\overline{R}_N(w_1^N, \hat{u}_1^N) = \widetilde{O}\left(G\sqrt{D^2 + DP_N(\hat{u}_1^N)}\Sigma_N b^{1/2} + D^2L + DLP_N(\hat{u}_1^N) + G^2/L\right)$$

$$= \widetilde{O}\left(\sqrt{b + P_N(g_1^N)\Sigma_N b + b^2 + bP_N(g_1^N)}\right).$$

Здесь также использовались оценка 88) и оценки из леммы 3.2. □

Замечание 3.1. Для медленно изменяющихся h_n может показаться целесообразным применение жадной стратегии предсказания: $x_n = \mathcal{S}(h_{n-1})$. Поскольку справедливо выражение (см. (75) для первого неравенства):

$$\begin{aligned} |J_n(x_n) - J_n(y_n)| &\le Cb\|x_n - y_n\|_{H^1} \\ &\le Cb\|h_{n-1} - g_n\|_{L^2}. \end{aligned}$$

Тогда

$$R_N(x_1^N, y_1^N) \le Cb\sum_{n=1}^{N} \|h_{n-1} - g_n\|_{L^2}.$$

В наихудшем случае, когда игрок соревнуется с пророком: $g_n = h_n$, эта оценка $R_N(x_1^N, y_1^N) \le CbP_N(h_1^N)$ является лучшей по сравнению с оценкой (94). Однако, в общем случае сожаление не является малым. Например, легко проверить, что для осциллирующей последовательности:

$$h_n = a_n \cos(2\pi j_1 t) + (1 - a_n)\cos(2\pi j_2 t), \quad 0 < j_1 < j_2,$$

где $a_n = 1$ для чётных n и $a_n = 0$ для нечётных n, статическое сожаление жадного алгоритма относительно тривиальной последовательности сравнения $y_n = 0$ растёт линейно по N, так как $\int_0^1 h_n \mathcal{S}(h_{n-1})\, dt = 0$, и $J_n(\mathcal{S}(h_{n-1}))$ равномерно ограничен снизу положительной константой.

3.3.1 Вычислительные эксперименты

Приведем результаты численного моделирования, проведенного для оценки эффективности предложенных алгоритмов. Комплекс программ для воспроизведения экспериментов находится в открытом доступе на платформе GitHub [4]. В качестве внешнего воздействия в выражении (58) была использована функция

[4] **Gurtovaya, O. V.** Online learning in a one-dimensional periodic quadratic variational problem with an adversarial external force [Электронный ресурс] / O. V. Gurtovaya — 2025. — URL: `https://github.com/O-Gurt/PQVP_1d` (дата обращения: 13.08.2025).

$h_n(t) = 10\pi^2 \cos(2\psi_n \pi t)$, где поведение параметра ψ_n варьировалось в зависимости от шага n для моделирования различных сценариев.

Рассматривались следующие пять сценариев для параметра ψ_n:

- Сценарий с постоянным воздействием: $\psi_n = \text{const} = 1.5$, что соответствует случаю, когда функция h_n не зависит от n.
- Сценарий со стохастическим воздействием: $\psi_n \sim N(0, 0.03)$, что моделирует внешнее воздействие со стохастической природой.
- Сценарий с медленно меняющимся воздействием: $\psi_n = 2 + \frac{n}{T-1}$, где $n = 0, \ldots, T-1$, то есть значения ψ_n выбираются равномерно из дискретного набора в интервале $[2, 3]$.
- Сценарий с медленно меняющимся воздействием и шумом: $\psi_n = 2 + \frac{n}{T-1} + \xi$, где $\xi \sim N(0, 0.03)$, что является усложнением предыдущего случая за счет добавления стохастической компоненты.
- Сценарий с переключением внешнего воздействия: интервал шагов разбивается на две равные части, в каждой из которых параметр ψ_n принимает постоянное значение: в первой половине $\psi_n = 0.5$, во второй — $\psi_n = 1.5$.

Вычислительные эксперименты были реализованы на языке программирования Python. Для нахождения аналитического решения на каждом шаге использовалась библиотека символьной математики SymPy. Все эксперименты проводились с общим количеством шагов $N = 5000$. Количество используемых косинусов и синусов было установлено по 30. Прочие параметры были заданы следующим образом: $q = 1$, $b = 80$, $C = 1$. Коэффициенты α_j были вычислены по формуле (67), а остальные параметры (диаметр множества допустимых решений, константа сильной выпуклости, константа Липшица функции потерь и константа гладкости) — на основе леммы 3.2.

Несмотря на то, что каждый алгоритм был разработан для конкретного сценария, его эффективность была проанализирована для всех пяти случаев. В качестве метрики эффективности алгоритмов использовалось усреднённое по числу итераций N сожаление

$$\frac{1}{N}\overline{R}_N(w_1^N, \hat{x}_1^N) = \frac{1}{N}\sum_{n=1}^{N}(J_n(w_n \cdot \Psi_\alpha) - J_n(\hat{x}_n \cdot \Psi_\alpha)), \tag{95}$$

вычисленное по отношению к оптимальному решению $\hat{x}_1^N$. Результаты численного моделирования представлены в таблице 9.

	$\psi = 1.5$	$\psi \sim$ N(0, 0.03)	$\psi \sim$ U(2, 3)	$\psi = \{0.5, 1.5\}$	$\psi \sim U(2,3)$ $+N(0, 0.03)$
OGD v1	$5.958 \cdot 10^2$	$7.011 \cdot 10^2$	$5.954 \cdot 10^2$	$5.954 \cdot 10^2$	$5.983 \cdot 10^2$
OGD v2	$2.674 \cdot 10^1$	$1.656 \cdot 10^2$	$8.178 \cdot 10^0$	$1.636 \cdot 10^1$	$8.436 \cdot 10^0$
Ader	$2.110 \cdot 10^{-1}$	$1.696 \cdot 10^2$	$1.980 \cdot 10^{-1}$	$2.400 \cdot 10^{-1}$	$1.228 \cdot 10^0$
OptOGD v1	$1.014 \cdot 10^2$	$1.760 \cdot 10^3$	$5.022 \cdot 10^2$	$5.250 \cdot 10^2$	$2.063 \cdot 10^2$
OptOGD v2	$3.367 \cdot 10^0$	$1.689 \cdot 10^2$	$1.151 \cdot 10^1$	$1.970 \cdot 10^1$	$1.178 \cdot 10^1$
OMDHedge	$6.850 \cdot 10^{-1}$	$1.813 \cdot 10^2$	$5.400 \cdot 10^{-1}$	$1.100 \cdot 10^0$	$9.800 \cdot 10^{-1}$

Таблица 9: Сравнение эффективности численных методов онлайн оптимизации при разном внешнем воздействии.

Анализ результатов (табл. 9) показал, что ни один численный метод не является универсально лучшим. При этом алгоритмы OGD и его оптимистичная версия (OptOGD) исследовались в двух модификациях: с учётом сильной выпуклости (v2) и без неё (v1).

Наименьшее сожаление в стабильных и предсказуемых сценариях (постоянное, медленно меняющееся и переключающееся воздействие) демонстрирует Ader. В условиях стохастического воздействия наиболее эффективен OGD v2. Алгоритм OMDHedge (алгоритм 2 из работы [18]) показал превосходство в самом сложном сценарии (медленно меняющееся воздействие с шумом), подтвердив устойчивость к стохастическим помехам. Наибольшее сожаление, особенно при сильных стохастических компонентах, наблюдалось у OGD v1 и его оптимистичных вариантов (OptOGD v1/v2).

3.4 Заключение к главе 3

В данной главе были рассмотрены численные методы для решения задачи вариационного исчисления с неизвестным внешним воздействием. Было продемонстрировано сведение исходной задачи, определённой на функциональных пространствах, к конечномерной задаче с помощью разложения по ортонормированному базису тригонометрических полиномов. Этот подход, обоснованный леммой 3.1, позволил перенести анализ сожаления из бесконечномерного в конечномерное пространство. Лемма 3.2 дополнительно обеспечила, что ключевые параметры конечномерной задачи — диаметр множества допустимых решений, константа сильной выпуклости, константы Липшица и гладкости — остаются равномерно ограниченными, что является критически важным условием для применения стандартных теоретических результатов онлайн оптимизации.

На основе данного сведения были проанализированы различные численные методы онлайн оптимизации. В теореме 3.1 установлены оценки сожаления для алгоритма онлайн градиентного спуска (OGD), демонстрируя, что с соответствующим выбором шага можно гарантировать сублинейный рост сожаления, в частности, $\widetilde{O}(N^{1-\alpha})$ в статическом случае. Для динамической среды был проанализирован улучшенный алгоритм Ader, чья теоретическая оценка сожаления, согласно теореме 3.2, зависит от длины последовательности сравнения ($P_N(g_1^N)$), что отражает способность алгоритма адаптироваться к изменяющимся условиям. Кроме того, теоремы 3.3 и 3.4 показали, что для медленно меняющихся сред оптимистичные версии OGD могут обеспечить более сильные гарантии, где сожаление зависит от вариации градиента (Σ_N). Список таких алгоритмов может быть продолжен. Например, аналогичные результаты справедливы для стохастически расширенной состязательной модели [77, 18] и для оптимистичного онлайн градиентного спуска с произвольным методом предсказания градиента [82].

Для эмпирической валидации этих теоретических результатов и оценки относительной эффективности алгоритмов в различных условиях были проведены вычислительные эксперименты. Их результаты в целом подтвердили тео-

ретические предсказания.

Глава 3 основана на материалах, опубликованных в работе [71].

Заключение

Сформулируем основные результаты диссертационного исследования.

1. Разработана методика применения численного метода Вовка–Азури–Вармута (VAW) к случайным признакам Фурье для моделирования данных с марковской зависимостью с теоретическим обоснованием и экспериментальной проверкой на моделях AR(1) и VAR(1).

2. Предложен новый численный метод VAW^2 для моделирования различных регрессионных зависимостей. Получена оценка сожаления порядка $O(T^{1/2} \ln T)$. Алгоритм VAW^2 обладает значительно более низкой вычислительной сложностью по сравнению с алгоритмом VAW, применённым к конкатенированным векторам случайных признаков. В ходе численных экспериментов установлено, что алгоритм VAW^2 превосходит известные из литературы родственные алгоритмы на ряде эталонных наборов данных.

3. Разработан трёхуровневый численный метод S-VAW^2, объединяющий иерархическое агрегирование и стратегии масштабирования данных. Сравнительный анализ показал, что его точность сопоставима с ведущим AutoML-фреймворком, при этом он является вычислительно более эффективной альтернативой.

4. Сформулирована и решена задача моделирования внешнего воздействия динамической системы с квадратичным функционалом качества, включая её сведение к конечномерной задаче и вывод границ статического и динамического сожалений.

5. Реализован комплекс программ на Python для численного моделирования и сравнительного анализа предложенных онлайн-алгоритмов: алгоритм VAW с использованием случайных признаков Фурье; двухуровневый алгоритм VAW^2; трёхуровнеый алгоритм S-VAW^2. Проведённые эксперименты подтвердили теоретические результаты и продемонстрировали практическую эффективность методов.

Список литературы

[1] **Agarwal A.** Algorithms for portfolio management based on the newton method / A. Agarwal, E. Hazan, S. Kale, R. E. Schapire // Proceedings of the 23rd International Conference on Machine Learning (ICML 2006). — 2006. — P. 9–16. — DOI: 10.1145/1143844.1143846.

[2] **Agarwal A.** The generalization ability of online algorithms for dependent data / A. Agarwal, J. C. Duchi // IEEE Transactions on Information Theory. — 2012. — Vol. 59, no. 1. — P. 573–587. — DOI: 10.1109/TIT.2012.2214202.

[3] **Anava O.** Online learning for time series prediction / O. Anava, E. Hazan, S. Mannor // Proceedings of the 28th Conference on Learning Theory (COLT 2015). — 2015. — P. 172–184.

[4] **Arbabi H.** Nonautonomous Koopman Operator Approximation / H. Arbabi, I. Mezic // IEEE Conference on Decision and Control (CDC 2023). — 2023. — P. 1234–1241. — DOI: 10.1109/CDC49753.2023.10383363.

[5] **Auer P.** Near-optimal regret bounds for reinforcement learning / P. Auer, T. Jaksch, R. Ortner // Advances in Neural Information Processing Systems 21 (NIPS 2008). — 2008. — P. 89–96.

[6] **Azoury K. S.** Relative loss bounds for on-line density estimation with the exponential family of distributions / K. S. Azoury, M. K. Warmuth // Machine Learning. — 2001. — Vol. 43, no. 3. — P. 211–246. — DOI: 10.1023/A:1010601366626.

[7] **Bach F. R.** Multiple kernel learning, conic duality, and the SMO algorithm / F. R. Bach, G. R. G. Lanckriet, M. I. Jordan // Proceedings of the 21st International Conference on Machine Learning (ICML 2004). — 2004. — P. 41–48. — DOI: 10.1145/1015330.1015424.

[8] **Beck A.** Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB / A. Beck. — Philadelphia: SIAM, 2014. — 270 p.

[9] **Beck A.** First-order methods in optimization / A. Beck. — Philadelphia: SIAM, 2017. — 487 p.

[10] **Bhattacharya R.** Nonparametric Learning in L^2_π Spaces / R. Bhattacharya, Y. Lin // Annals of Statistics. — 2023. — Vol. 51, no. 3. — P. 1345–1372. — DOI: 10.1214/23-AOS2291.

[11] **Blackwell D.** An analog of the minimax theorem for vector payoffs / D. Blackwell // Pacific Journal of Mathematics. — 1956. — Vol. 6, no. 1. — P. 1–8. — DOI: 10.2140/pjm.1956.6.1.

[12] **Bradley R. C.** Basic properties of strong mixing conditions. A survey and some open questions / R. C. Bradley // Probability Surveys. — 2005. — Vol. 2. — P. 107–144. — DOI: 10.1214/154957805100000104.

[13] **Breiman L.** Random forests / L. Breiman // Machine Learning. — 2001. — Vol. 45, no. 1. — P. 5–32. — DOI: 10.1023/A:1010933404324.

[14] **Carmeli C.** Vector valued reproducing kernel Hilbert spaces and universality / C. Carmeli, E. De Vito, A. Toigo, V. Umanità // Analysis and Applications. — 2010. — Vol. 8, no. 1. — P. 19–61. — DOI: 10.1142/S0219530510001487.

[15] **Carratino L.** Learning with SGD and random features / L. Carratino, A. Rudi, L. Rosasco // Advances in Neural Information Processing Systems 31 (NeurIPS 2018). — 2018. — P. 10213–10224.

[16] **Cesa-Bianchi N.** On the generalization ability of on-line learning algorithms / N. Cesa-Bianchi, A. Conconi, C. Gentile // IEEE Transactions on Information Theory. — 2004. — Vol. 50, no. 9. — P. 2050–2057. — DOI: 10.1109/TIT.2004.833339.

[17] **Cesa-Bianchi N.** Prediction, learning, and games / N. Cesa-Bianchi, G. Lugosi. — Cambridge: Cambridge University Press, 2006. — 394 p.

[18] **Chen S.** Optimistic online mirror descent for bridging stochastic and adversarial online convex optimization / S. Chen, Y.-J. Zhang, W.-W. Tu, P. Zhao, L. Zhang // Journal of Machine Learning Research. — 2024. — Vol. 25, no. 178. — P. 1–62.

[19] **Chen T.** Diffusion Models for Markovian Dynamics Learning / T. Chen, Y. Lipman // Advances in Neural Information Processing Systems 35 (NeurIPS 2022). — 2022. — P. 28765–28779.

[20] **Chen T.** Xgboost: A scalable tree boosting system / T. Chen, C. Guestrin // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. — 2016. — P. 785–794. — DOI: 10.1145/2939672.2939785.

[21] **Chen X.** Sparse High-Dimensional Vector Autoregressive Modeling / X. Chen [et al.] // Journal of Computational and Graphical Statistics. — 2019. — Vol. 28, no. 2. — P. 321–333. — DOI: 10.1080/10618600.2018.1518237.

[22] **Chiang C.-K.** Online optimization with gradual variations / C.-K. Chiang [et al.] // Proceedings of the 25th Annual Conference on Learning Theory (COLT 2012). — 2012. — P. 6.1–6.20.

[23] **Davis R. A.** Sparse VAR Modeling for Large Scale Time Series / R. A. Davis [et al.] // Econometric Theory. — 2019. — Vol. 35, no. 4. — P. 713–752. — DOI: 10.1017/S0266466618000247.

[24] **Dean S.** Regret bounds for robust adaptive control of the linear quadratic regulator / S. Dean, H. Mania, N. Matni, B. Recht, S. Tu // Advances in Neural Information Processing Systems 31 (NeurIPS 2018). — 2018. — P. 4188–4197.

[25] **Dekel O.** The forgetron: A kernel-based perceptron on a budget / O. Dekel, S. Shalev-Shwartz, Y. Singer // SIAM Journal on Computing. — 2008. — Vol. 37, no. 5. — P. 1342–1372. — DOI: 10.1137/060666998.

[26] **Douc R.** Markov Chains / R. Douc, E. Moulines, P. Priouret, P. Soulier. — Cham: Springer International Publishing, 2018. — 622 p.

[27] **Doukhan P.** Mixing: Properties and Examples / P. Doukhan. — New York, NY: Springer New York, 1994. — 142 p.

[28] **Duchi J.** Adaptive subgradient methods for online learning and stochastic optimization / J. Duchi, E. Hazan, Y. Singer // Journal of Machine Learning Research. — 2011. — Vol. 12, no. 7. — P. 2121–2159.

[29] **Erickson N.** Autogluon-tabular: Robust and accurate automl for structured data / N. Erickson [et al.] // arXiv preprint. — 2020. — arXiv:2003.06505. — URL: https://arxiv.org/abs/2003.06505 (дата обращения: 15.01.2025).

[30] **Gaillard P.** A second-order bound with excess losses / P. Gaillard, G. Stoltz, T. Van Erven // Proceedings of the 27th Conference on Learning Theory (COLT 2014). — 2014. — P. 176–196.

[31] **Gaillard P.** Opera: Online Prediction by Expert Aggregation [Электронный ресурс] / P. Gaillard, O. Wintenberger. — 2022. — URL: https://github.com/Dralliag/opera-python (дата обращения: 15.01.2025).

[32] **Gama J.** Knowledge Discovery from Data Streams / J. Gama. — Boca Raton: CRC Press, 2013. — 237 p.

[33] **Gao R.** Online dynamic ensemble deep random vector functional link neural network for forecasting / R. Gao, R. Li, M. Hu, P. N. Suganthan, K. F. Yuen // Neural Networks. — 2023. — Vol. 166. — P. 51–69. — DOI: 10.1016/j.neunet.2023.07.015.

[34] **Geurts P.** Extremely randomized trees / P. Geurts, D. Ernst, L. Wehenkel // Machine Learning. — 2006. — Vol. 63, no. 1. — P. 3–42. — DOI: 10.1007/s10994-006-6226-1.

[35] **Ghari P. M.** Graph-aided online multi-kernel learning / P. M. Ghari, Y. Shen // Journal of Machine Learning Research. — 2023. — Vol. 24, no. 21. — P. 1–44.

[36] **Gilbarg D.** Elliptic partial differential equations of second order / D. Gilbarg, N. S. Trudinger. — Berlin, Heidelberg: Springer, 2001. — 517 p.

[37] **Gönen M.** Multiple kernel learning algorithms / M. Gönen, E. Alpaydın // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2211–2268.

[38] **Gönen M.** Multiple kernel learning algorithms / M. Gönen, E. Alpaydın // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2211–2268.

[39] **Hannan J.** Approximation to Bayes risk in repeated play / J. Hannan // Contributions to the Theory of Games. — 1957. — Vol. 3. — P. 97–139.

[40] **Hastie T.** The elements of statistical learning: data mining, inference, and prediction / T. Hastie, R. Tibshirani, J. Friedman. — 2nd ed. — New York: Springer, 2009. — 745 p. — DOI: 10.1007/978-0-387-84858-7.

[41] **Hazan E.** Logarithmic Regret Algorithms for Online Convex Optimization / E. Hazan, A. Kalai, S. Kale, A. Agarwal // Learning Theory: Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006). — Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. — P. 499–513. — DOI: 10.1007/11776420_37.

[42] **Hazan E.** Introduction to online convex optimization / E. Hazan // Foundations and Trends® in Optimization. — 2016. — Vol. 2, no. 3–4. — P. 157–325. — DOI: 10.1561/2400000013.

[43] **Hofmann T.** Kernel methods in machine learning / T. Hofmann, B. Schölkopf, A. J. Smola // Annals of Statistics. — 2008. — Vol. 36, no. 3. — P. 1171–1220. — DOI: 10.1214/009053607000000677.

[44] **Hoi S. C. H.** Online learning: A comprehensive survey / S. C. H. Hoi, D. Sahoo, J. Lu, P. Zhao // Neurocomputing. — 2021. — Vol. 459. — P. 249–289. — DOI: 10.1016/j.neucom.2021.06.088.

[45] **Jin C.** Inverse Reinforcement Learning for Markovian Systems / C. Jin, A. Sidford // Journal of Machine Learning Research. — 2023. — Vol. 24, no. 1. — P. 1–48.

[46] **Kaggle Inc.** Kaggle Datasets [Электронный ресурс]. — 2025. — URL: `https://www.kaggle.com/datasets` (дата обращения: 14.08.2025).

[47] **Ke G.** Lightgbm: A highly efficient gradient boosting decision tree / G. Ke [et al.] // Advances in Neural Information Processing Systems 30 (NIPS 2017). — 2017. — P. 3146–3154.

[48] **Kivinen J.** Online learning with kernels / J. Kivinen, A. J. Smola, R. C. Williamson // IEEE Transactions on Signal Processing. — 2004. — Vol. 52, no. 8. — P. 2165–2176. — DOI: 10.1109/TSP.2004.830991.

[49] **Kivinen J.** Online learning with kernels / J. Kivinen, A. J. Smola, R. C. Williamson // IEEE Transactions on Signal Processing. — 2004. — Vol. 52, no. 8. — P. 2165–2176. — DOI: 10.1109/TSP.2004.830991.

[50] **Lanckriet G. R. G.** Learning the kernel matrix with semidefinite programming / G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M. I. Jordan // Journal of Machine Learning Research. — 2004. — Vol. 5. — P. 27–72. — URL: https://www.jmlr.org/papers/volume5/lanckriet04a/lanckriet04a.pdf (дата обращения: 15.01.2025).

[51] **Ledzewicz U.** Pitfalls in applying optimal control to dynamical systems: An overview and editorial perspective / U. Ledzewicz, H. Schättler // Discrete and Continuous Dynamical Systems - S. — 2022. — Vol. 15, no. 9. — P. 1–20.

[52] **Lecué G.** Empirical risk minimization is optimal for the convex aggregation problem / G. Lecué // Bernoulli. — 2013. — Vol. 19, no. 5. — P. 2153–2166. — DOI: 10.3150/12-BEJ452.

[53] **Li Q.** Koopman Embeddings for Nonlinear Control Systems / Q. Li, Y. Chen // Automatica. — 2023. — Vol. 157. — P. 111246. — DOI: 10.1016/j.automatica.2023.111246.

[54] **Li L.** Hyperband: A novel bandit-based approach to hyperparameter optimization / L. Li [et al.] // Journal of Machine Learning Research. — 2018. — Vol. 18, no. 185. — P. 1–52.

[55] **Littlestone N.** Mistake bounds and logarithmic linear-threshold learning algorithms / N. Littlestone // University of California, Santa Cruz, Technical Report. — 1989. — 43 p.

[56] **Lu J.** Recommender systems in e-commerce / J. Lu, Z. Liu, D. Wu // Electronic Commerce Research and Applications. — 2015. — Vol. 14, no. 5. — P. 286–296. — DOI: 10.1016/j.elerap.2015.04.002.

[57] **Lütkepohl H.** New Introduction to Multiple Time Series Analysis / H. Lütkepohl. — Berlin: Springer, 2005. — 764 p.

[58] **Mollenhauer M.** Kernel-based Approximation of Markov Operators / M. Mollenhauer, T. J. Sullivan, P. Koltai // Journal of Machine Learning Research. — 2021. — Vol. 22. — P. 1–56.

[59] **Nicholson W. B.** Bayesian VARs: Specification Choices and Forecast Accuracy / W. B. Nicholson [et al.] // Journal of Applied Econometrics. — 2020. — Vol. 35, no. 2. — P. 176–194. — DOI: 10.1002/jae.2745.

[60] **Nóbrega J. P.** A sequential learning method with Kalman filter and extreme learning machine for regression and time series forecasting / J. P. Nóbrega, A. L. I. Oliveira // Neurocomputing. — 2019. — Vol. 337. — P. 235–250. — DOI: 10.1016/j.neucom.2019.01.058.

[61] **Nummelin E.** Geometric ergodicity of Harris recurrent Markov chains with applications to renewal theory / E. Nummelin, P. Tuominen // Stochastic Processes and their Applications. — 1982. — Vol. 12, no. 2. — P. 187–202. — DOI: 10.1016/0304-4149(82)90041-2.

[62] **Orabona F.** The projectron: a bounded kernel-based perceptron / F. Orabona, J. Keshet, B. Caputo // Proceedings of the 25th International Conference on Machine Learning (ICML 2008). — 2008. — P. 720–727. — DOI: 10.1145/1390156.1390245.

[63] **Orabona F.** A modern introduction to online learning / F. Orabona // arXiv preprint. — 2019. — arXiv:1912.13213. — URL: https://arxiv.org/abs/1912.13213 (дата обращения: 15.01.2025).

[64] **Orabona F.** A modern introduction to online learning / F. Orabona // arXiv preprint. — 2023. — arXiv:1912.13213v6. — URL: https://arxiv.org/abs/1912.13213 (дата обращения: 15.01.2025).

[65] **Philipp M.** Spectral Methods for Markov Transition Operators / M. Philipp, P. Koltai // SIAM Journal on Applied Dynamical Systems. — 2024. — Vol. 23, no. 1. — P. 412–439. — DOI: 10.1137/23M1550388.

[66] **Pontryagin L. S.** The Mathematical Theory of Optimal Processes / L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, E. F. Mishchenko. — New York: Interscience, 1962. — 360 p.

[67] **Prokhorenkova L.** CatBoost: unbiased boosting with categorical features / L. Prokhorenkova [et al.] // Advances in Neural Information Processing Systems 31 (NeurIPS 2018). — 2018. — P. 6638–6648.

[68] **Rahimi A.** Random features for large-scale kernel machines / A. Rahimi, B. Recht // Advances in Neural Information Processing Systems 20 (NIPS 2007). — 2007. — P. 1177–1184.

[69] **Rahimi A.** Uniform approximation of functions with random bases / A. Rahimi, B. Recht // 2008 46th Annual Allerton Conference on Communication, Control, and Computing. — 2008. — P. 555–561. — DOI: 10.1109/ALLERTON.2008.4797607.

[70] **Rahimi A.** Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning / A. Rahimi, B. Recht // Advances in Neural Information Processing Systems 21 (NIPS 2008) / eds. D. Koller, D. Schuurmans, Y. Bengio, L. Bottou. — 2008. — P. 1313–1320.

[71] **Rokhlin D. B.** Online learning in a one-dimensional periodic quadratic variational problem with an adversarial external force / D. B. Rokhlin, O. V. Gurtovaya // Journal of Mathematical Sciences. — 2024. — Vol. 280, no. 3. — P. 1–12. — DOI: 10.1007/s10958-024-07245-3.

[72] **Rokhlin D. B.** Random feature-based double Vovk-Azoury-Warmuth algorithm for online multi-kernel learning / D. B. Rokhlin, O. V. Gurtovaya // arXiv preprint. — 2025. — arXiv:2503.20087. — URL: https://arxiv.org/abs/2503.20087 (дата обращения: 15.01.2025).

[73] **Rokhlin D. B.** Vovk–Azoury–Warmuth algorithm and random Fourier features for a regression problem with Markovian data / D. B. Rokhlin, O. V. Gurtovaya // Lobachevskii Journal of Mathematics. — 2024. — Vol. 45, no. 12. — P. 6186–6200. — DOI: 10.1134/S1995080224120368.

[74] **Rudi A.** Generalization Properties of Learning with Random Features / A. Rudi, L. Rosasco // Advances in Neural Information Processing Systems 30 (NIPS 2017). — 2017. — P. 3215–3225.

[75] **Rudi A.** Generalization properties of learning with random features / A. Rudi, L. Rosasco // Advances in Neural Information Processing Systems 30 (NIPS 2017). — 2017. — P. 3215–3225.

[76] **Russo G.** Markovian Recurrent Networks for Nonlinear System Identification / G. Russo, J.-J. Slotine // International Conference on Machine Learning (ICML 2023). — 2023. — P. 18934–18948.

[77] **Sachs S.** Between stochastic and adversarial online convex optimization: Improved regret bounds via smoothness / S. Sachs, H. Hadiji, T. van Erven, C. Guzmán // Advances in Neural Information Processing Systems 35 (NeurIPS 2022). — 2022. — Vol. 35. — P. 691–702.

[78] **Safikhani A.** Structural Vector Autoregressive Modeling for Network Data / A. Safikhani, A. Shojaie // Journal of the American Statistical Association. — 2020. — Vol. 115, no. 531. — P. 1267–1280. — DOI: 10.1080/01621459.2019.1660171.

[79] **Sahoo D.** Large scale online multiple kernel regression with application to time-series prediction / D. Sahoo, S. C. H. Hoi, B. Li // ACM Transactions on Knowledge Discovery from Data. — 2019. — Vol. 13, no. 1. — P. 1–33. — DOI: 10.1145/3278609.

[80] **Salgado A. J.** Classical numerical analysis: a comprehensive course / A. J. Salgado, S. M. Wise. — Cambridge: Cambridge University Press, 2022. — 750 p.

[81] **Schölkopf B.** Learning with kernels: support vector machines, regularization, optimization, and beyond / B. Schölkopf, A. J. Smola. — Cambridge: MIT Press, 2002. — 626 p.

[82] **Scroccaro P. Z.** Adaptive composite online optimization: predictions in static and dynamic environments / P. Z. Scroccaro, A. S. Kolarijani, P. M. Esfahani // IEEE Transactions on Automatic Control. — 2023. — Vol. 68, no. 5. — P. 2906–2921. — DOI: 10.1109/TAC.2022.3214578.

[83] **Shalev-Shwartz S.** Online Learning and Online Convex Optimization / S. Shalev-Shwartz // Foundations and Trends® in Machine Learning. — 2011. — Vol. 4, no. 2. — P. 107–194. — DOI: 10.1561/2200000018.

[84] **Shen Y.** Random feature-based online multi-kernel learning in environments with unknown dynamics / Y. Shen, T. Chen, G. B. Giannakis // Journal of Machine Learning Research. — 2019. — Vol. 20, no. 22. — P. 1–36.

[85] **Shiryaev A. N.** Probability-1 / A. N. Shiryaev. — New York: Springer, 2016. — 713 p.

[86] **Slavakis K.** Online learning in reproducing kernel Hilbert spaces / K. Slavakis, P. Bouboulis, S. Theodoridis // Academic Press Library in Signal Processing. — 2014. — Vol. 1. — P. 883–987. — DOI: 10.1016/B978-0-12-396502-8.00016-1.

[87] **Slavakis K.** Online learning in reproducing kernel Hilbert spaces / K. Slavakis, P. Bouboulis, S. Theodoridis // Academic Press Library in Signal Processing. — 2014. — Vol. 1. — P. 883–987. — DOI: 10.1016/B978-0-12-396502-8.00016-1.

[88] **Sriperumbudur B. K.** Universality, characteristic kernels and RKHS embedding of measures / B. K. Sriperumbudur, K. Fukumizu, G. R. G. Lanckriet // Journal of Machine Learning Research. — 2011. — Vol. 12, no. 70. — P. 2389–2410.

[89] **Streeter M.** Less regret via online conditioning / M. Streeter, H. B. McMahan // arXiv preprint. — 2010. — arXiv:1002.4862. — URL: https://arxiv.org/abs/1002.4862 (дата обращения: 15.01.2025).

[90] **Sun Y.** Adaptive Kernel Methods for Non-Ergodic Systems / Y. Sun, G. B. Giannakis // IEEE Transactions on Information Theory. — 2024. — Vol. 70, no. 2. — P. 1123–1141. — DOI: 10.1109/TIT.2023.3328798.

[91] **Syrgkanis V.** Fast convergence of regularized learning in games / V. Syrgkanis, A. Agarwal, H. Luo, R. E. Schapire // Advances in Neural Information Processing Systems 28 (NIPS 2015). — 2015. — P. 2989–2997.

[92] **Tedrake R.** Underactuated Robotics / R. Tedrake. — Cambridge: MIT Press, 2021. — 794 p.

[93] **Tweedie R. L.** Markov chains: structure and applications / R. L. Tweedie // Handbook of Statistics: Stochastic Processes: Theory and Methods / eds. C. R. Rao, D. N. Shanbhag. — Elsevier, 2001. — Vol. 19. — P. 817–851. — DOI: 10.1016/S0169-7161(01)19025-5.

[94] **Tsybakov A. B.** Optimal rates of aggregation / A. B. Tsybakov // Learning Theory and Kernel Machines: Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop (COLT/Kernel 2003). — 2003. — P. 303–313. — DOI: 10.1007/978-3-540-45167-9_23.

[95] **University of California, Irvine.** UCI Machine Learning Repository [Электронный ресурс]. — 2023. — URL: `https://archive.ics.uci.edu/` (дата обращения: 15.11.2023).

[96] **Van Vaerenbergh S.** Online regression with kernels / S. Van Vaerenbergh, I. Santamaría // Regularization, Optimization, Kernels, and Support Vector Machines. — New York: Chapman and Hall/CRC, 2014. — P. 477–501.

[97] **Vovk V. G.** Aggregating strategies / V. G. Vovk // Proceedings of the Third Annual Workshop on Computational Learning Theory (COLT 1990). — 1990. — P. 371–383.

[98] **Vovk V.** Competitive on-line statistics / V. Vovk // International Statistical Review. — 2001. — Vol. 69, no. 2. — P. 213–248. — DOI: 10.1111/j.1751-5823.2001.tb00457.x.

[99] **Wainwright M. J.** High-dimensional statistics: A non-asymptotic viewpoint / M. J. Wainwright. — Cambridge: Cambridge University Press, 2019. — 552 p.

[100] **Wang Z.** Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training / Z. Wang, K. Crammer, S. Vucetic // Journal of Machine Learning Research. — 2012. — Vol. 13, no. 1. — P. 3103–3131.

[101] **Wintenberger O.** Optimal learning with Bernstein online aggregation / O. Wintenberger // Machine Learning. — 2017. — Vol. 106, no. 1. — P. 119–141. — DOI: 10.1007/s10994-016-5587-3.

[102] **Zhang L.** Adaptive online learning in dynamic environments / L. Zhang, S. Lu, Z.-H. Zhou // Advances in Neural Information Processing Systems 31 (NeurIPS 2018). — 2018. — P. 1323–1333.

[103] **Zhang H.** Online sequential ELM algorithm with forgetting factor for real applications / H. Zhang, S. Zhang, Y. Yin // Neurocomputing. — 2017. — Vol. 261. — P. 144–152. — DOI: 10.1016/j.neucom.2017.03.071.

[104] **Zhang K.** Data-Driven Model Predictive Control via Operator Learning / K. Zhang, N. Matni // American Control Conference (ACC 2024). — 2024. — P. 412–419. — DOI: 10.23919/ACC53348.2024.10594321.

[105] **Zhao P.** Adaptivity and non-stationarity: Problem-dependent dynamic regret for online convex optimization / P. Zhao, Y.-J. Zhang, L. Zhang, Z.-H. Zhou // Journal of Machine Learning Research. — 2024. — Vol. 25, no. 98. — P. 1–52.

[106] **Ziemann I. M.** Single trajectory nonparametric learning of nonlinear dynamics / I. M. Ziemann, H. Sandberg, N. Matni // Proceedings of Thirty Fifth Conference on Learning Theory (COLT 2022). — 2022. — P. 3333–3364.

[107] **Zinkevich M.** Online convex programming and generalized infinitesimal gradient ascent / M. Zinkevich // Proceedings of the 20th International Conference on Machine Learning (ICML 2003). — 2003. — P. 928–935.

[108] **Боровков А. А.** Теория вероятностей / А. А. Боровков. — М.: Наука, 1986. — 432 с.

[109] **Гуртовая О. В.** Алгоритм Вовка-Азури-Вармута для аппроксимации условного математического ожидания марковского процесса по одной траектории / О. В. Гуртовая, Д. Б. Рохлин // XIX Владикавказская молодежная математическая школа: тезисы докладов. — Владикавказ, 2024. — С. 48–50.

[110] **Гуртовая О. В.** Об аппроксимации решения периодической одномерной квадратичной задачи вариационного исчисления в режиме онлайн с неизвестным внешним воздействием / О. В. Гуртовая // St. Petersburg Youth Meeting on Probability and Mathematical Physics: тезисы докладов. — Санкт-Петербург, 2024. — С. 6–7.

[111] **Гуртовая О. В.** Мультиядерная онлайн-оптимизация: двойной алгоритм Вовка-Азури-Вармута, основанный на использовании случайных признаков Фурье / О. В. Гуртовая, Д. Б. Рохлин // Международная научная конференция «Порядковый анализ и смежные вопросы математического моделирования, XVIII: Теория операторов и дифференциальные уравнения»: тезисы докладов. — РСО-А, Дзинага, 2025. — С. 80–82.

[112] **Гуртовая О. В.** О двойном алгоритме VAW с масштабированием для многоядерной онлайн-линейной регрессии / О. В. Гуртовая // Программная инженерия. — 2025. — (Принято к публикации).

[113] **Гуртовая О. В.** Свидетельство о государственной регистрации программы для ЭВМ. Программная реализация трёхуровневого алгоритма Вовка-Азури-Вармута / О. В. Гуртовая. — № 2025680750; заявл. 21.07.2025 ; опубл. 08.08.2025 (Рос. Федерация).

[114] **Рохлин Д. Б.** Алгоритм Вовка-Азури-Вармута для аппроксимации условного математического ожидания марковского процесса по одной траектории / Д. Б. Рохлин, О. В. Гуртовая // Всероссийская научно-практическая конференция «Математика, Информатика, Компьютерные науки, Моделирование, Образование»: сб. науч. тр. — Симферополь, 2024. — С. 37–43.

[115] **Цлаф Л. Я.** Вариационное исчисление и интегральные уравнения / Л. Я. Цлаф. — М.: Наука, 1970. — 208 с.

Приложение А. Листинги и описание предложенных алгоритмов

А.1. Реализация алгоритма VAW с использованием случайных признаков Фурье

Функция `run_VAW(x_data, y_data)` реализует численный метод Вовка-Азури-Вармута с использованием метода случайных признаков Фурье для модели непараметрической регрессии на марковских данных. На вход функция принимает массив признаков `x_data` размерности (n, d) и массив целевых значений `y_data` размерности $(n,)$, где n — количество наблюдений, d — размерность признакового пространства.

Инициализация алгоритма начинается с генерации параметров случайных признаков Фурье. Формируется матрица ω размерности (m, d), где $m =$ `n_components`, элементы которой представляют собой независимые реализации стандартной нормальной случайной величины $\mathcal{N}(0, 1)$. Одновременно генерируется вектор b размерности $(m,)$, компоненты которого распределены равномерно на интервале $[-\pi, \pi]$.

Преобразование исходных данных в пространство случайных признаков осуществляется вычислением матрицы X по формуле:

$$X = \big[\cos(x\omega^T + b),\ \sin(x\omega^T + b)\big]$$

что дает матрицу размерности $(n, 2m)$. Далее реализуется сам численный метод VAW(6) .

Функция `run_VAW(x_data, y_data)` возвращает усреднённый по всем итерациям вектор весов $\hat{w} = \frac{1}{n}\sum_{t=1}^{n} w[t]$, а также параметры ω и b. Данная реализация позволяет эффективно аппроксимировать сложные нелинейные зависимости благодаря комбинации метода случайных признаков и процедуры онлайн оптимизации.

Импорт библиотек и инициализация глобальных переменных:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error as MSE
from sklearn.neural_network import MLPRegressor

n_components=40
```

Определение алгоритма VAW со случайными признаками Фурье:

```
def run_VAW(x_data, y_data):
    np.random.seed(2)
    omega = np.random.normal(loc=0, scale=1, size=(n_components, d)
       )
    b = np.random.uniform(-np.pi, np.pi, n_components)
    lam = 1
    X = np.hstack((np.cos(x_data@omega.T + b),np.sin(x_data@omega.T
        + b)))
    n = X.shape[0]

    A=np.zeros((X.shape[1],X.shape[1]))
    S=lam*np.eye(X.shape[1])
    Z=np.zeros(X.shape[1])
    A=X[:n][0].reshape(-1,1)@X[:n][0].reshape(1,-1)
    S+=A
    w=np.zeros((n, X.shape[1]))
    for t in range(n - 1):
        Z+=y_data[:n][t]*X[:n][t]
        A=X[:n][t+1].reshape(-1,1)@X[:n][t+1].reshape(1,-1)
        S+=A
        w[t+1]=np.linalg.solve(S, Z)
    w_hat=np.mean(w,axis=0)
    return w_hat, omega, b
```

A.3. Реализация алгоритма VAW^2

Данный программный комплекс предназначен для экспериментального исследования численного метода VAW^2 на задаче регрессии с использованием ансамбля случайных признаков. Комплекс реализует двухуровневую структуру

обучения, где на первом уровне генерируются экспертные предсказания с помощью различных ядерных преобразований, а на втором уровне применяется алгоритм VAW для агрегации этих предсказаний.

В качестве примера работы алгоритма VAW^2 был выбран реальный набор данных Airfoil. На первом этапе производится загрузка и предварительная обработка данных из файла `airfoil_self_noise.dat`. Целевой вектор Y и матрица признаков X нормализуются для обеспечения численной стабильности алгоритма.

Далее формируется ансамбль ядерных преобразований, включающий гауссовские и лапласовские ядра с различными параметрами γ.

Главная составляющая программного комплекса реализована в функции `vaw_forecaster`, которая применяет алгоритм VAW с использованием формулы Шермана-Моррисона для эффективного обновления обратной матрицы ковариации.

Экспериментальная часть включает пять независимых запусков алгоритма для генерации случайных признаков. Для каждого запуска вычисляются предсказания алгоритма VAW^2 и строится график кумулятивной среднеквадратической ошибки. Дополнительно исследуется влияние урезанных экспертных предсказаний на качество агрегации.

Финальная оценка качества работы алгоритма производится путем усреднения результатов по всем запускам. Визуализация включает график зависимости MSE от номера итерации и график распределения весов, назначенных алгоритмом различным ядерным преобразованиям. Комплекс обеспечивает воспроизводимость результатов и позволяет исследовать влияние различных параметров алгоритма на качество предсказаний.

Импорт библиотек и инициализация глобальных переменных

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error as MSE
from copy import copy

data = np.genfromtxt('airfoil_self_noise.dat')
```

```
X = data[:,:-1]
Y = data[:,-1:]

# Normalizing the target vector Y
Y = (Y / (np.max(Y) - np.min(Y))) - np.min(Y) / (np.max(Y) - np.min
   (Y))

# Normalizing the feature matrix X
M, N = X.shape
X_norms = np.linalg.norm(X, axis=1) #Calculate the norms of each
   row
X = X / np.max(X_norms) # Divide each row by the maximum norm

# Creating lists of gammas and kernels for feature transformation
gamma = []
kernel_list = []
num_rbf = 51
for i in range(num_rbf):
    gamma.append(10 ** (4 * (i / 50) - 2))
    kernel_list.append('Gaussian')
num_lap = 25
for i in range(num_lap):
    gamma.append(10 ** ((i / 6) - 2))
    kernel_list.append('Laplacian')
gamma = np.array(gamma)
n_components = 50  # Number of random features per kernel
P = num_rbf + num_lap  # Total number of kernels

# Initialize lists to store MSE, predictions and weights for VAW^2
mse_vaw2 = []
mse_vaw2_trunc = []
cc_predictions_vaw2 = np.zeros((5, Y.shape[0]))
weights_vaw2 = np.zeros((5, P))
cc_mse_vaw2 = np.zeros((5, Y.shape[0]))
```

Определение функции для генерации случайных признаков

```
def generate_random_features_dict_ran(X, ran_feature, gamma,
   kernel_list):
    """
    Generates random features using Fourier features with given
```

```
        kernels and gammas.

    Args:
        X (np.ndarray): Feature matrix.
        ran_feature (np.ndarray): Random feature matrix.
        gamma (np.ndarray): Gamma values for kernels.
        kernel_list (list): List of kernel names.

    Returns:
        dict: Dictionary of random features for each kernel.
    """
    M, N = X.shape
    _, n_components, b = ran_feature.shape

    random_features = {}

    for i, kernel_type in enumerate(kernel_list):
        features = np.zeros((M, n_components * 2))  # Features for
            current kernel
        for j in range(M):
            X_f = X[j:j + 1, :].dot(ran_feature[:, :, i])
            features[j, :] = (1 / np.sqrt(n_components)) * np.
                concatenate((np.sin(X_f), np.cos(X_f)), axis=1)
        random_features[f"{kernel_type}_{i}"] = features

    return random_features
```

Определение алгоритма VAW

```
def vaw_forecaster(features, target, lambda_reg=1, weights=None):

    """
    Vovk-Azoury-Warmuth forecaster with closed-form solution using
        Sherman-Morrison update.

    Args:
        features (np.ndarray): Feature matrix, where each row is a
            feature vector (shape: n_samples, n_features).
        target (np.ndarray): Target vector, with target values (
            shape: n_samples).
        lambda_reg (float): Regularization parameter (default: 1.0)
```

```
        .
    weights (np.ndarray): Initial weights vector (default: None
        , initialized to zeros).

Returns:
    tuple: predictions, rmse_history, updated weights.
"""

n_samples, n_features = features.shape
predictions = []
rmse_history = []  # To store RMSE at each horizon
squared_errors_cumulative = 0  # Cumulative squared errors

# Initialize weights if not provided, else use provided weights
if weights is None:
    weights = np.zeros(n_features)

# Initialize the inverse of (lambda * I + sum of zi zi^T),
   using first the value lambda * I
reg_matrix_inverse = (1/lambda_reg) * np.eye(n_features) # S^-1

# Initialize w, which is the result of sum(yizi)
sum_yizi = np.zeros(n_features)

for t in range(n_samples):
    zt = features[t]  # Current feature vector
    yt = target[t]    # Current target value

    # Compute the prediction, since x_t = reg_matrix_inverse *
       sum_yizi
    if t==0:
         prediction = 0
    else:
        prediction = np.dot(sum_yizi, reg_matrix_inverse @ zt)
           # This is xt^T * zt

    predictions.append(prediction)

    # Compute squared error
    squared_error = (yt - prediction) ** 2
```

```
        squared_errors_cumulative += squared_error

        # Calculate RMSE up to the current horizon
        rmse_t = np.sqrt(squared_errors_cumulative / (t + 1))
        rmse_history.append(rmse_t)

        # Update sum_yizi
        sum_yizi += yt*zt

        # Update the inverse matrix using Sherman-Morrison formula:
        # S_t = S_{t-1} + z_t z_t^T
        # S_t^-1 = S_{t-1}^-1 - (S_{t-1}^-1 @ z_t @ z_t^T @ S_{t
           -1}^-1) / (1 + z_t^T @ S_{t-1}^-1 @ z_t)
        zt = zt.reshape(-1, 1) # Convert to a column vector
        numerator = reg_matrix_inverse @ zt @ zt.T @
           reg_matrix_inverse
        denominator = 1 + zt.T @ reg_matrix_inverse @ zt
        reg_matrix_inverse = reg_matrix_inverse - (numerator/
           denominator[0,0])

    # Get the last weight vector
    if n_samples == 0:
        weights = np.zeros(n_features)
    else:
        weights = reg_matrix_inverse @ sum_yizi

    return predictions, rmse_history, weights
```

Генерация экспертных предсказаний

```
all_predictions_df_lst = []
for cc in range(5):
    np.random.seed(cc)
    ran_feature = np.zeros((N, n_components, gamma.shape[0]))
    for i in range(num_rbf):
        ran_feature[:, :, i] = np.random.randn(N, n_components) *
           np.sqrt(1 / gamma[i])
    for i in range(num_lap):
        ran_feature[:, :, i + num_rbf] = np.random.standard_cauchy
           ((N, n_components)) * (1 / gamma[i + num_rbf])
    fourier_features = generate_random_features_dict_ran(X,
```

```
        ran_feature, gamma, kernel_list)
    all_predictions_df = pd.DataFrame()
    rmse_individual = []
    for kernel_name, features in fourier_features.items():
        predictions, rmse_history, _ = vaw_forecaster(
            fourier_features[kernel_name],
            Y,
            lambda_reg=1.
        )
        rmse_individual.append(rmse_history[-1])
        predictions_df = pd.DataFrame(predictions, columns=[
            kernel_name])
        all_predictions_df = pd.concat([all_predictions_df,
            predictions_df], axis=1)
    all_predictions_df_lst.append(all_predictions_df)

# Truncating expert predictions to the range of Y
all_predictions_df_lst_trunc = [df.map(lambda u: min(max(u, 0), 1))
    for df in copy(all_predictions_df_lst)]
```

Применение алгоритма VAW^2

```
# VAW^2 algorithm
for cc in range(5):
    predictions, _, weights_vaw2[cc] = vaw_forecaster(np.array(
        all_predictions_df_lst[cc]), Y, lambda_reg=1.)
    cc_predictions_vaw2[cc] = predictions
    mse_vaw2.append(MSE(predictions, Y))

# VAW^2 algorithm (with truncated expert predictions)
for cc in range(5):
    predictions, _, weights_vaw2[cc] = vaw_forecaster(np.array(
        all_predictions_df_lst_trunc[cc]), Y, lambda_reg=1.)
    mse_vaw2_trunc.append(MSE(predictions, Y))
```

Расчёт и вывод результатов

```
# Calculate cumulative MSE for VAW^2
for cc in range(5):
    for t in range(Y.shape[0]):
        cc_mse_vaw2[cc, t] = 1. / (t + 1) * MSE(cc_predictions_vaw2
            [cc][:t + 1], Y[:t + 1])
```

```
# Calculate mean cumulative MSE across all repetitions
cc_mse_vaw2 = np.mean(cc_mse_vaw2, axis=0)

# Print final MSE values
print('MSE␣of␣VAW^2␣is␣', np.mean(mse_vaw2))
print('MSE␣of␣VAW^2␣with␣truncated␣expert␣predictions␣is␣', np.mean
    (mse_vaw2_trunc))

# Plotting cumulative MSE for VAW^2
start_index = 200
fig1 = plt.figure(1)
ax = fig1.add_subplot()
plt.title('Airfoil')
plt.xlabel('Iteration␣number')
plt.ylabel('MSE')
ax.plot(np.arange(start_index, Y.shape[0]), cc_mse_vaw2[start_index
    :], 'r', label=r'VAW$^2$')
plt.legend()
plt.show()

# Plotting weights of VAW^2
fig2 = plt.figure(2)
ax = fig2.add_subplot()
P = num_rbf + num_lap
ax.plot(np.arange(1, P + 1), np.mean(weights_vaw2, axis = 0), 'r',
    label=r'VAW$^2$')
plt.legend()
plt.xlabel('Kernel')
plt.ylabel('Weights')
plt.title('Airfoil')
plt.show()
```

A.3. Реализация алгоритма S-VAW^2

Программный комплекс реализует программу для ЭВМ «Программная реализация трёхуровневого алгоритма Вовка-Азури-Вармута» на языке программирования Python. Для этой программы было получено свидетельство о государ-

ственной регистрации (см. приложение Б.). Данная программа предназначена для моделирования временных рядов и решения различных регрессионных задач. Комплекс основан на современном ансамблевом подходе, использующем трёхуровневую систему комбинирования экспертных прогнозов с применением методов многоядерного преобразования признаков и различных техник масштабирования данных.

Программный комплекс реализован в виде класса `SVAW2`, предназначенного для моделирования сложных зависимостей в данных с использованием трёхуровневого численного метода Вовка-Азури-Вармута. При инициализации класса устанавливаются ключевые параметры алгоритма, включая количество случайных признаков `n_components`, словарь используемых ядер `dict_of_kernels` и коэффициент регуляризации `lambda_reg`.

В основе комплекса лежат численные методы обработки данных, включая генерацию параметров для различных ядерных преобразований (гауссовских и лапласовских ядер) и создание комбинаций методов масштабирования данных. Моделирование процессов масштабирования осуществляется через поддержку различных стратегий: для матрицы признаков X доступны варианты `StandardScaler`, `MinMaxScaler`, `RobustScaler`, а также возможность работы без масштабирования; для целевой переменной Y применяется `MinMaxScaler` и отсутствие масштабирования.

Численный метод преобразования признаков реализован через генерацию случайных проекционных матриц в методе `generate_random_features_and_dict`, осуществляющих преобразование исходных данных в признаки Фурье для каждого типа ядра.

Центральным вычислительным блоком программного комплекса является реализация численного метода онлайн-оптимизации Вовка-Азури-Вармута в методе `_vaw_forecaster`. Данный алгоритм осуществляет итеративное обновление весовых коэффициентов путём последовательной обработки каждого наблюдения с использованием эффективных матричных вычислений.

Моделирование прогнозной функции организовано по трёхуровневой схеме. На первом уровне для каждой комбинации методов масштабирования и каждого типа ядра формируется отдельный эксперт VAW. На втором уровне числен-

ный метод агрегации прогнозов объединяет результаты экспертов первого уровня. На третьем уровне осуществляется финальное моделирование оптимальной комбинации прогнозов через взвешенное среднее с вычислением весовых коэффициентов.

Численный метод прогнозирования в методе `predict` воспроизводит структуру обучения, последовательно применяя цепочку преобразований: генерацию признаков Фурье, вычисление прогнозов экспертов первого уровня и их агрегацию на втором и третьем уровнях модели.

Для верификации работоспособности программного комплекса используется демонстрационный пример с набором данных diamonds. Моделирование включает предварительную обработку данных, разделение на обучающую и тестовую выборки, обучение модели и вычисление среднеквадратичной ошибки между спрогнозированными и фактическими значениями, что подтверждает эффективность реализованных численных методов.

Импорт библиотек

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error as MSE
from sklearn.preprocessing import LabelEncoder, StandardScaler,
   OneHotEncoder, MinMaxScaler, RobustScaler
```

Инициализация алгоритма S-VAW^2

```
class SVAW2:
def __init__(self, n_components=50,
             dict_of_kernels={'Gaussian': 51, 'Laplacian': 25}):
    self.n_components = n_components
    self.lambda_reg = 1
    self.dict_of_kernels = dict_of_kernels

    # Initialize kernel parameters
    self.gamma, self.kernel_list, self.num_rbf, self.num_lap, self.
       P = self._generate_kernel_params(dict_of_kernels)

    self.final_predictions = None
    self.final_weights = None
```

```
        # List of X-scaler options, including None (no scaling)
        self.x_scalers_options = [
            ('StandardScaler', StandardScaler),
            ('MinMaxScaler', MinMaxScaler),
            ('RobustScaler', RobustScaler),
            ('NoScaler', None)
        ]
        # Y-scaler class option
        self.y_scaler_option = MinMaxScaler

        # Stores trained scalers and model components for each
           combination
        self.scaler_combinations = []

        # Dictionary to store ALL trained scalers, random features, and
            weights for each combination
        self.trained_scalers = {}

        # Set random seed for reproducibility of random features
        np.random.seed(1)

        # Generate all possible X and Y scaler combinations
        self._generate_scaler_combinations()

    def _generate_kernel_params(self, dict_of_kernels):
        """
        Generates kernel parameters (gamma values and kernel types).
        """
        gamma = []
        kernel_list = []
        num_rbf, num_lap = 0, 0
        P = 0 # Total number of kernels
        for kernel, num_kernel in dict_of_kernels.items():
            P += num_kernel
            if num_kernel > 0:
                gamma_values = np.logspace(-2, 2, num_kernel)
                if kernel == 'Gaussian':
                    num_rbf = num_kernel
                    gamma.extend(gamma_values)
```

```
                kernel_list.extend(['Gaussian'] * num_kernel)
            elif kernel == 'Laplacian':
                num_lap = num_kernel
                gamma.extend(gamma_values)
                kernel_list.extend(['Laplacian'] * num_kernel)
            else:
                raise ValueError(f"Unknown kernel type: {kernel}")
    return np.array(gamma), kernel_list, num_rbf, num_lap, P

def _generate_scaler_combinations(self):
    """
    Creates a list of all X and Y scaler combinations.
    """
    y_scaler_templates = [
        ('Yraw', None), # No scaling for Y
        ('Yscaled', self.y_scaler_option) # MinMaxScaler for Y
    ]

    for x_name, x_scaler_class in self.x_scalers_options:
        for y_name, y_scaler_class in y_scaler_templates:
            # Create X-scaler instance or None
            current_x_scaler_instance = x_scaler_class() if
               x_scaler_class is not None else None
            # Create Y-scaler instance or None
            current_y_scaler_instance = y_scaler_class() if
               y_scaler_class is not None else None

            combo_full_name = f"{x_name}_{y_name}"
            self.scaler_combinations.append((combo_full_name,
               current_x_scaler_instance, current_y_scaler_instance
               ))

def generate_random_features_and_dict(self, X):
    """
    Generates random projection matrices and corresponding Fourier
       features for X.
    """
    M, N = X.shape
    ran_feature_matrix = np.zeros((N, self.n_components, self.P))
    random_features = {}
```

```
    kernel_counters = {'Gaussian': 0, 'Laplacian': 0}

    for i, kernel_type in enumerate(self.kernel_list):
        features = np.zeros((M, self.n_components * 2))
        gamma = self.gamma[i]

        if kernel_type == 'Gaussian':
            ran_feature_matrix[:, :, i] = np.random.randn(N, self.
               n_components) * np.sqrt(1 / gamma)
        elif kernel_type == 'Laplacian':
            laplacian_index = i - self.num_rbf
            if not (0 <= laplacian_index < self.num_lap):
                raise IndexError("Laplacian index out of bounds")
            ran_feature_matrix[:, :, i] = np.random.standard_cauchy
               ((N, self.n_components)) * (1 / gamma)
        else:
            raise ValueError(f"Unknown kernel type: {kernel_type}")

        # Compute Fourier features for the current kernel
        for j in range(M):
            X_f = X[j:j + 1, :].dot(ran_feature_matrix[:, :, i])
            features[j, :] = (1 / np.sqrt(self.n_components)) * np.
               concatenate((np.sin(X_f), np.cos(X_f)), axis=1)

        random_features[f"{kernel_type}_{kernel_counters[
           kernel_type]}"] = features
        kernel_counters[kernel_type] += 1

    return random_features, ran_feature_matrix

def _vaw_forecaster(self, features, target):
    """
    Implements the Vovk-Azoury-Warmuth (VAW) online linear
       regressor.
    """
    n_samples, n_features = features.shape
    predictions = []

    reg_matrix_inverse = (1 / self.lambda_reg) * np.eye(n_features)
    sum_yizi = np.zeros(n_features)
```

```
        for t in range(n_samples):
            zt = features[t]
            yt = target[t]

            if t == 0:
                prediction = 0
            else:
                prediction = np.dot(sum_yizi, reg_matrix_inverse @ zt)
            predictions.append(prediction)

            sum_yizi += yt * zt

            zt_reshaped = zt.reshape(-1, 1)
            numerator = reg_matrix_inverse @ zt_reshaped @ zt_reshaped.
               T @ reg_matrix_inverse
            denominator = 1 + zt_reshaped.T @ reg_matrix_inverse @
               zt_reshaped

            if denominator[0, 0] == 0:
                print("Warning: Denominator in VAW update is zero.
                   Skipping update.")
            else:
                reg_matrix_inverse = reg_matrix_inverse - (numerator /
                   denominator[0, 0])

        weights = np.zeros(n_features)
        if n_samples > 0:
            weights = reg_matrix_inverse @ sum_yizi

        return np.array(predictions), weights

    def fit(self, X, Y):
        """
        Fits the SVAW2 model on input data X and Y.
        """
        Y_original_np = Y.values if isinstance(Y, pd.Series) else Y
        Y_2d_original = Y_original_np.reshape(-1, 1)

        combined_predictions_for_final_layer = pd.DataFrame()
```

```
train_size = int(X.shape[0]*0.75)
# Iterate through each scaler combination (Level 2)
for combo_name, x_scaler_instance, y_scaler_instance in self.
   scaler_combinations:
    # 1. Scale X for the current combination
    if x_scaler_instance is not None:
        x_scaler_instance.fit(X[:train_size])
        X_scaled = x_scaler_instance.transform(X)
        self.trained_scalers[f"{combo_name}_x_scaler"] =
           x_scaler_instance
    else:
        X_scaled = X.copy()
        self.trained_scalers[f"{combo_name}_x_scaler"] = None

    # 2. Scale Y for the current combination
    if y_scaler_instance is not None:
        y_scaler_instance.fit(Y_2d_original[:train_size])
        Y_scaled = y_scaler_instance.transform(Y_2d_original).
           flatten()
        self.trained_scalers[f"{combo_name}_y_scaler"] =
           y_scaler_instance
    else:
        Y_scaled = Y_original_np.copy()
        self.trained_scalers[f"{combo_name}_y_scaler"] = None

    # 3. Level 1 VAW: Generate random features and train
       experts
    fourier_features_for_combo, ran_feature_for_combo = self.
       generate_random_features_and_dict(X_scaled)
    self.trained_scalers[f"{combo_name}_ran_feature"] =
       ran_feature_for_combo

    experts_weights_for_combo = np.zeros((self.P, self.
       n_components * 2))
    experts_online_predictions_df = pd.DataFrame()

    i = 0
    for kernel_name, features in fourier_features_for_combo.
       items():
        preds_online, experts_weights_for_combo[i] = self.
```

```
                _vaw_forecaster(features, Y_scaled)
            experts_online_predictions_df = pd.concat([
                experts_online_predictions_df,
                pd.DataFrame(preds_online, columns=[f"{combo_name}_
                    {kernel_name}"])
            ], axis=1)
            i += 1
        self.trained_scalers[f"{combo_name}_experts_weights"] =
            experts_weights_for_combo

        # 4. Level 2 VAW: Combine expert predictions for the
            current combination
        level2_preds_scaled, weights_vaw2_for_combo = self.
            _vaw_forecaster(np.array(experts_online_predictions_df),
             Y_scaled)
        self.trained_scalers[f"{combo_name}_weights_vaw2"] =
            weights_vaw2_for_combo

        # 5. Inverse scale Level 2 predictions if Y was scaled
        if y_scaler_instance is not None:
            level2_preds_unscaled = y_scaler_instance.
                inverse_transform(level2_preds_scaled.reshape(-1, 1)
                ).flatten()
        else:
            level2_preds_unscaled = level2_preds_scaled.copy()

        combined_predictions_for_final_layer = pd.concat([
            combined_predictions_for_final_layer,
            pd.DataFrame(level2_preds_unscaled, columns=[combo_name
                ])
        ], axis=1)

    # 6. Level 3 VAW: Final combination of all 8 predictions
    self.final_predictions, self.final_weights = self.
        _vaw_forecaster(
         np.array(combined_predictions_for_final_layer),
            Y_original_np
    )

def predict(self, X):
```

```
    """
    Makes predictions on new data X.
    """
    combined_predictions_for_final_layer_predict = pd.DataFrame()

    # Iterate through each scaler combination, using stored trained
        objects
    for combo_name, _, _ in self.scaler_combinations:
        # Retrieve trained scalers and components for the current
            combination
        x_scaler_trained = self.trained_scalers.get(f"{combo_name}
            _x_scaler")
        y_scaler_trained = self.trained_scalers.get(f"{combo_name}
            _y_scaler")
        ran_feature_for_combo = self.trained_scalers.get(f"{
            combo_name}_ran_feature")
        experts_weights_for_combo = self.trained_scalers.get(f"{
            combo_name}_experts_weights")
        weights_vaw2_for_combo = self.trained_scalers.get(f"{
            combo_name}_weights_vaw2")

        if any(item is None for item in [ran_feature_for_combo,
            experts_weights_for_combo, weights_vaw2_for_combo]):
            # If basic components are missing, this combination
                cannot predict correctly.
            combined_predictions_for_final_layer_predict = pd.
                concat([
                combined_predictions_for_final_layer_predict,
                pd.DataFrame(np.zeros(len(X)), columns=[combo_name
                    ])
            ], axis=1)
            continue

        # 1. Scale X for prediction
        if x_scaler_trained is not None:
            X_scaled = x_scaler_trained.transform(X)
        else:
            X_scaled = X.copy()

        M, N = X_scaled.shape
```

```
experts_predictions_from_features = np.zeros((M, self.P))
random_features_list = []

# Generate Fourier features for each kernel
for i, kernel_type in enumerate(self.kernel_list):
    features = np.zeros((M, self.n_components * 2))
    for j in range(M):
        X_f = X_scaled[j:j + 1, :].dot(
           ran_feature_for_combo[:, :, i])
        features[j, :] = (1 / np.sqrt(self.n_components)) *
            np.concatenate(
            (np.sin(X_f), np.cos(X_f)), axis=1
        )
    random_features_list.append(features)

# Calculate predictions from each expert
for p in range(self.P):
    experts_predictions_from_features[:, p] = np.dot(
       random_features_list[p], experts_weights_for_combo[p
       ])

# 2. Apply Level 2 VAW weights. Predictions will be in the
   Y_scaled scale (if Y was scaled)
level2_preds_scaled = experts_predictions_from_features @
   weights_vaw2_for_combo

# 3. Inverse scale Level 2 predictions if Y was scaled
if y_scaler_trained is not None:
    level2_preds_unscaled = y_scaler_trained.
       inverse_transform(level2_preds_scaled.reshape(-1, 1)
       ).flatten()
else:
    level2_preds_unscaled = level2_preds_scaled.copy()

combined_predictions_for_final_layer_predict = pd.concat([
    combined_predictions_for_final_layer_predict,
    pd.DataFrame(np.zeros(len(X)) if level2_preds_unscaled
       is None else level2_preds_unscaled, columns=[
       combo_name])
], axis=1)
```

```
    # 4. Final prediction - combine all 8 predictions using
       final_weights
    final_pred = combined_predictions_for_final_layer_predict.
       values @ self.final_weights

    return final_pred
```

Пример запуска алгоритма S-VAW^2 на загруженных данных

```
#Initializating data
df = pd.DataFrame(np.genfromtxt('airfoil_self_noise.dat'))
target = 5
X = df.drop(target, axis = 1).values
Y = df[target].values

train_size = int(df.shape[0]*0.75)

# Split data into training and testing sets
X_test = X[train_size:]
Y_test = Y[train_size:]

# Train the model
predictor = SVAW2()
predictor.fit(X, Y)

#Make predictions on test
preds = predictor.predict(X_test)

# Print MSE
print('MSE is ', MSE(preds, Y_test))
```

Приложение Б. Свидетельство о регистрации программы для ЭВМ

Свидетельство о государственной регистрации программы для ЭВМ № 2025680750 от 08.08.2025 г. Российская Федерация. Программная реализация трёхуровневого алгоритма Вовка-Азури-Вармута для решения задач регрессии. Автор: Гуртовая Ольга Владимировна. Правообладатель: Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет».

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025680750

Программная реализация трехуровневого алгоритма Вовка-Азури-Вармута

Правообладатель: *федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет» (RU)*

Автор(ы): *Гуртовая Ольга Владимировна (RU)*

Заявка № 2025669850
Дата поступления 21 июля 2025 г.
Дата государственной регистрации
в Реестре программ для ЭВМ *08 августа 2025 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов