

pecialityRelationTXTОбласть исследования.



Федеральное государственное бюджетное образовательное учреждение
высшего образования «Белгородский государственный технологический
университет им. В.Г. Шухова» (БГТУ им. В. Г. Шухова)



На правах рукописи

Каратач Сергей Александрович

**Разработка методов интеллектуального анализа данных
на основе нечетких систем при несинглтонной
фазификации**

Специальность 2.3.8 —
«Информатика и информационные процессы»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
кандидат технических наук, профессор
Синюк Василий Григорьевич

Белгород — 2025

Оглавление

	Стр.
Введение	4
Глава 1. Системы нечеткого вывода как метод анализа зашумленных или неопределенных входных данных	10
1.1 Нейро-нечеткие системы	11
1.2 Методы нечеткого вывода	12
1.3 Несинглтонная фазификация	14
1.4 Нечеткое значение истинности	18
1.4.1 Вычисление нечеткого значения истинности, когда функции принадлежности высказываний задаются гауссовыми функциями	25
1.5 Применение нечетких моделей для прогнозирования временных рядов	25
1.5.1 Формирование знаний в нечетких системах в контексте задачи прогнозирования временных рядов	26
1.5.2 Анализ способов построения нечетких временных моделей	28
1.6 Постановка задачи исследования	30
1.7 Выводы	31
Глава 2. Метод нечеткого вывода на основе нечеткого значения истинности	33
2.1 Постановка задачи нечеткого вывода	33
2.2 Вывод на основе нечеткого значения истинности	34
2.3 Вывод для систем логического типа	37
2.4 Нечеткий вывод с использованием различных методов дефазификации	38
2.4.1 Дефазификация по методу среднего центра	38
2.4.2 Дефазификация по методу центра тяжести	40
2.4.3 Дефазификация по методу центра области	43
2.4.4 Дефазификация по методу среднего максимума	44

2.5	Классификация объектов на основе нечеткого вывода с использованием нечеткого значения истинности	44
2.6	Прогнозирование временных рядов на основе нечетких систем логического типа с использованием нечеткого значения истинности	46
2.7	Выводы	46
Глава 3. Программная реализация разработанного метода нечеткого вывода с применением технологии CUDA		48
3.1	Вычисление нечеткого значения истинности посредством дискретизации	53
3.2	Алгоритм свертки НЗИ при $T_1 = \min$ и T_3 - неубывающая по всем аргументам	57
3.3	Реализация дефазификации	59
3.4	Ускорение вывода за счет фильтрации ближайших правил	64
3.5	Реализация алгоритма построения базы правил	66
3.6	Реализация обертки в Python-модуль	67
3.7	Выводы	68
Глава 4. Применение разработанной нечеткой модели для прогнозирования временных рядов в задачах экономики и финансов		71
4.1	Задача прогнозирования стоимости ценных бумаг Тайваньской биржи	72
4.2	Задача прогнозирования активности клиента по банковскому продукту	76
4.3	Выводы	81
Заключение		82
Словарь терминов		84
Список рисунков		85
Список таблиц		86

Приложение А. Текст программного кода Python-модуля нечеткого вывода с использованием технологии CUDA	87
Приложение Б. Свидетельства о государственной регистрации программ для ЭВМ	116

Введение

Высокопроизводительный интеллектуальный анализ данных дает возможность принимать обоснованные решения на основе знаний, получаемых посредством обработки данных со скоростью близкой к реальному времени. Семейство методов мягких вычислений с применением техник высокопроизводительного анализа данных открывает возможность находить закономерности и взаимосвязи в данных, содержащих неопределенность. Одним из методов мягких вычислений предназначенным для анализа неопределенных данных являются методы нечеткого моделирования.

В описанной Л. Заде теории нечеткой логики важной проблемой остается задача нечеткого логического вывода. Распространение получили подходы опирающиеся на использованием методов нечеткого вывода выработанных Э. Мамдани, П. Ларсеном, Т. Такаги, М. Сугено и Ю. Цукамато. Эти подходы, а также основанные на них производные методы, как правило, используют четкие значения входов и t -норму вместо импликации, что позволяет упростить реализацию нечеткого вывода. Однако такое упрощение приводит к несоответствию с теорией Заде, что можно выявить при рассмотрении лингвистических моделей со многими нечеткими входами, то есть когда используется несинглтонный метод фазификации.

В теории нечеткой логики нечеткий логический вывод реализуется за с помощью обобщенных нечетких правил *modus ponens* и *modus tollens* на основе *композиционного правила вывода*. При нескольких входах вычисление по данным правилом приводит к экспоненциальной зависимости вычислительной сложности от количества входов. Данное ограничение является основным препятствием для применения нечеткого логического вывода с несколькими посылками, тогда как необходимость анализа многомерных данных **является актуальной задачей, например, ...**.

Разработку нечеткого вывода с использованием несинглтонной фазификации возродил Д. Мендель. Он продемонстрировал прирост качества нечеткого моделирования с использованием фазификации типа non-singleton, например, в задаче прогнозирования временных рядов. Однако его исследования ограничены проработкой нечеткого вывода типа Мамдани и Такаги-Сугено. Кроме того Мендель строит формальные выкладки процедуры нечеткого вывода при

использовании одной и той же t -нормы, что необоснованно сужает гибкость формул вывода.

Проблемы связанные с нечетким выводом и нечетким моделированием в России изучались и прорабатывались И. А. Ходашинским, Н. Г. Ярушкина, А. И. Аверкин.

Целью данной работы является повышение производительности анализа неопределенных данных путем разработки математического и программного обеспечения на основе нечетких систем при несинглтонной фаззификации С одной стороны цель должна согласоваться с названием диссертационной работы, которое уже зафиксировано и не должно меняться, а с другой стороны по итогу достижения цели должны доказываться выносимые на защиту положения. Поэтому, возможно, стоит как-то дополнить цель?.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Провести обзор проблем и предлагаемых подходов построения и реализации нечетких систем анализа нечетких данных или данных с качественным описанием.
2. Разработать метод нечеткого вывода на основе нечеткого значения истинности для системы MISO-структуры логического типа, обеспечивающий полиномиальную вычислительную сложность.
3. Разработать метод классификации объектов с нечетким или качественным описанием атрибутов на основе метода нечеткого вывода с использованием нечеткого значения истинности.
4. Разработать нечеткую модель регрессии временных рядов с нечеткими оценками значений временной последовательности на основе метода нечеткого вывода с использованием нечеткого значения истинности.
5. Выполнить программную реализацию выработанного метода нечеткого вывода и разработанной модели регрессии временных рядов с использованием технологии параллельных вычислений CUDA, обеспечив эффективность реализации. Реализовать алгоритм построения базы правил на основе данных.
6. Применить разработанный модуль нечеткого логического вывода для высокопроизводительного анализа зашумленных данных в выбранной предметной области.

Объектом исследования являются методы высокопроизводительного анализа данных на основе нечеткой логики.

Субъектом исследования являются методы и алгоритмы нечеткого логического вывода с несинглтонной фазификацией, а также методы высокопроизводительного анализа данных на его основе.

Методология и методы исследования. В работе использованы методы теории нечетких множеств, нечетких отношений, нечеткого логического вывода и мягких вычислений, а также методы оптимизации и методы анализа данных.

Достоверность полученных результатов обеспечивается корректным применением математических методов, доказанностью выводов, результатами проведенных экспериментов и их сопоставлением с результатами экспериментов других научных групп, апробацией на научно-практических конференциях.

Научная новизна:

1. Впервые применено нечеткое значение истинности и принцип обобщения для получения выходного значения при нескольких нечетких входах в соответствии с обобщенным нечетким правилом вывода *modus ponens* для нечетких систем логического типа, в результате чего была получена новая структура базы правил: «Если *истинно*, то B_k ».
2. Разработан метод нечеткого вывода логического типа с использованием нечеткого значения истинности, имеющий полиномиальную вычислительную сложность при многих нечетких входах.
3. Разработан метод классификации для объектов с нечеткими оценками признаков на основе предложенного метода нечеткого логического вывода с использованием нечеткого значения истинности.
4. Разработан метод регрессии временных рядов с нечеткими оценками измеренных значений на основе предложенного метода нечеткого логического вывода и алгоритм построения базы правил
5. Разработан параллельный алгоритм, реализующий нечеткий вывод на основе нечеткого значения истинности

заключается в расширении класса задач анализа данных, эффективно решаемых при помощи нечеткого моделирования, соответствующего теории нечеткого вывода Л. Заде, за счет использования нечеткого значения истинности и разработанного алгоритма параллельной свертки нечетких значений истинности по входам.

заключается в разработанных на основе предложенного метода нечеткого вывода нечетких логических моделях адаптированные для задач классификации и регрессии при задании измеренных характеристиках моделируемых объектов нечеткими значениями, а также в реализованном на основе разработанных моделей нечеткого вывода программного обеспечения и результатами проведенных вычислительных экспериментов по оценке производительности этой реализации.

Основные положения, выносимые на защиту:

1. Разработан метод нечеткого логического вывода при фазификации типа non-singleton на основе нечеткого значения истинности. Разработанный метод нечеткого вывода имеет новый вид нечетких правил «Если истинно, то B_k » и обеспечивает полиномиальную вычислительную сложность при многих нечетких входах. Выполненная параллельная реализация данного метода продемонстрировала линейный рост времени работы алгоритма с увеличением количества входов.
2. Разработан метод классификации для объектов с нечеткими оценками признаков с использованием нечетких систем логического типа на основе нечеткого значения истинности.
3. Разработан параллельный алгоритм свертки НЗИ, сокращающий вычислительную сложность до $O(|V| \cdot \log n)$.
4. Выполнена программная реализация разработанного метода нечеткого вывода с применением разработанного алгоритма свертки НЗИ на основе технологии параллельного программирования CUDA. Высокая производительность реализации обеспечена за счет эффективного использования аппаратных ресурсов графического ускорителя.
5. Разработана нечеткая модель регрессии для временных рядов с нечеткими оценками неопределенности измеренных значений с использованием нечеткого вывода логического типа на основе нечеткого значения истинности. Данная нечеткая модель показала прирост качества прогнозирования временных рядов (8% по метрике sMAPE) на наборе данных Mackey-Glass по сравнению с нечетким моделированием на основе синглтонной фазификации (с точностью $\approx 40\%$), а также требуемое количество правил при логическом типа вывода (30) оказалось значительно меньше количества правил при вывода типа Мамдани (184) для достижения сопоставимого качества ($\approx 10\%$).

Содержание диссертации соответствует следующим пунктам паспорта специальности 2.3.8. Информатика и информационные процессы по следующим направлениям исследований:

- п. 4 паспорта специальности: Разработка методов и технологий цифровой обработки аудиовизуальной информации с целью обнаружения закономерностей в данных, включая обработку текстовых и иных изображений, видео контента. Разработка методов и моделей распознавания, понимания и синтеза речи, принципов и методов извлечения требуемой информации из текстов.
- п. 13 паспорта специальности: Разработка и применение методов распознавания образов, кластерного анализа, нейро-сетевых и нечетких технологий, решающих правил, мягких вычислений при анализе разнородной информации в базах данных.

Внедрение результатов диссертационного исследования. Разработанная в процессе диссертационного исследования нечеткая модель прогнозирования временных рядов была зарегистрирована как программа ЭВМ в Роспатенте и внедрена в процесс составления прогнозов активности клиентов по одному из банковских продуктов в ПАО «Сбербанк». Предложенные алгоритмы также использованы при выполнении научного проекта при поддержке РФФИ №20-07-00030 «Разработка высокопроизводительных методов интеллектуального анализа данных на основе нечеткого моделирования и создание компьютерной системы поддержки принятия решений для классификации и прогнозирования».

Апробация работы. Основные результаты работы докладывались на:

1. Международная конференция «Перспективные компьютерные и цифровые технологии» (ACDT 2021)», г. Белгород, 2021.
2. XV Международная научная конференция «Параллельные вычислительные технологии (ПаВТ) 2021», г. Волгоград, 2021.
3. XI Международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте (ИММВ-2022)», г. Коломна, 2022 г.
4. XX Национальная конференция по искусственному интеллекту с международным участием (КИИ-2022), г. Москва, 2022.
5. XVII Международная научная конференция «Параллельные вычислительные технологии (ПаВТ) 2021», г. Санкт-Петербург, 2023.

6. XXI Национальная конференция по искусственному интеллекту с международным участием (КИИ-2023), г. Смоленск, 2023.
7. X Всероссийская научно-техническая конференция «Информационные технологии в науке, образовании и производстве» (ИТНОП-2025), г. Орел, 2025.

Личный вклад. Постановка цели и задач научного исследования, планирование экспериментов, подготовка публикаций по выполненной работе проводилась совместно с научным руководителем. Автором самостоятельно разработан алгоритм параллельной свертки нечетких значений истинности, а также выполнена реализация механизма нечеткой вывода с использованием нечеткого значения истинности и модели регрессии временных рядов для систем MISO-структуры при использовании фазификации типа non-singleton, а также выполнена реализация алгоритма оптимизации для настройки параметров термов в базе правил на основе набора данных. С использованием данной реализации автором выполнены эксперименты, подтверждающие полиномиальную зависимость времени выполнения нечеткого вывода от количества входов, а также проведены эксперименты для оценки качества моделирования нечеткой логической системой с использованием фазификации типа non-singleton в задаче прогнозирования временных рядов для набора данных TWSE и набора данных из практической задачи прогнозирования транзакционной активности клиентов банка. Проведена апробация разработанных методов нечеткого вывода и нечеткого моделирования.

Публикации. Основные результаты по теме диссертации изложены в 0 печатных изданиях, 0 из которых изданы в журналах, рекомендованных ВАК.

Объем и структура работы. Диссертация состоит из введения, 4 глав, заключения и 2 приложений. Полный объём диссертации составляет 117 страниц, включая 35 рисунков и 2 таблицы. Список литературы содержит 0 наименований.

Глава 1. Системы нечеткого вывода как метод анализа зашумленных или неопределенных входных данных

Высокопроизводительный анализ данных (HPDM) позволяет обрабатывать и анализировать огромные массивы данных с использованием современных вычислительных технологий – как специализированных аппаратных средств (высокопроизводительных вычислительных кластеров, GPU ускорителей), так и оптимизированного программного обеспечения. Этот подход объединяет имеющиеся модели анализа данных с возможностями масштабируемых технологий Big Data и высокопроизводительных вычислений (HPC), что обеспечивает получение знаний на основе данных практически в реальном времени и значительно сокращает время от сбора данных до принятия обоснованных решений. Широкое промышленное применение получили программные пакеты (например, Apache Spark, NVIDIA RAPIDS), включающие традиционные модели анализа данных, а также инструменты JIT-компиляции готового программного кода (например, Numba).

Для определенных задач традиционные «жёсткие» вычисления (основанные на точных математических моделях) оказываются неэффективными или неприменимыми. Модели мягких вычислений принимают неопределенность, шум в данных и частичную истинность, что позволяет находить решения в условиях реального мира, где информация часто неполна или противоречива. Вместо строгих математических моделей используются эвристические и адаптивные методы, имитирующие человеческое мышление (например, лингвистические переменные вида «высокая температура»). Мягкие вычисления часто комбинируются с нейросетями (для обучения) и эволюционными алгоритмами (для оптимизации), усиливая их способность к обработке сложных систем.

Во многих случаях решение таких задач является вычислительно сложным, из-за чего технологии HPC оказываются востребованными в методах мягких вычислений. Наиболее ярко эта потребность выражается в задачах: планирование «умных» городов с использованием эволюционных алгоритмов для многокритериальной оптимизации [Toutouh2019; Gora2015], использование систем на основе нечеткой логике для управления и мониторинга производственных процессов в режиме реального времени [Zhang2023; Vivas2022], анализ сложных структур с большим количеством связей в задачах фармацев-

тики и генетики с применением генетических и роевых алгоритмов [Liu2016; Easwaran2024]. Из этих примеров видно, что широкий спектр методов мягких вычислений хорошо поддаются распараллеливанию их алгоритмов.

Некоторые высокопроизводительные реализации нейро-нечетких систем были оформлены в самостоятельные программные модули. Частой практикой достижения высокой производительности нечетких систем является эффективная утилизация аппаратных ресурсов [FuzzyLite], в том числе, встраиваемых систем и плат ПЛИС [Aldair2018]. В [Lopez2015] и [Elkano2017] представлена реализация схемы *MapReduce* для классификации несбалансированных больших данных с использованием подхода Chi [Chi1996]. В ситуации, когда большой набор данных целиком или фрагментарно может быть проанализирован с использованием памяти только одного вычислительного узла, но требуется провести много итераций анализа, для быстрого получения промежуточного результата нечеткого анализа целесообразно использовать (графический) ускоритель [<empty citation>].

1.1 Нейро-нечеткие системы

Текущее развитие адаптивных нейро-нечетких систем (ANFIS) направлено на повышение применимости таких моделей за счет повышения точности моделирования и увеличения скорости получения результата моделирования. Прогресс по этим двум направлениям подкреплен экспериментами по использованию различных способов фазификации [Symmetry2021; Qian2023; Pekaslan2020], импликации [Shi2013; FernandezPeralta2025; Zhang2022; fern2021], дефазификации [VanLeekwijck1999; Nasiboglu2022], выборе t -норм, в том числе, t -норм в композиционном правиле вывода [Pourabdollah2015].

Например, в [Wagner2016] предлагается использовать меру Жаккара между нечеткими множествами на входе и н. м. антецедента для определения уровня срабатывания правила в нейро-нечетких системах при несинглтонной фазификации (NSFLS).

Существует также комбинированный подход с использованием так называемых гибких нейро-нечетких систем [rutkovskiy2010; Cingolani2012],

сочетающих в себе два метода нечеткого вывода. Использование параметрических t -норм дает возможность осуществлять плавный переход от одного метода вывода к другому. Это позволяет совершать подбор оптимального метода вывода для конкретной задачи посредством оптимизации данного дополнительного гиперпараметра.

Чаще всего нейро-нечеткие системы используются для анализа четких входных данных взятых из четких наборов данных. Однако существуют примеры использования нечетких систем для моделирования нечетких данных. Например, прогнозирования временных рядов [Pekaslan2020; Pourabdollah2017].

Исследуются также и подходы для обучения нейро-нечетких систем. В работах широко применяется метод [Wang1992] из-за своей простоты. Также появляются методы с использованием прогрессивных методов кластеризации правил [Svetlakov2021], эволюционного подхода [<empty citation>], градиентного спуска [<empty citation>] и непрерывного обучения на потоковых данных [Lima2010; Alves2021].

1.2 Методы нечеткого вывода

Эффективная организация нечеткого вывода является одной из главных точек повышения производительности при использовании нечетких систем. **Проблема разрабатывается.** Предложенная в 1965 году Заде теория нечетких множеств [zadeh1965] позже была использована для построения схемы нечеткого логического вывода [Zadeh1996] и нечетких логических систем [rutkovskiy2010]. Использование этой схемы нечеткого вывода для составных посылок оказывается затруднено из-за экспоненциальной временной сложности операций над нечеткими отношениями.

В ответ на проблему сложности использования классического нечеткого вывода спустя время появились методы Мамдани, Такаги-Сугено, Ларсена и Цукамото, вносящие в формулы нечеткого вывода упрощения для облегчения реализации. Упрощения прежде всего выражаются в использовании t -нормы вместо функции импликации, удовлетворяющей свойствам нечеткой импликации [rutkovskiy2010]. Недостатком такого упрощения является искажение

законов классической логики. Кроме того возникает расхождение при использовании лингвистической модели для анализа данных при многих правилах. Из-за этих недостатков данные подходы подвергаются критике.

В статьях [Dubois2012; Izquierdo2018] авторы отмечают несколько наблюдений о различиях вывода типа Мамдани и логического типа при четких входах:

- При полном соответствии входного нечеткого множества A' одному из антецедентов $A' = A_i$ в подходе Мамдани нельзя получить на выходе $B' = B_i$ при срабатывании еще одного правила $A_j \neq A_i, j \neq i$, в отличии от конъюнкции правил $B' = B_i \wedge B_j$ при логическом подходе.
- В подходе Мамдани при срабатывании в базе правил R^M нескольких противоречивых правил выходное нечеткое множество всегда $A \circ R^M \neq \emptyset$, а выходное дефазифицированное значение является усреднением центров противоречивых консеквентов B_1 и B_2 . При этом точка y может оказаться за пределами носителей нечетких множеств B_1 и B_2 , из-за чего для $\mu_{A'}(x_0) = 1$ в полученной выходной точке $\mu_{B'}(y) = 0$. В случае логического вывода такие правила взаимоисключаются, а на выходе будет получено пустое нечеткое множество \emptyset при достаточном удалении друг от друга B_1 и B_2 .

Изучение вопроса возможности использования несинглтонной фазификации возродил и развил американский ученый Джерри Мендель [Mendel2017]. Мендель привел аналитическое сравнение, показывающее, что NSFLS превосходят синглтонные нейро-нечетки системы в задаче прогнозирования зашумленных хаотических временных рядов благодаря своей способности учитывать неопределенность входных данных непосредственно при выводе. Он также сформировал доказательство того, что NSFLS является универсальным аппроксиматором любой непрерывной функции в некоторой области.

Однако исследования Менделя ограничены проработкой несинглтонной фазификации в системах типа Мамдани и Такаги-Сугено, которые наследуют описанные выше расхождения с каноническим логическим выводом. Также формулы нечеткого вывода в его работах легко переписываются в удобный для вычисления вид, что достигается в упрощении в виде использования одной и той же T -нормы в выражении вывода, что сужает возможную вариативность нечеткой модели.

Таким образом на момент проведения исследования не существовало подходов обеспечивающих эффективный канонический нечеткий логический вывод Заде с возможностью полноценно обрабатывать фазифицированные методом non-singleton входные значения. Актуальность решения описанной проблемы подтверждается:

1. увеличением количества публикаций с использованием non-singleton фазификации для вывода типа Мамдани за последние 10 лет, в которых достигается значимый прирост в качестве нечеткого моделирования;
2. низкой проработанностью логического типа нечеткого вывода при non-singleton фазификации, тогда как в определенных задачах синглтонный логический вывод превосходил по качеству моделирования метод Мамдани.

1.3 Несинглтонная фазификация

Большинство имеющихся реализаций нечеткого вывода предоставляют возможность анализа лишь четких — числовых данных.

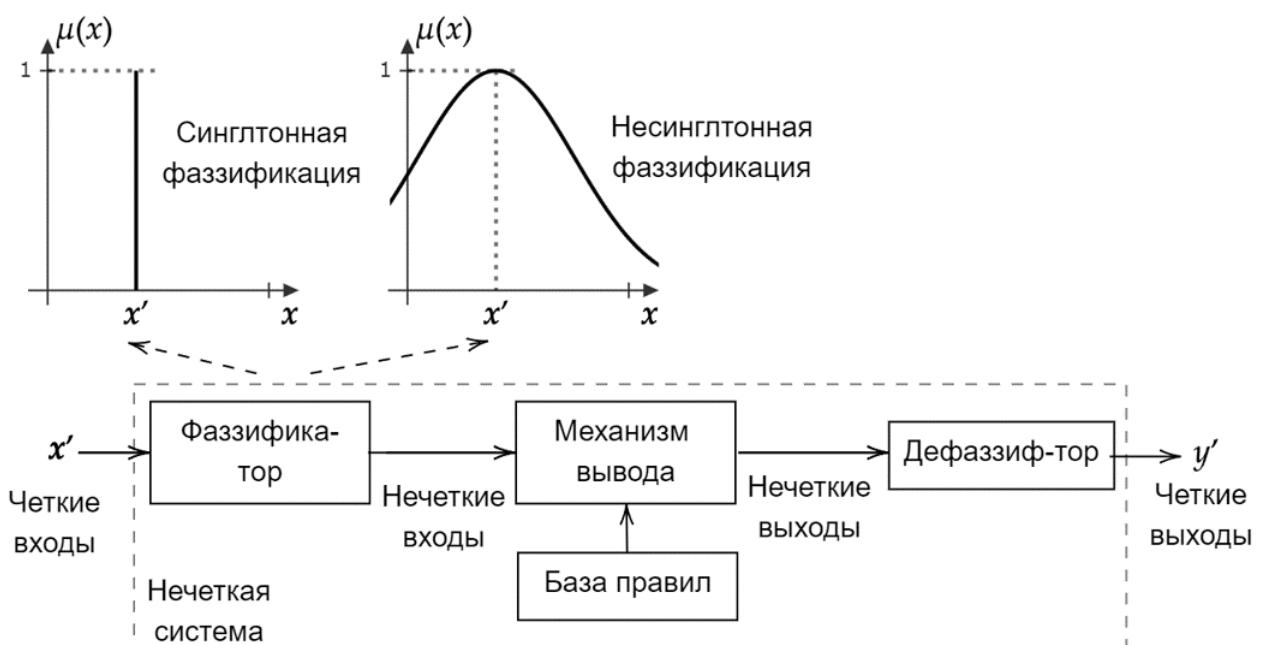
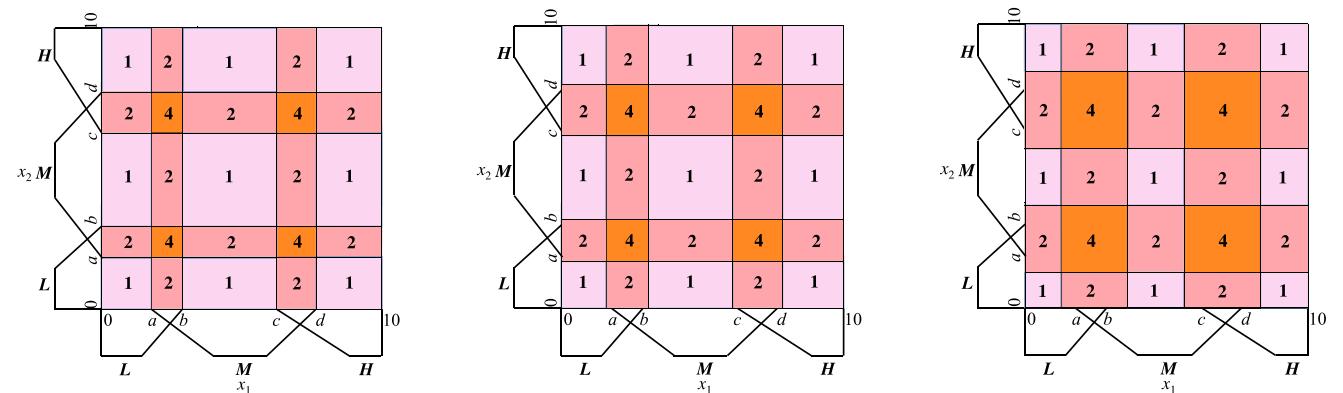


Рисунок 1.1 — Схема системы нечеткого вывода с использованием синглтонной и несинглтонной фазификации

Нечеткая система представляется в виде композиции фаззификатора, базы правил, модуля вывода и дефаззификатора, как показано на рисунке ???. Фаззификация — отображение входных данных из исходного пространства в пространство нечетких множеств. Наиболее распространенным является метод синглтонной фаззификации. Основная причина его широкого использования является существенное упрощение реализации систем нечеткого вывода. При использовании синглтонной фаззификации поданное на вход значение x' интерпретируется как истинное значение измеренной величины, что эквивалентно использованию функции принадлежности входного нечеткого множества $\mu_{A'}(x) = [x = x']$. Информация о неопределенности входных данных при этом игнорируется.

Альтернативный подход с использованием несинглтонной фаззификации предусматривает формализацию входного значения нечетким множеством, содержащим информацию о неопределенности значения точки входных данных. Эти неопределенности могут возникать как результат несовершенства процедуры измерений (например, шумом измерительного оборудования, дефектами или деградацией качества датчиков), или когда входные данные описываются качественными понятиями естественного языка.

В случае с анализом числовых данных, неопределенность измерений формализуется функцией принадлежности, которая $\mu_{A'}(x') = 1$ и $\mu_{A'}(x)$ уменьшается по мере удаления от x' . При таком способе моделирования измеренное значение x' рассматривается как истинное, а значения в его окрестности — как возможные.



$$a) \sigma_{A'} = 0\%$$

$$b) \sigma_{A'} = 4\%$$

$$c) \sigma_{A'} = 12\%$$

Рисунок 1.2 — Сравнение областей активации правил при переходе от синглтонной фаззификации к несинглтонной и при увеличении ширины окрестности погрешности $\sigma_{A'}$.

Влияние перехода от синглтонной фазификации к несинглтонной и величины окрестности погрешности на внутреннее поведение и итоговое качество нечеткой системы продемонстрировал Мендель в [Mendel2021; Mendel2017].

Для анализа влияния от использования того или иного способа фазификации на корректность получаемого результата в статье [Mendel2020] рассматривается карта разбиений первого и второго порядка на декартовом произведении базовых множеств входных лингвистических переменных. Мендель в своей книге проводил такое сравнение для систем типа Мамдани. Поскольку в системах типа Мамдани в качестве функции импликации выступает t -норма, то разница от применения двух способов фазификации проявляется в различных максимальных уровнях линии пересечения ф. п. входной посылки и антецедента правила.

Применение композиционного правила вывода sup здесь дает эффект *префильтрации* (или *корректировки*) входного значения. То есть ядра функций принадлежности антецедентов правил выполняют функцию эталонов, а уровень пересечения функции принадлежности для входного значения и ф. п. антецедента правила позволяет интерпретировать входное значение как суперпозицию эталонных значений антецедентов с долями равными этому уровню.

Показанная на этих схемах динамика более ясно раскрыта на рисунке 1.3 для различных значениях среднеквадратичного отклонения в гауссовой ф. п. входного значения на примере агрегации выходной ф. п. нечеткой системы с тремя правилами в базе правил. Видно, что при переходе от синглтонной фазификации к несинглтонной и при дальнейшем увеличении ширины среднеквадратичного отклонения ф. п. входного нечеткого множества, повышается уровень срабатывания первого правила, и, как следствие использования импликации Мамдани, уровень задействования ф. п. выходного нечеткого множества этого правила в результирующей агрегации. Кроме того, можно проанаблюдать, упомянутый ранее, эффект корректировки входного значения антецедентом третьего правила.

В случае с нечеткой системой логического типа, разница от использования различных способов фазификации будет проявляться в различных формах выходного нечеткого множества.

Можно проследить за влиянием увеличения ширины окна для измеренного значения на область выходного нечеткого множества нечеткой системы при использовании логического подхода к нечеткому выводу. При логическом подхо-

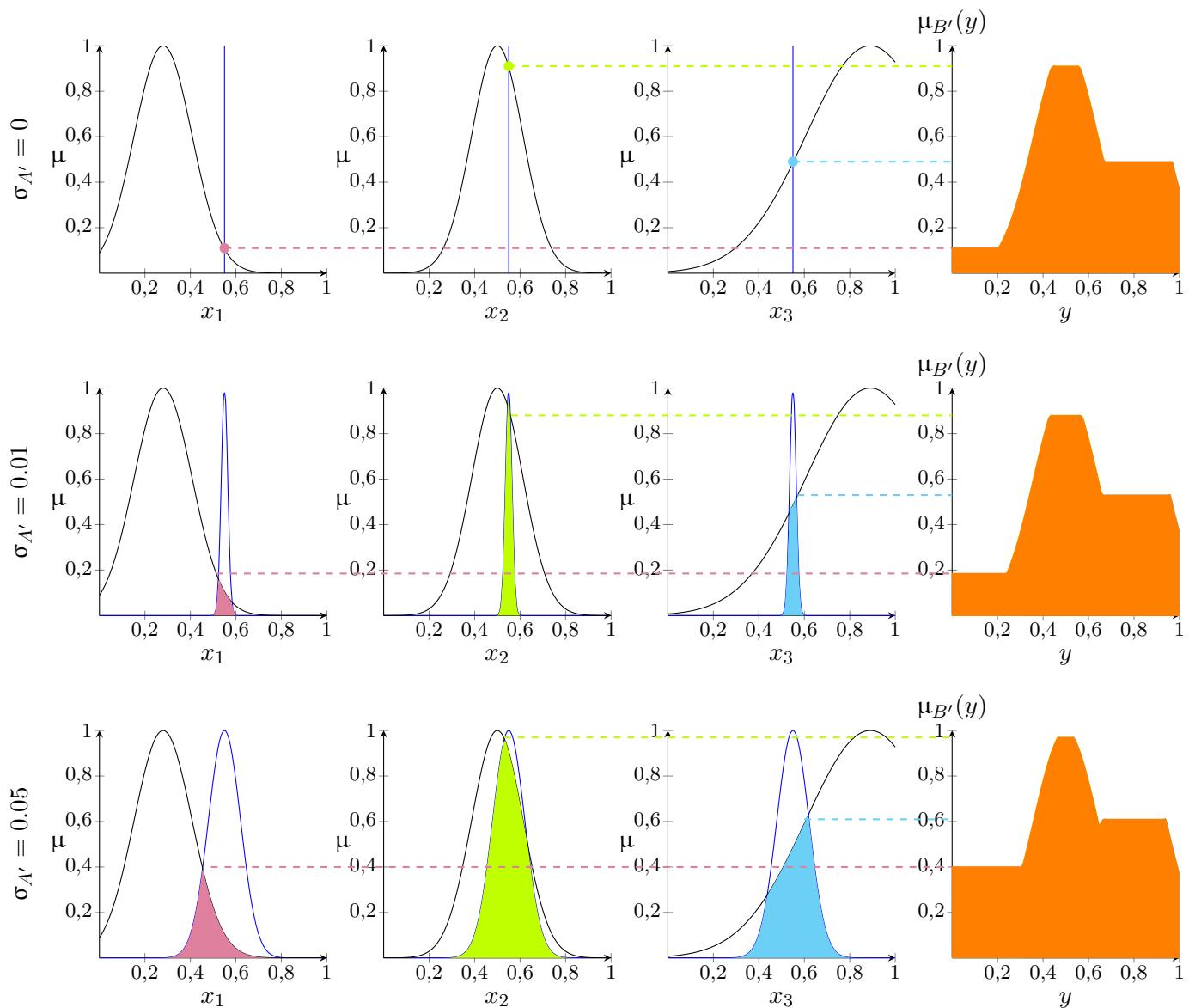


Рисунок 1.3 – Сравнение формы функций принадлежности выходных нечетких множеств для подхода Мамдани

де функция принадлежности выходного нечеткого множества формируется как результат пересечения (в данном случае операцией *min*), что можно представить как постепенное вырезание области функции принадлежности выходного нечеткого множества. Из рисунка 1.4 видно, что при увеличении ширины в пересечении проекций импликации на пространство выходной переменной оказывается более «сложно выкроенная» область.

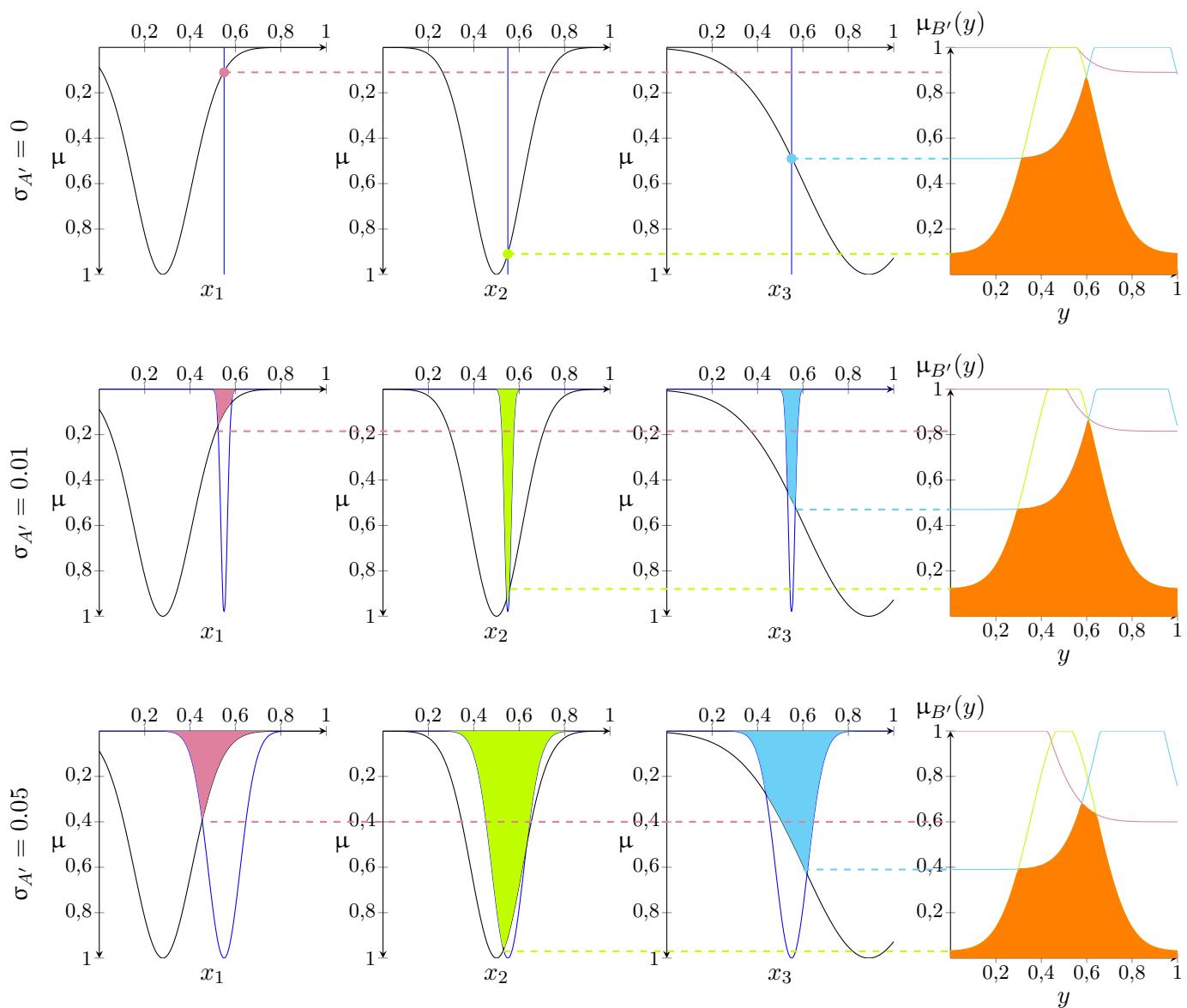


Рисунок 1.4 – Сравнение формы функций принадлежности выходных нечетких множеств для логического подхода

1.4 Нечеткое значение истиности

В рамках нечеткой логики лингвистическая переменная «истинность» расширяет классическую бинарную модель истиности, позволяя выражать градуированные оценки. Возможные значения ее термов включают значения «истинно», «ложно», «очень истинно», «очень ложно», «слегка истинно», «слегка ложно», «абсолютно истинно», «абсолютно ложно» и др. Данная концепция позволяет строить связи, содержащие качественные зависимости, в цепочках знаний в экспертных системах и системах поддержки принятия решений.

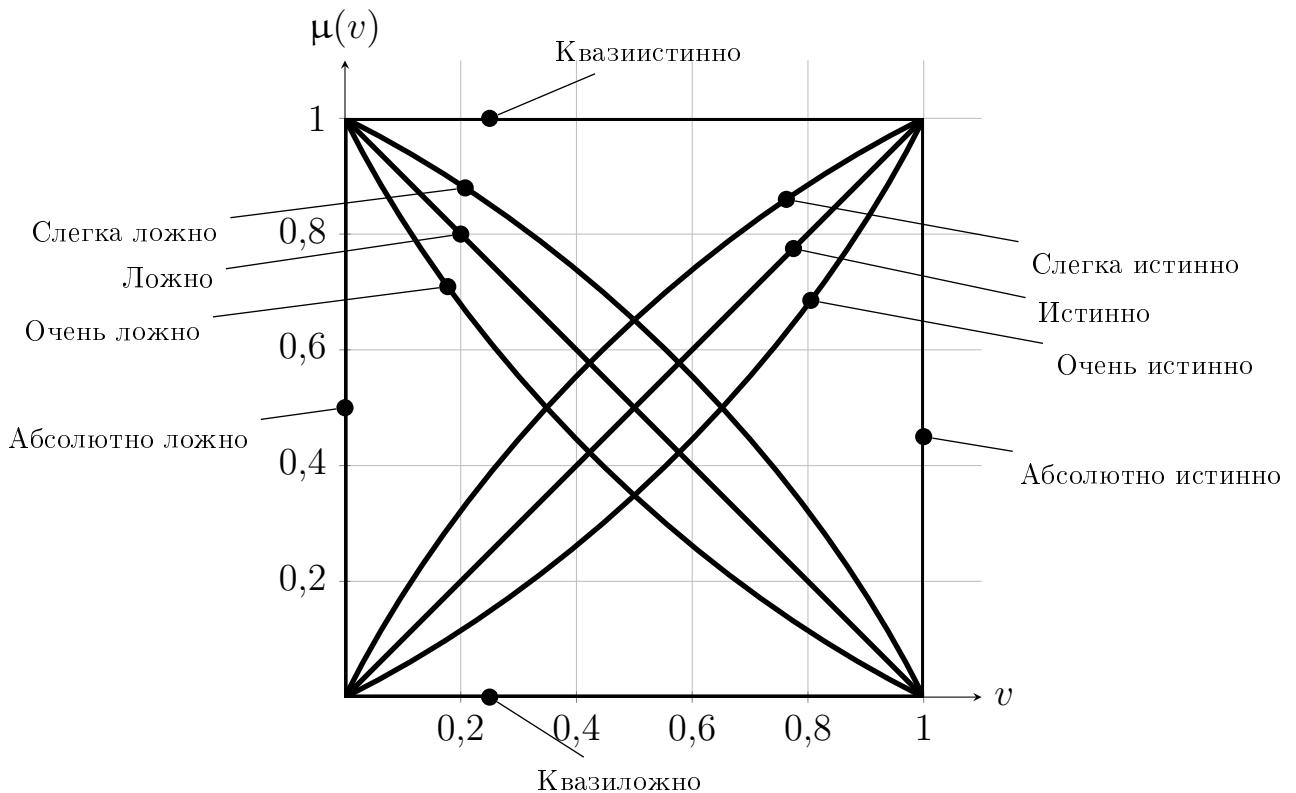


Рисунок 1.5 – Значения лингвистической переменной «истинность»

На рисунке 1.5 изображены функции принадлежности термов лингвистической переменной «истинность». Этим функциям принадлежности соответствуют следующие аналитические способы задания:

$$\begin{aligned}
 M[\text{«истинно»}] &= \int_0^1 v/v; & M[\text{«ложно»}] &= \int_0^1 1 - v/v; \\
 M[\text{«слегка истинно»}] &= \int_0^1 \sqrt{v}/v; & M[\text{«слегка ложно»}] &= \int_0^1 \sqrt{1-v}/v; \\
 M[\text{«очень истинно»}] &= \int_0^1 v^2/v; & M[\text{«очень ложно»}] &= \int_0^1 \frac{(1-v)^2}{v}; \\
 M[\text{«абсолютно истинно»}] &= \frac{1}{1} + \int_0^1 \frac{0}{v}; & M[\text{«абсолютно ложно»}] &= \frac{1}{0} + \int_0^1 \frac{0}{v}; \\
 M[\text{«квазиистинно»}] &= \int_0^1 1/v; & M[\text{«квазиложно»}] &= \int_0^1 0/v.
 \end{aligned}$$

В данной работе лингвистическая переменная «истинность» предлагается использовать для нечеткой оценки истинности одних нечетких высказываний относительно других. Значения этих высказываний формализуются нечеткими множествами A и A' , определенными на базовом множестве X . Здесь высказывание соответствующее нечеткому множеству A' рассматривается как

достоверная, относительно которой оценивается истинность высказывания заданного нечетким множеством A .

Определение. Нечеткой истинностью множества A относительно нечеткого множества A' называется нечеткое множество $CP(A, A')$ такое, что:

$$\mu_{CP(A, A')}(v) = \sup_{\substack{\mu_{A'}(x)=v \\ x \in X}} \{\mu_A(x)\}. \quad (1.1)$$

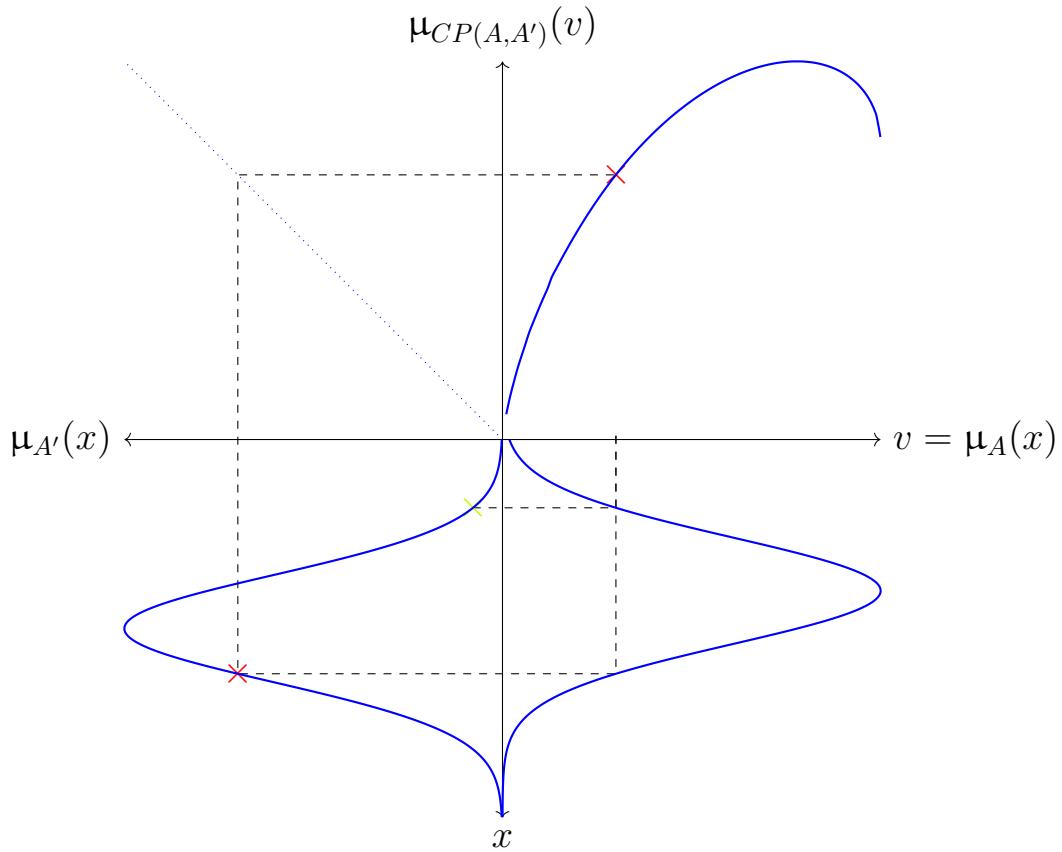


Рисунок 1.6 — Пример вычисления нечеткого значения истинности

Процедура вычисления истинности одного нечеткого множества относительно другого, согласно формуле (1.1), отражена на рисунке 1.6.

Понятие *относительной истинности* достоверного высказывания A' относительно оцениваемого высказывания A опирается на несколько аксиом:

- *Аксиома истинности.* Нечеткое значение истинности ИСТИННО задается нечетким множеством:

$$CP(A, A') = \{ \langle \mu_{CP(A, A')}(v), v \rangle \} = \{v/v\}, v \in [0; 1],$$

что выполняется тогда и только тогда, когда A относительно соответствует A' , т. е. функции принадлежности нечетких множеств A' и A совпадают.

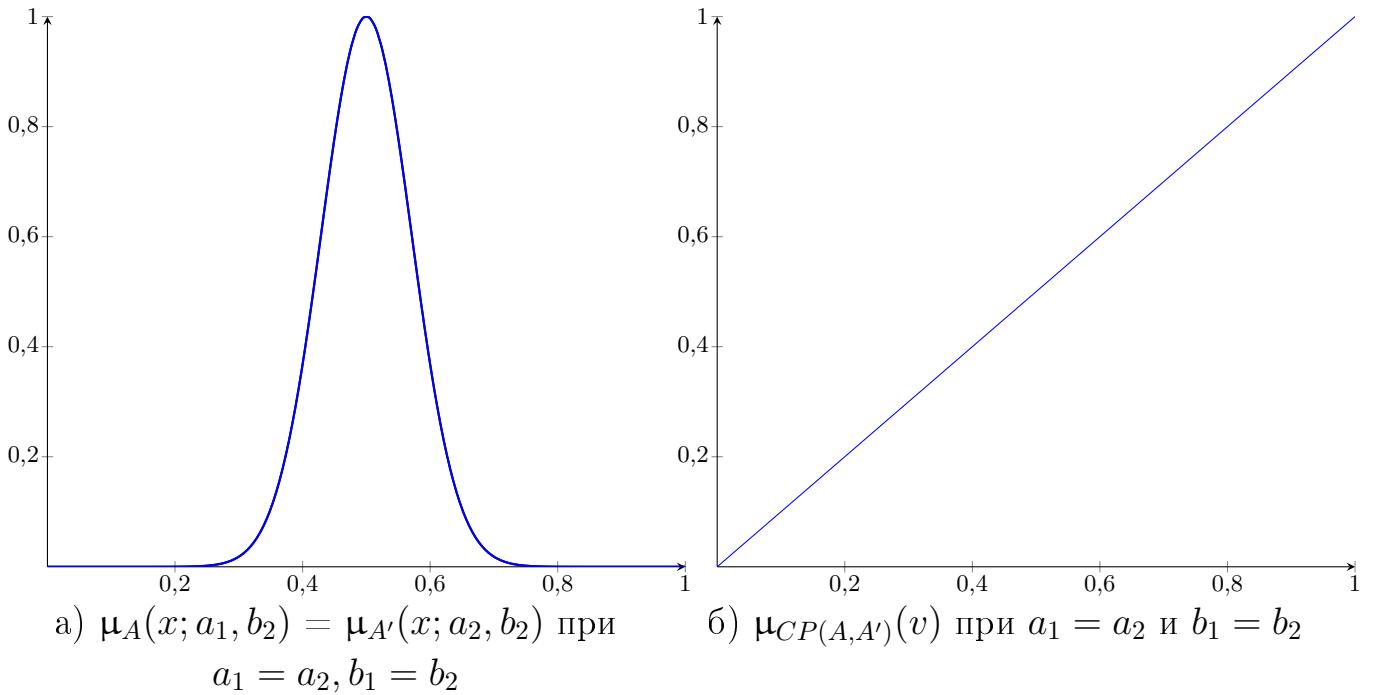


Рисунок 1.7 — Иллюстрация случая истинного отношения высказываний

На рис. 1.7 представлены графики совпадающих функций принадлежности высказываний и построенной функции принадлежности нечеткого значения истинности.

- *Аксиома ложности.* Нечеткое значение истинности ЛОЖНО задается нечетким множеством:

$$CP(A, A') = \{ \langle \mu_{CP(A, A')}(v), v \rangle \} = \{ (1-v)/v \} = \{ v/(1-v) \}, v \in [0; 1],$$

что выполняется тогда и только тогда, когда утверждаемое высказывание A противоположно утверждаемому в A' , т. е. функции принадлежности высказываний A' и A удовлетворяют одному из условий:

$$\mu_{A'}(x) = 1 - \mu_A(x)$$

или

$$\mu_{A'}(x) = \begin{cases} 1 - \mu_A(x), & x \leq x_{max} \\ 0, & x > x_{max} \end{cases}$$

или

$$\mu_{A'}(x) = \begin{cases} 0, & x < x_{max} \\ 1 - \mu_A(x), & x \geq x_{max}, \end{cases}$$

где $x_{max} = \arg \max_x \mu_A(x)$.

На рис. 1.8 представлены графики противоположных по значению функций принадлежности высказываний A' и A и построенной функции принадлежности нечеткого значения истинности.

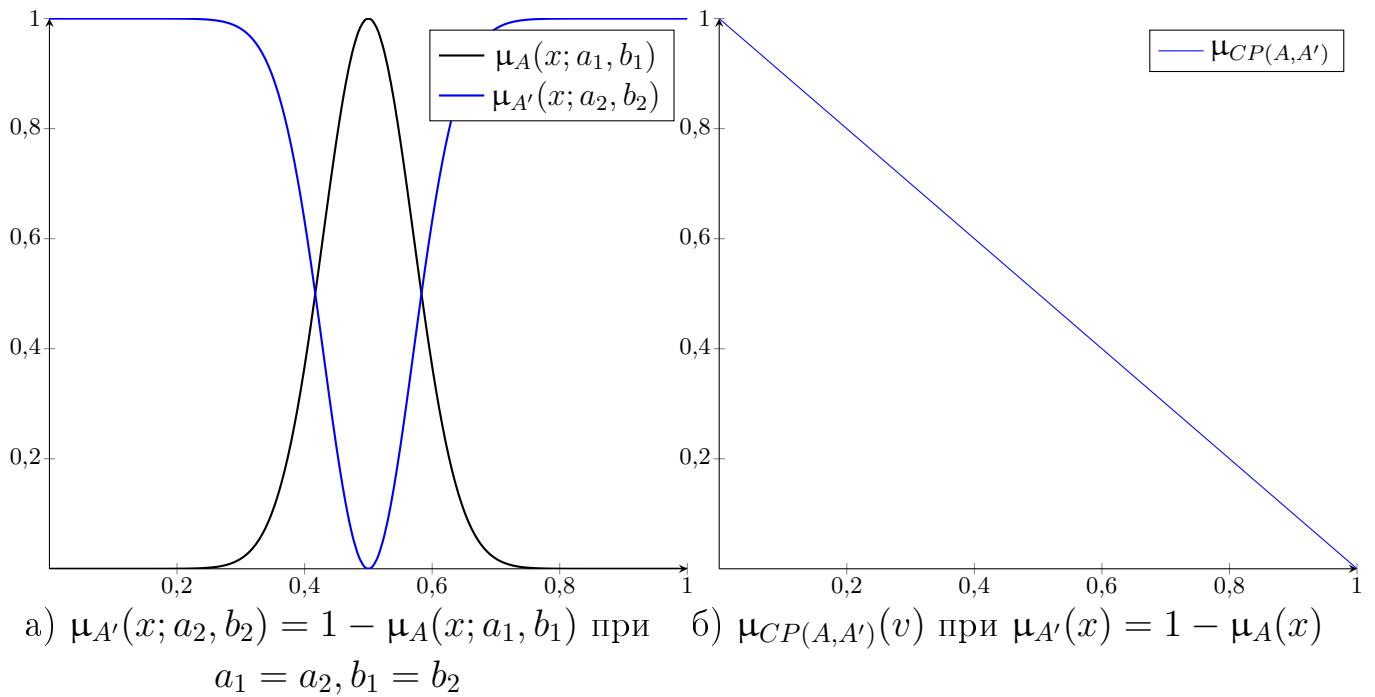


Рисунок 1.8 — Иллюстрация случая ложного отношения высказываний

- *Аксиома абсолютной истинности.* Значение истинности АБСОЛЮТНО ИСТИННО задается нечетким множеством:

$$CP(A, A') = \{ \langle \mu_{CP(A, A')}(v), v \rangle \} = \{v/1\} = \{1/1\}, v \in [0, 1],$$

что выполняется тогда и только тогда, когда A' абсолютно соответствует A , то есть в случае когда оценка данная в высказываниях A' и A является четкой или нечеткой, когда носитель высказывания A' включен в носитель высказывания A .

На рис. 1.9 представлены графики функций принадлежности высказывания A' , включенного в A и функции принадлежности нечеткого значения истинности, соответствующие данной ситуации. Для моделирования четкого значения функции принадлежности (синглтона) взята гауссова функция кривая с дисперсией, стремящейся к нулю.

- *Аксиома абсолютной ложности.* Значение истинности АБСОЛЮТНО ЛОЖНО задается нечетким множеством:

$$CP(A, A') = \{ \langle \mu_{CP(A, A')}(v), v \rangle \} = \{v/0\} = \{1/0\}, v \in [0, 1],$$

что выполняется тогда и только тогда, когда A' абсолютно не соответствует A , то есть в случае когда оценки данные в высказываниях A' и A имеют несовпадающие носители.

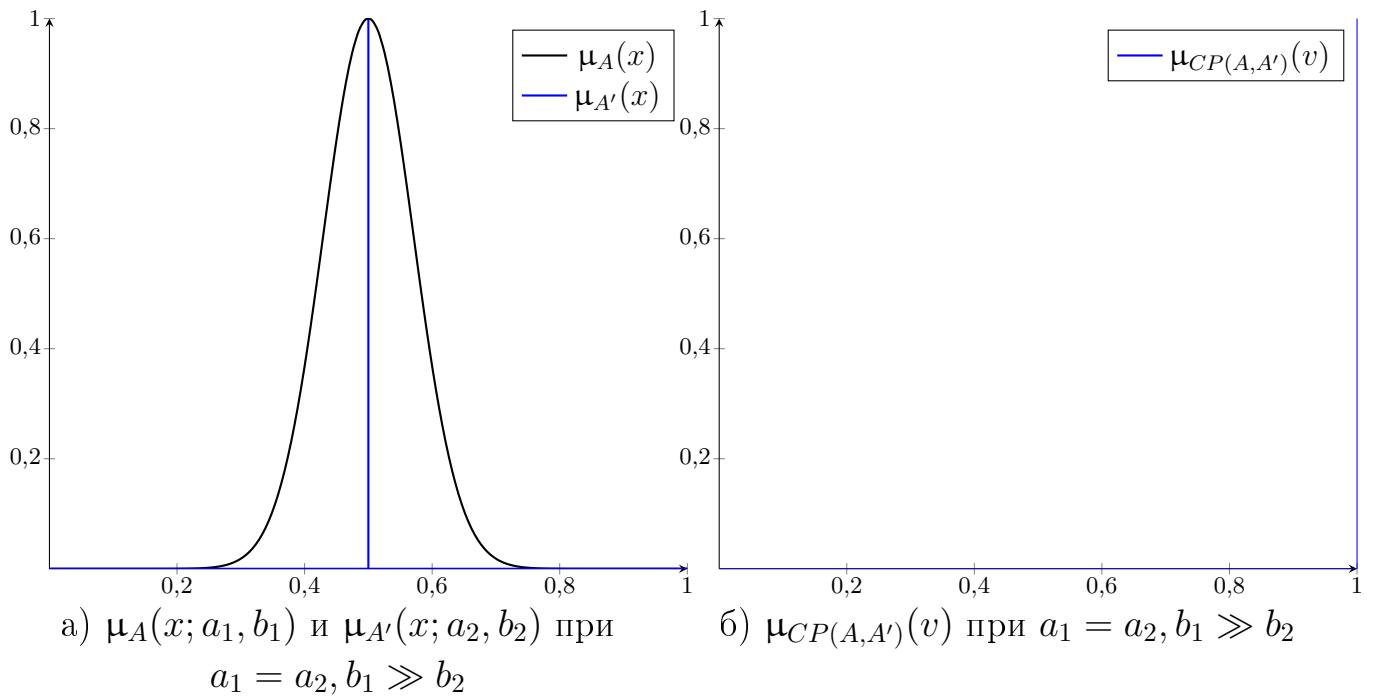


Рисунок 1.9 — Иллюстрация случая абсолютно истинного отношения высказываний

На рис. 1.10 представлены графики непересекающихся гауссовых функций принадлежности высказываний A' и A с удаленными центрами и построенная для этого случая функция принадлежности нечеткого значения истинности.

- *Аксиома квазистинности.* Нечеткое значение истинности КВАЗИСТИННО задается нечетким множеством:

$$CP(A, A') = \{ \langle \mu_{CP(A, A')}(v), v \rangle \} = \{1/v\}, v \in [0; 1],$$

что выполняется тогда и только тогда, когда утверждение в высказывании A' является частным по отношению к A , то есть оценка данная в высказывании A интервальная и совпадает с носителем высказывания A .

На рис. 1.11 представлены графики функций принадлежности нечеткого множества A , полностью содержащегося в нечетком множестве A' , и рассчитанной функции принадлежности нечеткого значения истинности.

- *Аксиома квазиложности.* Нечеткое значение истинности КВАЗИЛОЖНО задается нечетким множеством:

$$CP(A, A') = \{ \langle \mu_{CP(A, A')}(v), v \rangle \} = \{0/v\}, v \in [0; 1],$$

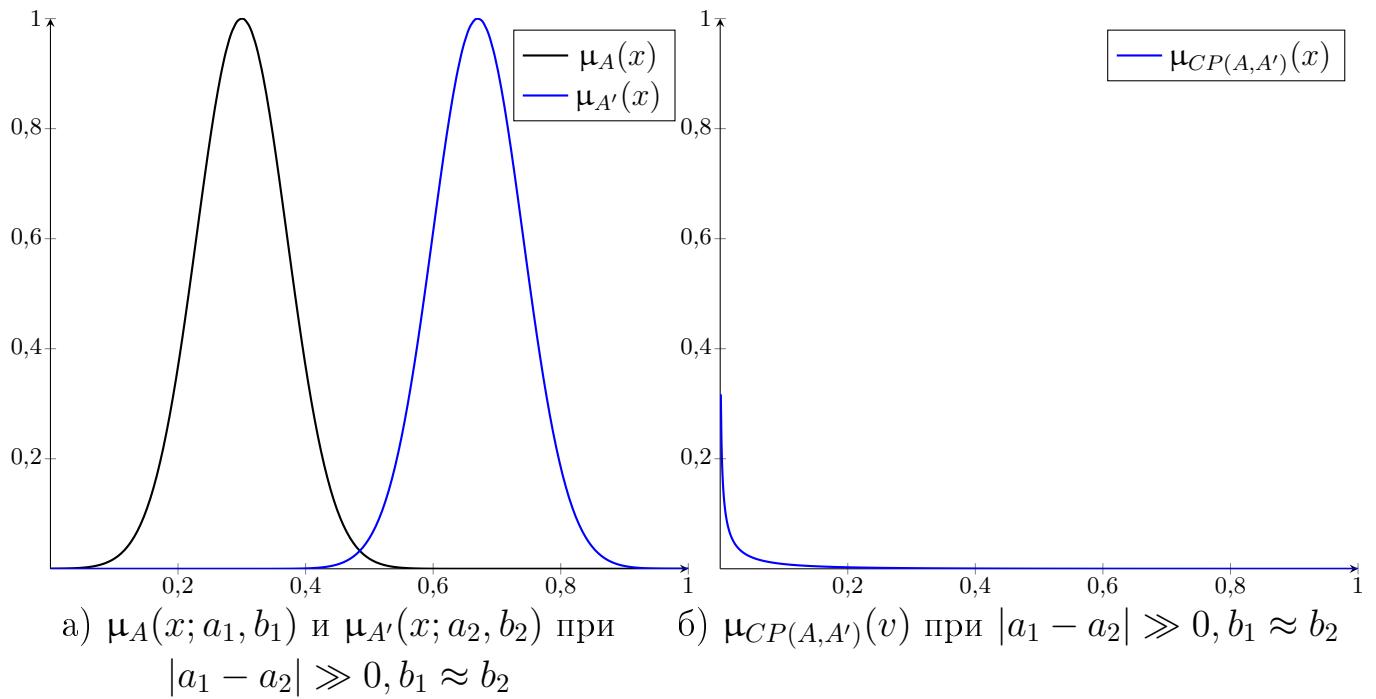


Рисунок 1.10 — Иллюстрация случая абсолютно ложного отношения высказываний

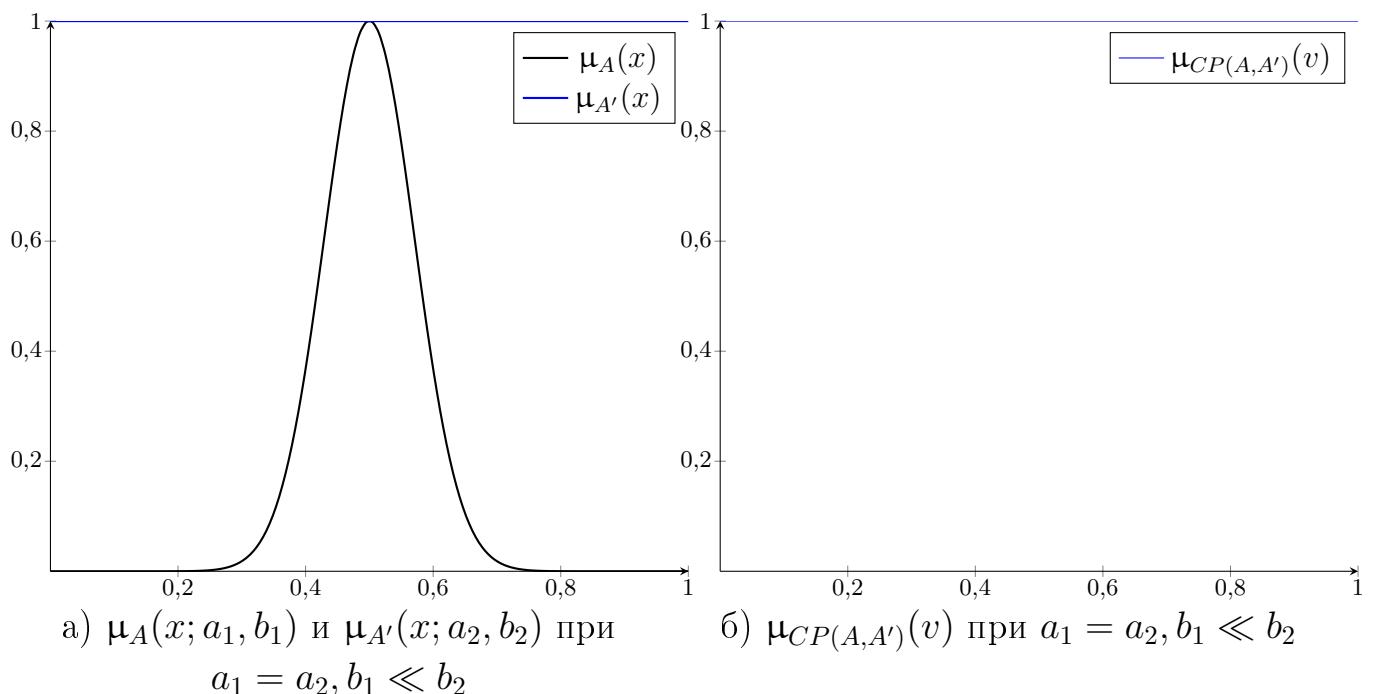


Рисунок 1.11 — Иллюстрация случая квазистинного отношения высказываний

что справедливо тогда и только тогда, когда утверждаемое в A' не имеет реального подтверждения в действительности. Иными словами, отсутствует возможность установления истинности высказывания A' , так как не определено, существует ли в действительности то, что утверждается в A' .

Данную ситуацию нельзя выразить математически и изобразить, поскольку истинность функция принадлежности высказывания A' не может быть оценена.

1.4.1 Вычисление нечеткого значения истинности, когда функции принадлежности высказываний задаются гауссовыми функциями

Пусть функции принадлежности нечетких множеств высказываний A и A' заданы разновидностью гауссовых функций:

$$\mu_A(x; a, b) = e^{-\frac{(x-a)^2}{2b^2}} \quad \mu_{A'}(x; c, d) = e^{-\frac{(x-c)^2}{2d^2}}.$$

Тогда, согласно формуле нечеткого значения истинности (1.1), для вычисления НЗИ в точке v_0 необходимо сперва найти все точки из области определения функции принадлежности факта, в которых он принимает значение v_0 . В случае с гауссовой функцией это можно сделать, с помощью обратной гауссовой функции:

$$x(v) = a \pm b\sqrt{-2 \ln v},$$

тогда

$$\begin{aligned} \mu_{CP(A,A')}(v) &= \max \left\{ e^{-\frac{((a-b\sqrt{-2 \ln v})-c)^2}{2d^2}}, e^{-\frac{((a+b\sqrt{-2 \ln v})-c)^2}{2d^2}} \right\} \\ &= \max \left\{ e^{-\frac{((a-c)-b\sqrt{-2 \ln v})^2}{2d^2}}, e^{-\frac{((a-c)+b\sqrt{-2 \ln v})^2}{2d^2}} \right\} \end{aligned} \quad (1.2)$$

1.5 Применение нечетких моделей для прогнозирования временных рядов

Для временного ряда $\mathbf{y}_t = (y_1, \dots, y_T)$ величина y_t представляет измеренное значение наблюдаемой переменной в момент времени t . Ставится задача предсказания значений \hat{y}_{t+h} для заданного горизонта прогнозирования h .

Модель временного ряда $f(\cdot)$ порядка p использует последние p значений до момента t для оценки значения:

$$\hat{y}_{t+h} = f(y_{t-p}, \dots, y_t),$$

где p - размер лагового окна.

При моделировании временных последовательностей с использованием нейро-нечетких систем каждое значение $y_t \in Y$ фазифицируется в нечеткое множество A_t . Эти нечеткие множества составляют множество термов лингвистической переменной y_t со значениями, определенными на базовом множестве Y . Такая система принимает p входов, а правила в ее базе знаний устанавливают нечеткую последовательно-временную связь в рамках заданного окна $p + 1$. Параметр p называется порядком нечеткой системы прогнозирования временных рядов.

База правил в такой системе представляется набором из N правил вида:

$$R_k : \begin{aligned} &\text{Если } y_{t-p} \text{ есть } A_{k1} \text{ и } \dots \text{ и } y_t \text{ есть } A_{kp}, \\ &\text{то } y_{t+1} \text{ есть } A_{kp+1}, \end{aligned} \quad k \in \overline{1, N}, \quad (1.3)$$

где каждое нечеткое отношение $T_1 \{A_{k1}, \dots, A_{kp}\} \rightarrow A_{kp+1}$, заданное в правиле R_k , выражает единицу **логических** знаний о моделируемом протекающем во времени процессе.

Входы нечеткой системы по каждому измерению могут описываться одинаковыми лингвистическими переменными \tilde{y} , заданными на базовом множестве области значений временного ряда значениями из одного и того же множества термов $\{A_t\}$ и имеющей одинаковую порождающую процедуру для терм-множеств.

1.5.1 Формирование знаний в нечетких системах в контексте задачи прогнозирования временных рядов

Развитие способов обучения нечетких систем в рамках задачи прогнозирования временных рядов проходило взаимосвязано с адаптацией нечеткого моделирования как непосредственно к задаче регрессии временных рядов, так и к задачам моделирования другого рода. Ранние подходы следовали более

типовому набору шагов при построении нечетких систем [Chellai2022]. При формирования набора термов лингвистической переменной \tilde{y} ее пространство значений ограничивалось минимальным и максимальными значениями временного ряда и разбивалось на накладывающиеся друг на друга участки, каждый из которых составлял носитель соответствующего нечеткого множества из набора термов. В широко цитируемом подходе Chen-a [Chen1996] разбиение производилось на равные отрезки. Очевидным недостатком такого примитивного способа разбиения пространства значений является несоответствие часто отличному от равномерного распределению функции плотности вероятности этих значений. База правил формировалась напрямую из экземпляров данных, посредством сопоставления каждому экземпляру комбинации термов, имеющих наибольшую степень принадлежности по входам в совокупности, что соответствует довольно популярному из-за своей простоты методу Wang-a [Wang1992]. Такой способ построения базы правил нечеткой системы до сих пор популярен при исследовании нечетких моделей временных рядов и других данных, когда необходимо сравнить качество непосредственно моделирования при том же наборе данных и способе построения базы правил.

В более поздних способах стало более распространенным формирование базы правил с соответствующими терм-множествами из данных, то есть подбор параметров функций принадлежности термов выполняется одновременно с формированием набора правил. Этот подход является более простым и точным для выделение шаблонных отрезков из временных рядов, особенно в случаях сложной функции плотности распределения значений временного ряда, когда проявляются ограничения в увеличении точности нечеткой системы при разбиении базового множества лингвистической переменной \tilde{y} . Таким образом одни и те же методы можно применять для разбиения пространства значений временных рядов (одно измерение) или для разделения пространства окон временных рядов (когда последовательность значений в окне временного ряда интерпретируется как точка в n -мерном пространстве). В последнем случае посредством разбиения пространства окон временного ряда может осуществляться формирование базы правил.

Распространенной группой таких методов являются подходы на основе различных алгоритмов кластеризации [Lucas2021]: k -средних, алгоритмы учитывающие плотность распределения точек, агломеративная кластеризация.

Эволюционные подходы, например, Particle Swarm Optimization (PSO)[**Davari2009**] могут обеспечить оптимальную настройку параметров нечеткой модели при сложной конфигурации обучающего набора данных. Основная сложность использования этих методов состоит в необходимости выполнять большое количество вычислений функции приспособленности на всем обучающем наборе данных на каждой итерации для всех особей популяции. Поэтому для сокращения времени оценки параметров модели стараются минимизировать набор параметров, для которых применяются методы глобальной оптимизации.

Стоит также отметить гибридные подходы построения и обучения нечетких систем моделирования временных рядов в комбинации с другими моделями машинного обучения, такими как, Support Vector Machine (SVM), Long-Sort Term Memory (LSTM), Transformer, и статистические модели, например, Auto Regressive Integrated Moving Average (ARIMA).

Отдельно в случае авторегрессионного прогнозирования временных рядов подходы потокового обучения нечетких систем (*evolving fuzzy systems, EFS*). Метод **participatory learning** — подход к адаптивному обучению, при котором нечеткая система динамически обновляет свою структуру и параметры, основываясь на степени согласованности новых данных с текущей моделью [**Lima2010; Alves2021**].

Описанные выше методы используют для нечеткого вывода методы Мамдани и Такаги-Сугено. Последний особенно популярен в задаче прогнозирования временных рядов.

1.5.2 Анализ способов построения нечетких временных моделей

Для раскрытия потенциала нечеткого моделирования нужно отразить информацию о прогнозируемой величине во входных нечетких множествах нечеткой системы. Это достигается путем Существующие подходы фазификации значений временных рядов комбинируют различные формы функций принадлежности, способы оценку истинного значения в термах и формализации неопределенности. В [**Pekaslan2020**] описан подход к фазификации измеренных значений временного ряда в нечеткие множества с гауссовыми функциями

принадлежности. Измеренные значения полагаются истинными и задают центры гауссовых функций принадлежности, а среднеквадратичное отклонение оценивается как среднеквадратичная разность между соседними значениями в некотором окне вокруг данной точки во временной последовательности. [Написать формулу](#) В [Pourabdollah2017] для фазифицированных таким образом значений в нечеткие множества выполняется переоценка истинных значений посредством преобразования взвешенным скользящим средним центров гауссовой ф. п. нечетких множеств. [Написать формулу](#) Стоит уточнить, что в этих двух работах рассматривается нечеткий вывод с использованием non-singleton фазификации.

Для выбора лучшей конфигурации нечеткой модели в задаче регрессии ключевыми метриками как правило являются *Root Mean Square Error (RMSE)* и *Mean Absolute Error (MAE)*. Выбрать наиболее подходящие методов импликации и дефазификации также можно изучив эмпирический опыт использования различных вариантов при решении задачи регрессии. В нескольких исследованиях, в которых непосредственно сравнивались операторы импликации в регрессионных задачах, Лукасевич неизменно оказывался лучшим с наименьшим отклонением между прогнозируемыми и фактическими значениями, заnim следовали Райхенбах и Гедель (или Клини–Динес) в большинстве экспериментов [Gkountakou2023]. Напротив, *R*-импликациям, таким как Goguen и Gödel, часто не хватает плавности [Krieken2022]. В [Kiszka1985] операторы, основанные на *S*-нормах (например, Лукасевич, Райхенбах), неизменно давали более низкий RMSE, чем простое минимальное значение (Мамдани), что часто сокращало погрешность на 10-20% при моделировании параметров двигателя постоянного тока.

Общим практикой является включение метода дефазификации в набор оптимизируемых гиперпараметров. Для задачи регрессии часто используются методы дефазификации центра тяжести и среднего максимума. Например, в [Ali2022] метод центра тяжести показал лучшее значение метрики RMSE, а в [Yan2010] — метод среднего максимума.

Также в [Bede2018] была показана лучшая точность прогнозирования временного ряда при использовании импликации Лукасевича и дефазификации по методу среднего максимума, превзойдя по точности дефазификацию по методу центра тяжести при импликации Лукасевича и минимуме (Мамдани).

1.6 Постановка задачи исследования

С учетом описанного в разделе 1.2 состояния в области исследования нечеткого логического вывода, а именно: обоснованной примерами прироста качества нечеткого моделирования перехода к использованию несинглтонной фазификации в моделях типа Мамдани, продемонстрированного несоответствия таких подходов как Мамдани принципам классического нечеткого логического вывода, вызванного определенным упрощением, а также низкой проработанностью этих проблем в совокупности в научных публикациях, актуальным является развитие нечеткого логического вывода при несинглтонной фазификации. Одна из основных задач в этом направлении исследования состоит в преодолении существовавшего до сих пор барьера в виде высокой вычислительной сложности при увеличении количества входов системы.

Изучение возможных решений данной проблемы стоит начать с разбиения нечеткого логического вывода на составляющие математические операции. Описанная выше концепция нечеткого значения истинности предоставляет способ нахождения нечеткой меры сходства входного нечеткого множества и нечеткого множества в антецеденте правила, что по сути соответствует определенной части в композиции операций механизма нечеткого вывода. Кроме того, в литературе определена операция свертки НЗИ для отдельных независимых входов системы в единое пространство, что выносит проблемы обработки нескольких входов системы вывода «за скобки» непосредственно нечеткого логического вывода.

Затем нужно выявить особенности и возможные сложности высокопроизводительной реализации выработанного метода при анализе неопределенных данных. Следует организовать вычисления с применением параллельных технологий программирования.

Поскольку нечеткие системы логического типа демонстрировали хорошую точность в задачах регрессии имеет смысл оценить качество моделирования с использованием разработанной нечеткой модели при решении задачи прогнозирования временных рядов. Кроме того следует сделать замеры и провести сравнительный анализ времени работы для выполненной параллельной реализации.

1.7 Выводы

1. В ответ на сложность использования канонического нечеткого вывода Заде появился ряд упрощенных методов вывода Мамдани, Такаги-Сугено, Ларсена и Цукамото, тогда как логический подход показывал лучшее качество моделирования в некоторых задачах. Мендель показал как можно использовать фазификацию типа non-singleton на основе подходов Мамдани и Такаги-Сугено, и продемонстрировал значимый прирост в качестве нечеткого моделирования в некоторых задачах. Вывод основанный на подходах Мамдани, Такаги-Сугено и др. не , в то время как нечеткий вывод логического типа при non-singleton фазификации остается не исследован.
2. Нечеткое значение истинности выражает истинность одного нечеткого множества относительно другого в нечетком пространстве истинности. Целесообразно будет выработать механизм эффективного вычисления этой истинности, а затем, рассматривая процедуру нечеткого логического вывода как композицию некоторых операций, внедрить операцию вычисления нечеткого значения истинности в эту композицию. Использование в качестве функции принадлежности нечетких множеств гауссовой функции позволяет легко аналитически выражать операции по вычислению нечеткого значения истинности без потери общности.
3. В качестве практической задачи для применения разрабатываемой нечеткой модели вывода выбрана задача прогнозирования временных рядов с зашумленными и неопределенными входными данными. Такая задача характерна для анализа реальных измерений в технических и природных системах, где присутствуют шумы, погрешности и неполнота информации. Использование нечетких моделей позволяет учитывать неопределенность входных данных непосредственно в процессе вывода, что способствует повышению устойчивости и точности прогнозирования по сравнению с традиционными методами. Для оценки эффективности предложенного подхода планируется провести экспериментальное исследование на реальных и синтетических временных рядах, сравнив качество и производительность с существующими решениями.

4. Исходя из обоснования актуальности разработки проблемы нечеткого вывода логического типа и характеристики механизма нечеткого значения истинности , необходимо... Выработанный метод нужно адаптировать для высокопроизводительной реализации. Затем полученную реализацию нечеткого логического вывода нужно применить для решения задачи прогнозирования временных рядов, и оценить качество и производительность реализованной нечеткой логической модели.

Глава 2. Метод нечеткого вывода на основе нечеткого значения истиности

2.1 Постановка задачи нечеткого вывода

Лингвистическая модель представляет собой базу правил вида:

R_k : Если x_1 есть A_{k1} и x_2 есть A_{k2} и … и x_n есть A_{kn} , то y есть B_k , (2.1)

где N – количество нечетких правил, $A_{ki} \subseteq X_i, i = \overline{1, n}, B_k \subseteq Y$ – нечеткие множества, которые характеризуются функциями принадлежности $\mu_{A_{ki}}(x_i)$ и $\mu_{B_k}(y)$ соответственно; x_1, x_2, \dots, x_n – входные переменные лингвистической модели, причем

$$[x_1, x_2, \dots, x_n]^T = \mathbf{x} \in X_1 \times X_2 \times \cdots \times X_n.$$

Символами $X_i, i = \overline{1, n}$ и Y обозначаются соответственно пространства входных и выходной переменных. Если ввести обозначения $\mathbf{X} = X_1 \times X_2 \times \cdots \times X_n$ и $\mathbf{A}_k = A_{k1} \times A_{k2} \times \cdots \times A_{kn}$, причем

$$\mu_{\mathbf{A}_k}(\mathbf{x}) = \bigwedge_{i=1}^n \mu_{A_{ki}}(x_i),$$

где T_1 – произвольная t -норма, то правило ?? представляется в виде нечеткой импликации

$$R_k : \mathbf{A}_k \rightarrow B_k, k = \overline{1, N}. \quad (2.2)$$

Правило R_k можно формализовать как нечеткое отношение, определенное на множестве $\mathbf{X} \times Y$, т.е. $R_k \subseteq \mathbf{X} \times Y$ – нечеткое множество с функцией принадлежности

$$\mu_{R_k}(\mathbf{x}, y) = \mu_{\mathbf{A}_k \rightarrow B_k}(\mathbf{x}, y).$$

Модель логического типа определяет задание функции $\mu_{\mathbf{A}_k \rightarrow B_k}(\mathbf{x}, y)$ на основе известных функций принадлежности $\mu_{\mathbf{A}_k}(\mathbf{x})$ и $\mu_{B_k}(y)$ с помощью одной из предложенных в [2] функций импликации:

$$\mu_{\mathbf{A}_k \rightarrow B_k}(\mathbf{x}, y) = I(\mu_{\mathbf{A}_k}(\mathbf{x}), \mu_{B_k}(y)),$$

где I – некоторая импликация.

Ставится задача определить нечеткий вывод $B'_k \subseteq Y$ для системы, представленной в виде (??), если на входах - нечеткие множества. $\mathbf{A}' = A'_1 \times A'_2 \times \cdots \times A'_n \subseteq \mathbf{X}$ или x_1 есть A'_1 и x_2 есть A'_2 и … и x_n есть A'_n с соответствующей функцией принадлежности $\mu_{\mathbf{A}'}(\mathbf{x})$, которая определяется как

$$\mu_{\mathbf{A}'}(\mathbf{x}) = \underset{i=1,n}{T_3} \mu_{A'_i}(x_i). \quad (2.3)$$

Несинглтонный фаззификатор отображает измеренное $x_i = x'_i, i = \overline{1,n}$ в нечеткое число, для которого $\mu_{A'_i}(x'_i) = 1$ и $\mu_{A'_i}(x_i)$ уменьшается от единицы по мере удаления от x'_i . В соответствии с обобщенным нечетким правилом modus ponens [2], нечеткое множество B'_k определяется композицией нечеткого множества \mathbf{A}' и отношения \mathbf{R}_k , т.е.

$$B'_k = \mathbf{A}' \circ (\mathbf{R}_k \rightarrow B_k),$$

или, на уровне функций принадлежности

$$\mu_{B'_k}(y|\mathbf{x}') = \sup_{\mathbf{x} \in \mathbf{X}} \left\{ \mu_{\mathbf{A}'}(\mathbf{x}') \overset{T_2}{\star} I(\mu_{\mathbf{A}_k}(\mathbf{x}), \mu_{B_k}(y)) \right\}. \quad (2.4)$$

В (2.4) применена условная нотация, так как ввод в нечеткую систему происходит при определенном значении \mathbf{x} , а именно \mathbf{x}' . Обозначение $\mu_{B'_k}(y|\mathbf{x}')$ показывает, что $\mu_{B'_k}$ изменяется с каждым значением \mathbf{x}' . Вычислительная сложность выражения (2.4) составляет $O(|X_1| \cdot |X_2| \cdot \dots \cdot |X_n| \cdot |Y|)$ т.е. экспоненциальная.

2.2 Вывод на основе нечеткого значения истинности

Используя правило истинностной модификации [1] можно выразить:

$$\mu_{A'}(\mathbf{x}) = \tau_{A|A'}(\mu_A(x))$$

где $\tau_{A|A'}$ — нечеткое значение истинности (НЗИ) нечеткого множества A относительно A' , представляющее собой функцию принадлежности совместности $CP(A_k, A')$ A_k по отношению к A' , причем A' рассматривается как достоверное [Дюбуа и др., 1990]:

$$\tau_{A_k|A'}(v) = \mu_{CP(A_k, A')}(v) = \sup_{\substack{\mu_{A_k}(x)=v \\ x \in X}} \{\mu_{A'}(x)\}. \quad (2.5)$$

Таким образом НЗИ отражает совместимость факта с посылкой в нечеткой форме. Упрощенные подходы отображают совместимость в одно значение из диапазона впервые представленное в [5].

Перейдем от переменной x к переменной v в выражении нечеткого вывода (??), обозначив

$$\mu_{A_k}(x) = v \text{ и } \mu_{A'}(x) = \tau_{A_k|A'}(v),$$

то есть выполним преобразование нечетких множеств на пространстве X в истинностное пространство $[0, 1]$:

?????

Получим:

$$\mu_{A'}(x) = \tau_{A_k|A'}(\mu_{A_k}(x)) = \tau_{A_k|A'}(v) \quad (2.6)$$

Тогда (2.4) примет вид:

$$\mu_{B'_k}(y|x') = \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'}(v) \stackrel{T_2}{\star} I(v, \mu_{B_k}(y)) \right\}. \quad (2.7)$$

При переходе к нечеткому выводу по n входам формула вычисления НЗИ для нечетких отношений посылки и факта имеет вид:

$$\tau_{\mathbf{A}_k|\mathbf{A}'}(v) = \sup_{\substack{\mu_{\mathbf{A}_k}(x_1, \dots, x_n) = v \\ (x_1, \dots, x_n) \in \mathbf{x}}} \{ \mu_{A'}(x_1, \dots, x_n) \}.$$

Или в выражении через операции сверток t -норм T_1 (2.2) и T_3 (2.3):

$$\tau_{\mathbf{A}_k|\mathbf{A}'}(v) = \sup_{\substack{T_1 \\ i=1,n}}_{\mu_{A_{ki}}(x_i) = v} \left\{ \prod_{i=1,n}^1 \mu_{A'_i}(x_i) \right\}. \quad (2.8)$$

Вместо выражения (2.8), НЗИ для n входов может быть вычислено как свертка НЗИ по каждомуциальному входу:

$$\tau_{\mathbf{A}_k|\mathbf{A}'}(v) = \tilde{\tau}_{i=1,n} \tau_{A_{ki}|A'_i}, v \in [0, 1], \quad (2.9)$$

где $\tilde{\tau}$ - расширенная по принципу обобщения n -местная t -норма [**kutsenko2015methods**], которая определяется как

$$\tilde{\tau}_{i=1,n} \tau_{A_{ki}|A'_i}(v) = \sup_{\substack{T_1 \\ i=1,n \\ v_i = v} \\ (v_1, \dots, v_n) \in [0,1]^n} \left\{ \prod_{i=1,n}^1 \tau_{A_{ki}|A'_i}(v_i) \right\} \quad (2.10)$$

в результате перехода

$$\mu_{A_{ki}}(x_i) = v_i \text{ и } \mu_{A'_i}(x_i) = \tau_{A_{ki}|A'_i}(v_i).$$

Тогда для системы с n входами выражения нечеткого вывода на основе НЗИ (2.7) примет вид:

$$\mu_{B'_k}(y|\mathbf{x}') = \sup_{v \in [0,1]} \left\{ \tau_{\mathbf{A}_k|\mathbf{A}'}(v) \stackrel{\mathrm{T}_2}{\star} I(v, \mu_{B_k}(y)) \right\} \quad (2.11)$$

Стоит отметить, что выражение (2.9) можно записать следующим образом, подчеркнув возможность попарного рекурсивного нахождения свертки НЗИ:

$$\begin{aligned} \tau_{A_k, A'}(v) &= \tilde{\mathrm{T}}_{\substack{i=1, n}} \tau_{A_{ki}|A'_i}(v_i) \\ &= \left(\dots \left(\left(\mu_{CP(A_{k1}, A'_1)}(v_1) \tilde{\mathrm{T}}_1 \mu_{CP(A_{k2}, A'_2)}(v_2) \right) \tilde{\mathrm{T}}_1 \dots \right) \tilde{\mathrm{T}}_1 \mu_{CP(A_{kn}, A'_n)} \right). \end{aligned}$$

Для $n = 2$, $\tilde{\mathrm{T}}$ записывается как:

$$\tilde{\mathrm{T}}_{\substack{i=1, 2}} \tau_{A_{ki}|A'_i}(v) = \sup_{\substack{v_1 \mathrm{T}_1 v_2 = v \\ v_1, v_2 \in [0,1]}} \left\{ \tau_{A_{k1}|A'_1}(v_1) \mathrm{T}_3 \tau_{A_{k2}|A'_2}(v_2) \right\}, v \in [0,1]. \quad (2.12)$$

При вербализации импликации в (2.10) она представится в виде:

$$\text{Если нзи есть ИСТИННО, то } y \text{ есть } B'_k \quad (2.13)$$

Таким образом, (2.13) представляет собой еще одну структуру правил в отличие от канонических структур Заде [10] и Такаги-Сугено [9]. Применение данного правила не зависит от количества входов в нечетких системах.

В формуле (2.11) данный подход позволяет переместить процесс вывода в единое пространство НЗИ, где функции истинности, в отличии от различных пространств в подходе Заде, могут быть объединены в более эффективный вычислительный процесс.

Порядок функции временной сложности вычисления B'_k на основе выражения (2.11) составляет $O(n|V|^2 + |V| \cdot |Y|)$, где $V = CP(A_k, A')$. Сравнение схем нечетких выводов с соответствием соотношениями (2.4) и (2.11) представлены на рис. 2.1.

Классический подход (метод Заде):

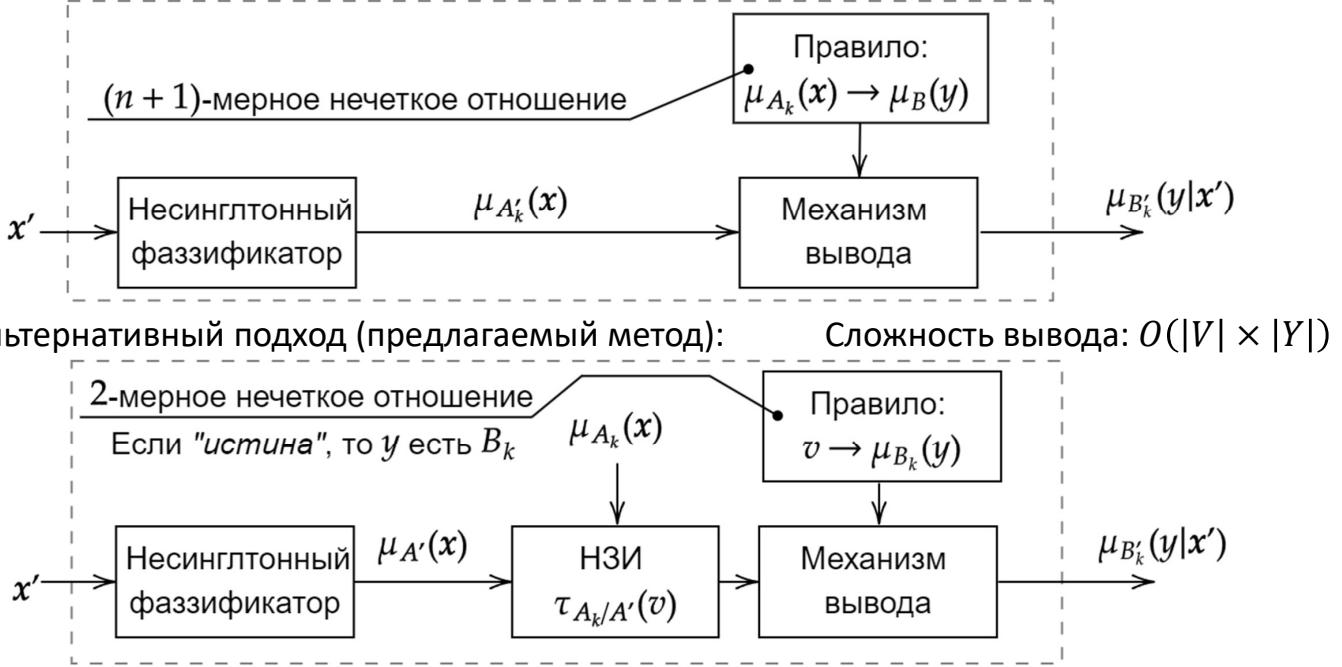


Рисунок 2.1 — Сравнение классической схемы нечеткого вывода и схемы нечеткого вывода на основе НЗИ

2.3 Вывод для систем логического типа

... Выделяют следующие специальные категории импликаций:

- S-импликация (Клине-Динеса, Лукасевича, Райхенбаха, Фодора):
 $I(a, b) = S(1 - a, b)$
- R-импликация (Гогуен, Гедель): $I(a, b) = \sup_z \{z | T(a, z) \leq b\}$
- Q-импликация (Заде, Вильмотта): $I(a, b) = S(1 - a, T(a, b))$

В логическом подходе правила объединяются связкой «И», тогда результирующее нечеткое множество является результатом произведения нечетких множеств, получаемых в результате нечеткого логического вывода по каждому правилу отдельно:

$$B' = \bigcap_{r=1}^N B'_r. \quad (2.14)$$

$$\mu_{B'}(y) = \bigcap_{r=1}^N \mu_{B'_r}(y) = \bigcap_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \star^{\text{T}_2} I(v, \mu_{B_r}(y)) \right\} \right\} \quad (2.15)$$

2.4 Нечеткий вывод с использованием различных методов дефазификации

В статье [VanLeekwijck1999] описывается подход к сравнению методов дефазификации.

Описанный метод реализации [eisele1994].

2.4.1 Дефазификация по методу среднего центра

$$\hat{y}_{CA} = \frac{\sum_{k=1}^N \bar{y}_k \mu_{B'_k}(\bar{y}_k)}{\sum_{k=1}^N \mu_{B'_k}(\bar{y}_k)}$$

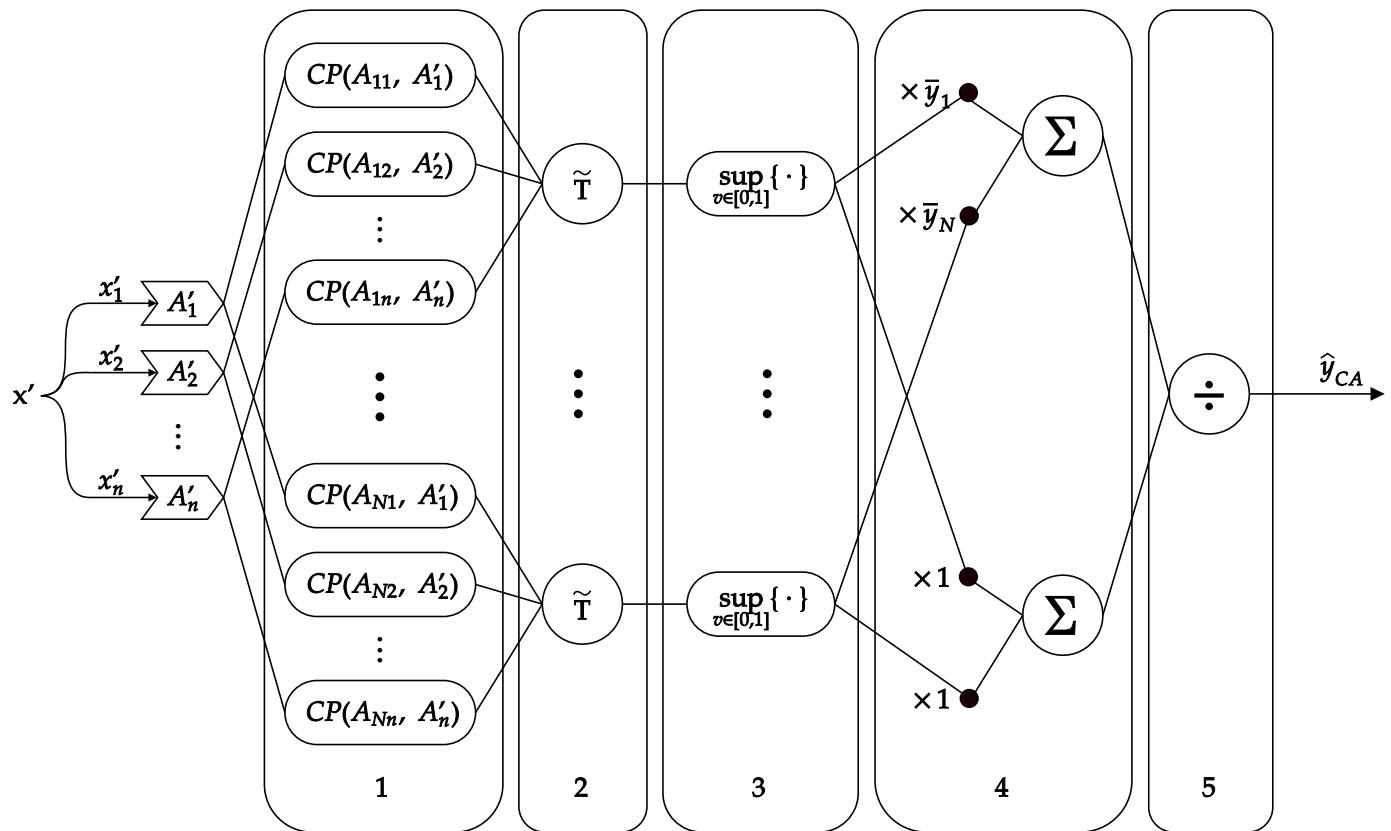


Рисунок 2.2 — Схема нейро-нечеткой системы с использованием дефазификации по методу среднего центра для S - и R -импликаций

Поскольку $\mu_{B_r}(\bar{y}_k) = 1$ при $k = r$, тогда по формуле (2.11) $\mu_{B'_k}(\bar{y}_k)$ выразится:

$$\mu_{B'_k}(\bar{y}_k) = \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'} \stackrel{T_2}{\star} I(v, \mu_{B_k}(\bar{y}_k)) \right\} \quad (2.16)$$

$$= \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'} \stackrel{T_2}{\star} I(v, 1) \right\}. \quad (2.17)$$

Обозначив $I(v, 1) = I_1(v)$, с учетом () формула дефазификации (2.16) примет вид:

$$\hat{y}_{CA} = \frac{\sum_{k=1}^N \bar{y}_k \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} I_1(v) \right\}}{\sum_{k=1}^N \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} I_1(v) \right\}} \quad (2.18)$$

Рассмотрим вычисление τ_k для различных категорий импликаций:

– для S -импликаций:

$$I_1(v) = I(v, 1) = S\{1 - a, 1\} = 1$$

– для R -импликаций:

$$I_1(v) = I(v, 1) = \sup_z \{z | T(v, z) \leq 1\} \quad v \in [0, 1]$$

$$= \sup_z \{z | \forall z\} \quad z \in [0, 1]$$

$$= 1$$

– для Q -импликаций:

$$\begin{aligned} I_1(v) &= I(v, 1) = S\{1 - v, T(v, 1)\} \\ &= S\{1 - v, v\} \end{aligned}$$

Тогда для S - и R -импликаций формула (2.18) с учетом свойства t -нормы $T(a, 1) = a$ примет вид:

$$\hat{y}_{CA} = \frac{\sum_{k=1}^N \bar{y}_k \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} 1 \right\}}{\sum_{k=1}^N \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} 1 \right\}} \quad (2.19)$$

$$= \frac{\sum_{k=1}^N \bar{y}_k \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \right\}}{\sum_{k=1}^N \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \right\}}, \quad (2.20)$$

а для Q -импликации:

$$\hat{y}_{CA} = \frac{\sum_{k=1}^N \bar{y}_k \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} S \{1-v, v\} \right\}}{\sum_{k=1}^N \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} S \{1-v, v\} \right\}} \quad (2.21)$$

Схема нейро-нечеткого системы соответствующая формуле дефазификации (??) изображена на рисунке 2.2.

2.4.2 Дефазификация по методу центра тяжести

Если выходное значение блока выработки решения представляет собой единственное агрегированное нечеткое множество B' , следует рассмотреть к использованию данный и последующие методы дефазификации. Этот метод можно сопоставить со схемой вычисления математического ожидания случайной величины при данном ее распределении.

$$\hat{y}_{CoG} = \frac{\int_Y y \mu_{B'}(y) dy}{\int_Y \mu_{B'}(y) dy}$$

Тогда (??) при использовании данной импликации запишется в виде:

$$\hat{y}_{CoG} = \frac{\int_Y y \prod_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} I(v, \mu_{B_r}(y)) \right\} \right\} dy}{\int_Y \prod_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} I(v, \mu_{B_r}(y)) \right\} \right\} dy} \quad (2.22)$$

Значение \hat{y}_{CoG} в данном способе дефазификации может быть вычислено с применением численных методов. Однако существует упрощенная схема нахождения выходного значения в данном методе с помощью дискретной формулы центра тяжести в точках центров функций принадлежности термов выходной лингвистической переменной или ф. п. консеквентов правил в базе правил [rutkovskiy2010]. Эта схема выражается формулой ниже.

$$\hat{y}_{CoG} = \frac{\sum_{k=1}^N \bar{y}_k \mu_{B'}(\bar{y}_k)}{\sum_{k=1}^N \mu_{B'}(\bar{y}_k)}, \quad (2.23)$$

где \bar{y}_k — центр ф. п. нечеткого множества B_k , то есть такое значение y , в котором $\max_y \mu_{B_k}(y) = 1$.

В этом случае формула (??) на основе (??) примет вид:

$$\hat{y}_{CoG} = \frac{\sum_{k=1}^N \bar{y}_k T_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} I(v, \mu_{B_r}(\bar{y}_k)) \right\} \right\}}{\sum_{k=1}^N T_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} I(v, \mu_{B_r}(\bar{y}_k)) \right\} \right\}}, \quad (2.24)$$

Выражение внутри фигурных скобок оператора $\sup_{v \in [0,1]} \{\cdot\}$, представляющее композицию двух нечетких множеств на пространстве истинности, есть определение *возможности*, то есть соответствие того, что $\tau_{A_r|A'}$ есть τ_{kr} и наоборот. Обозначим величину возможности:

$$\sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \stackrel{T_2}{\star} \tau_{kr} \right\} = \Pi_{kr} \quad (2.25)$$

Поскольку импликация в выражении (??eqn:defuz-cog-4)) не зависит от входных данных, то значения $I(v, \mu_{B_r}(\bar{y}_k)) = \tau_{kr}, k = \overline{1, N}, r = \overline{1, N}$ могут быть вычислены предварительно, то есть до использования композиционного правила вывода.

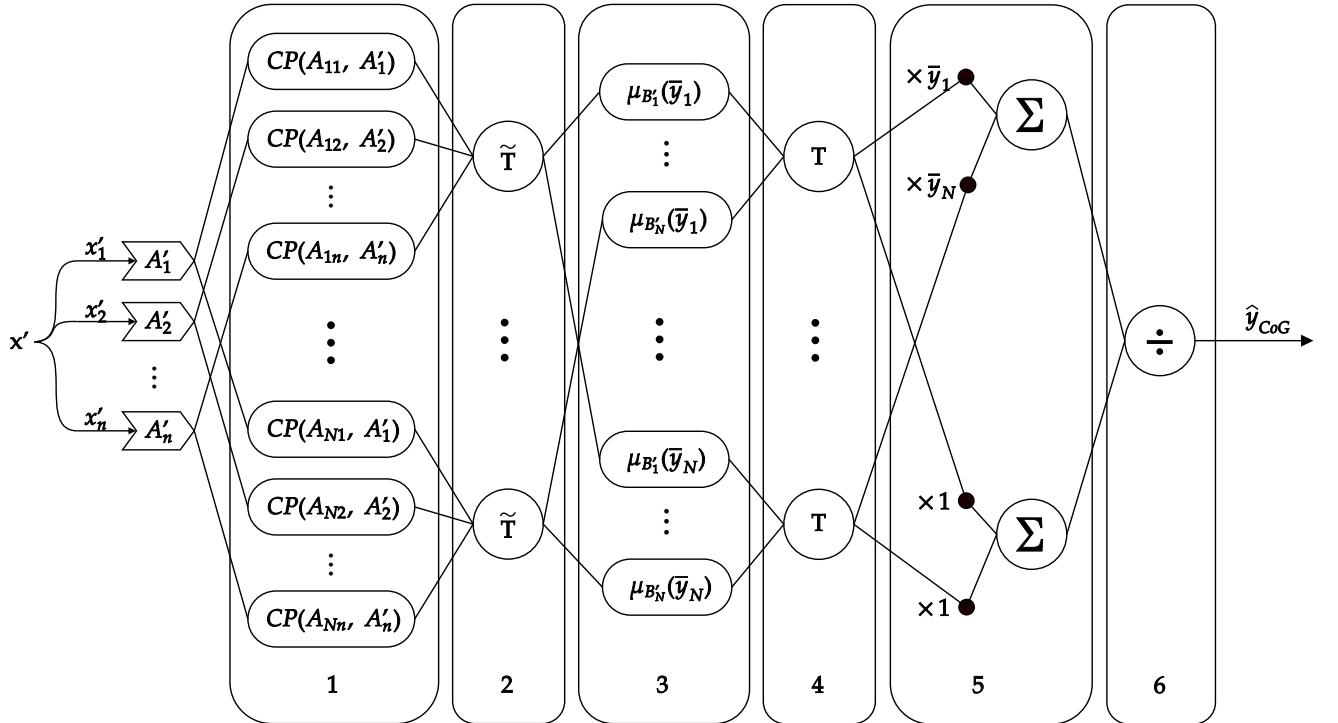


Рисунок 2.3 — Схема нейро-нечеткой системы с использованием дефазификации по методу центра тяжести

Одна из возможностей упрощения процедуры вывода возникает, когда функции принадлежностей термов выходной лингвистической переменной

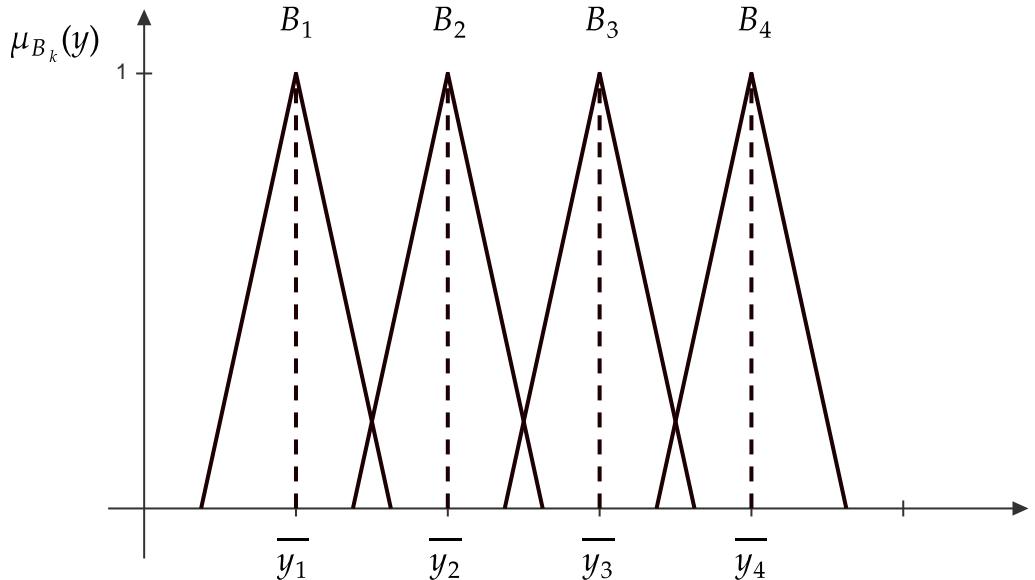


Рисунок 2.4 — Пример нечетких множеств, удовлетворяющих условию
 $\mu_{B_k}(y_r) = 0$ для $y \neq r$

достаточно удалены друг от друга и имеют низкую степень взаимного пересечения, то есть выполняется соотношение $\mu_{B_k}(y_r) \approx 0$ при $k \neq r$, что проиллюстрировано на рисунке 2.4.

Рассмотрим вычисление τ_{kr} для различных категорий импликаций:

— для S -импликации

$$\tau_{kr}(v) = \begin{cases} 1 - v, & \text{если } k \neq r \\ 1, & \text{если } k = r \end{cases}$$

$$\hat{y}_{CoG} = \frac{\sum_{k=1}^N \bar{y}_k T_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \star_{T_2} (1-v) \right\} \right\}}{\sum_{k=1}^N T_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \star_{T_2} (1-v) \right\} \right\}},$$

— для R -импликации

$$\tau_{kr}(v) = \begin{cases} \delta(v), & \text{если } k \neq r \\ 1, & \text{если } k = r \end{cases}, \quad \text{где } \delta(v) = \begin{cases} 1, & v = 0 \\ 0, & v > 0 \end{cases}$$

$$\hat{y}_{CoG} = \frac{\sum_{k=1}^N \bar{y}_k T_{r=1}^N \left\{ \tau_{A_r|A'}(0) \right\}}{\sum_{k=1}^N T_{r=1}^N \left\{ \tau_{A_r|A'}(0) \right\}},$$

— для S -импликации

$$\tau_{kr}(v) = \begin{cases} 1 - v, & \text{если } k \neq r \\ \max(1 - v, v), & \text{если } k = r \end{cases}$$

$$\hat{y}_{CoG} = \frac{\sum_{k=1}^N \bar{y}_k T_2 \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'} \star^{T_2} \max(1-v, v) \right\} \right\} T_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \star^{T_2} \max(1-v, v) \right\} \right\}}{\sum_{k=1}^N T_2 \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'} \star^{T_2} \max(1-v, v) \right\} \right\} T_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \star^{T_2} \max(1-v, v) \right\} \right\}}$$

Можно организовать вычисление всех значений b_{kr} τ_{kr} , но использовать разреженные матрицы в качестве структуры данных для хранения значений где $b_{kr} > 0$.

Приведенные выкладки справедливы не только для функций принадлежностей отдельных термов, а и для набора кластеризованных в небольшие группы функций принадлежности со значительной степенью пересечения. При такой конфигурации выходного нечеткого пространства нет необходимости включать в процесс вывода правила, в которых функции принадлежности консеквента имеют низкий уровень пересечения с ф. п. правил, имеющих высокий уровень срабатывания для данного входа нечеткой системы.

2.4.3 Дефазификация по методу центра области

(center of area)

Данный метод можно сопоставить со схемой вычисления медианы случайной величины при заданном ее распределении. Формула вычисления дефазифицированного значения y_{CoA}^* имеет вид:

$$\int_{\inf Y}^{\hat{y}_{CoA}} \mu_{B'}(y) dy = \int_{\hat{y}_{CoA}}^{\sup Y} \mu_{B'}(y) dy$$

$$\begin{aligned} & \int_{\inf Y}^{\hat{y}_{CoA}} \frac{N}{T} \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \star^{T_2} I(v, \mu_{B_r}(y)) \right\} \right\} dy = \\ & = \int_{\hat{y}_{CoA}}^{\sup Y} \frac{N}{T} \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'} \star^{T_2} I(v, \mu_{B_r}(y)) \right\} \right\} dy \end{aligned}$$

2.4.4 Дефазификация по методу среднего максимума

$$\hat{y}_{MeOM} = \frac{\sum_{x \in core(B')} x}{|core(B')|},$$

где $core(B') = \{y | y \in Y \text{ and } \mu_{B'}(y) = \sup_{y' \in Y} \mu_{B'}(y')\}$.

Данный метод аналогичен по схеме вычисления моде случайной величины при заданном ее распределении. В случае унимодального вида функции принадлежности $\mu_{B'}(y)$ данный способ дефазификации можно упростить до метода максимума функции принадлежности:

$$\hat{y}_{MeOM} = \arg \max_{y \in Y} \mu_{B'}(y).$$

Тогда

$$\hat{y}_{MeOM} = \arg \max_{y \in Y} \prod_{r=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r | A'} \star^{T_2} I(v, \mu_{B_r}(y)) \right\} \right\} \quad (2.26)$$

2.5 Классификация объектов на основе нечеткого вывода с использованием нечеткого значения истинности

Описанная в разделе 2.2 нечеткая система в общем случае используется для моделирования. Тогда, можно использовать этот логический вывод для определения принадлежности некоторого объекта к каждому из заданного множества классов, т. е. с помощью этой нечеткой системы можно решать задачу многоклассовой классификации.

В классических методах классификация производится для данного объекта q с набором значений атрибутов, каждый из которых формализуется числовым значением признака. В случае использования для этого нечеткой системы, значения признаков формализуются посредством термов лингвистических переменных $\mathbf{x} = [x_1, \dots, x_n]$, совокупность значений которых формирует вектор $\mathbf{A}' = [A'_1, A'_2, \dots, A'_n]$, что соответствует посылке:

$$\langle x_1 \text{ есть } A'_1 \text{ и } x_2 \text{ есть } A'_2 \text{ и } \dots \text{ и } x_n \text{ есть } A'_n \rangle$$

При этом рассмотренный выше метод нечеткого вывода позволяет обрабатывать значения признаков объекта, заданные с использованием фазификации типа non-singleton, учитываяющей зашумленность значений этих признаков.

Пусть классификация производится для множества из m классов $\Omega = \{\omega_1, \dots, \omega_m\}$. Тогда база знаний нечеткой систем описывается набором из N правил вида:

$$R_k : \text{Если } x_1 \text{ есть } A_{k1} \text{ и } x_2 \text{ есть } A_{k2} \text{ и } \dots \text{ и } x_n \text{ есть } A_{kn}, \\ \text{то } q_k \in \omega_1(\bar{z}_{k1}) \text{ и } q_k \in \omega_2(\bar{z}_{k2}) \text{ и } \dots \text{ и } q_k \in \omega_m(\bar{z}_{km}), k \in \overline{1, N}, \quad (2.27)$$

где z_{kj} — степень принадлежности объекта q к классу ω_j в соответствии с правилом R_k . Значения \bar{z}_{kj} могут быть определены при помощи метода парных сравнений [**<empty citation>**] на основании набора объектов-образцов $Q = \{q_k\}_{k=1}^N$. Каждому значению z_{kj} можно поставить в соответствие значение лингвистической переменной z_j , которое выражается нечетким множеством B_{kj} имеющим в качестве базового множества диапазон $[0, 1]$, обозначающий вероятность принадлежности объекта к j -му классу.

В случае, если база правил представляет собой набор эталонных объектов процедура нечеткого вывода может быть упрощена за счет использования в качестве термов выходных лингвистических переменных синглтонных нечетких множеств $B_{k1}, B_{k2}, \dots, B_{km}$ с функцией принадлежности:

$$\mu_{B_{kj}}(z_j) = \begin{cases} 1, & \text{если } z_j = \bar{z}_{kj}, \\ 0, & \text{если } z_j \neq \bar{z}_{kj}. \end{cases}$$

Тогда, согласно (??):

1

Принадлежность объекта к одному из m классов затем определяется по формуле:

$$j^* = \arg \max_{j=1, m} \hat{z}_j,$$

где \hat{z}_j — дефазифицированное значение выходной лингвистической переменной z_j .

Если задача мультиклассовой классификации позволяет отнесение объекта сразу к нескольким классам, окончательное решение может быть получено

на основании условий:

$$\begin{cases} q \in \omega_j, & \text{если } \hat{z}_j \geq z_{inc}, \\ q \notin \omega_j, & \text{если } \hat{z}_j \leq z_{exc}, \\ \text{не определено,} & \text{если } z_{exc} < \hat{z}_j < z_{inc}, \end{cases}$$

где числа z_{inc} и z_{exc} — фиксированные пороговые значения, такие, что:

$$0 \leq z_{exc} \leq z_{inc} \leq 1.$$

2.6 Прогнозирование временных рядов на основе нечетких систем логического типа с использованием нечеткого значения истинности

Определив НЗИ $CP(\mathbf{A}_k, \mathbf{A}')$ для антецедента правила R_k согласно (2.5) и (2.10), можно переписать правило R_k в виде:

$$R_k : \text{Если } CP(\mathbf{A}_k, \mathbf{A}'), \\ \text{то } y_{t+1} \text{ есть } A_{k,p+1}, \quad k \in \overline{1, N}.$$

Вычисленное значение истинности для каждого правила выражает соответствие среза измерений порождаемых некоторым величины некоторой единице знаний о динамике этого процесса.

Если используется логический метод вывода на основе дефазификации по центру тяжести, то согласно (??) нечеткий вывод выражается:

$$\hat{y}_{t+1} = \frac{\int_Y y_{t+1} \prod_{k=1, N}^T \left\{ \sup_{v \in [0, 1]} \left\{ \tau_{\mathbf{A}_k | \mathbf{A}'}(v) \stackrel{T_2}{\star} I(v, \mu_{A_{k,p+1}}(y_{t+1})) \right\} \right\} dy_{t+1}}{\int_Y \prod_{k=1, N}^T \left\{ \sup_{v \in [0, 1]} \left\{ \tau_{\mathbf{A}_k | \mathbf{A}'}(v) \stackrel{T_2}{\star} I(v, \mu_{A_{k,p+1}}(y_{t+1})) \right\} \right\} dy_{t+1}} \quad (2.28)$$

2.7 Выводы

1. С использованием нечеткого значения истинности можно построить альтернативный метод нечеткого логического вывода. Данный метод соответствует каноническому логическому выводу Заде за счет

эквивалентного преобразования пространства антецедента правил. Преобразование истинностной модификации отображает многомерное пространство антецедента в одномерное пространство истинности, из-за чего база правил приобретает новую структуру правил вида: «Если истинно, то B_k ». Вычисление посылки нечеткого значения истинности составляет предшествующий непосредственно логическому выводу этап, и формируется из нахождения НЗИ между нечетким множеством в антецеденте и входной посылки по каждому входу отдельно и последующей свертки в единое пространство НЗИ с помощью расширенной t -нормы — \tilde{T} .

2. Для предложенного метода нечеткого вывода применимы различные способы дефазификации выходного значения. Схема вычисления дефазификации по методу среднего центра значительно упрощается. Вычисление дефазификации по упрощенной схеме сводится к нахождению величины меры возможности 1. В случае, когда нечеткие множества удалены друг от друга и имеют низкую степень взаимного пересечения итоговые формулы дефазификации упрощаются за счет. **Это позволяет...**
3. Показано как ... применен для решения задач классификации объектов с нечеткими атрибутами и для прогнозирования временных рядов.

Проведен анализ...

В главе приведены итоговые формулы нечеткого вывода на основе нечеткого значения истинности для различных методов дефазификации. Также выведены упрощенные формулы вычисления дефазифицированного значения для различных специальных категорий импликаций.

Дано формальное описание применения разработанной модели нечеткого логического вывода в задаче прогнозирования временных рядов. Приведены существующие подходы фазификации значений временных рядов и методы автоматического подбора параметров термов лингвистической переменной и формирования базы правил на основе наборов обучающих данных.

Глава 3. Программная реализация разработанного метода нечеткого вывода с применением технологии CUDA

Согласно схемам [`<empty citation>`] изображенным в предыдущем разделе, для достижения высокой производительности процедура нечеткого вывода может быть скомпонована из последовательности параллельных участков независимых вычислений и сверток. Распространенным подходом реализации параллельных вычислений является использование вычислений на графическом процессоре.

Для удобства реализация выполнялась не посредством прямого использованием API библиотеки CUDA, а с помощью библиотеки реализации высокопроизводительных вычислений - **Kokkos** [KokkosWiki; KokkosCarterEdwards20143202], предназначеннной для портативного и эффективного параллельного программирования на различных аппаратных архитектурах, включая многоядерные процессоры, графические процессоры NVIDIA/AMD и другие ускорители. Интерфейс библиотеки позволяет абстрагироваться от деталей низкоуровневого параллелизма, позволяя разработчикам писать код на C++ с одним исходным кодом, который может быть оптимизирован для различных платформ без существенных изменений. Ключевые функциональные возможности библиотеки упростили выполнение программного реализации разработанного выше метода:

- Модели параллельного выполнения: абстрагирование параллельных циклов (например, `parallel_for`, `parallel_reduce`) и рабочих процессов на основе задач.
- Управление памятью: Автоматизированная обработка пространств памяти (например, между хостом и устройством) и расположением данных для оптимизации схем доступа и минимизации объема передаваемых данных.
- Поддержка серверной части: интеграция с такими моделями программирования, как CUDA, HIP, OpenMP и SYCL, для обеспечения кроссплатформенной совместимости.
- Переносимость производительности: Обеспечивает эффективное использование ресурсов (потоки, векторизация) с учетом особенностей каждой архитектуры.

Kokkos широко используется в научных вычислениях и HPC-приложениях, упрощая разработку масштабируемых кодов при сохранении производительности на постоянно развивающемся оборудовании.

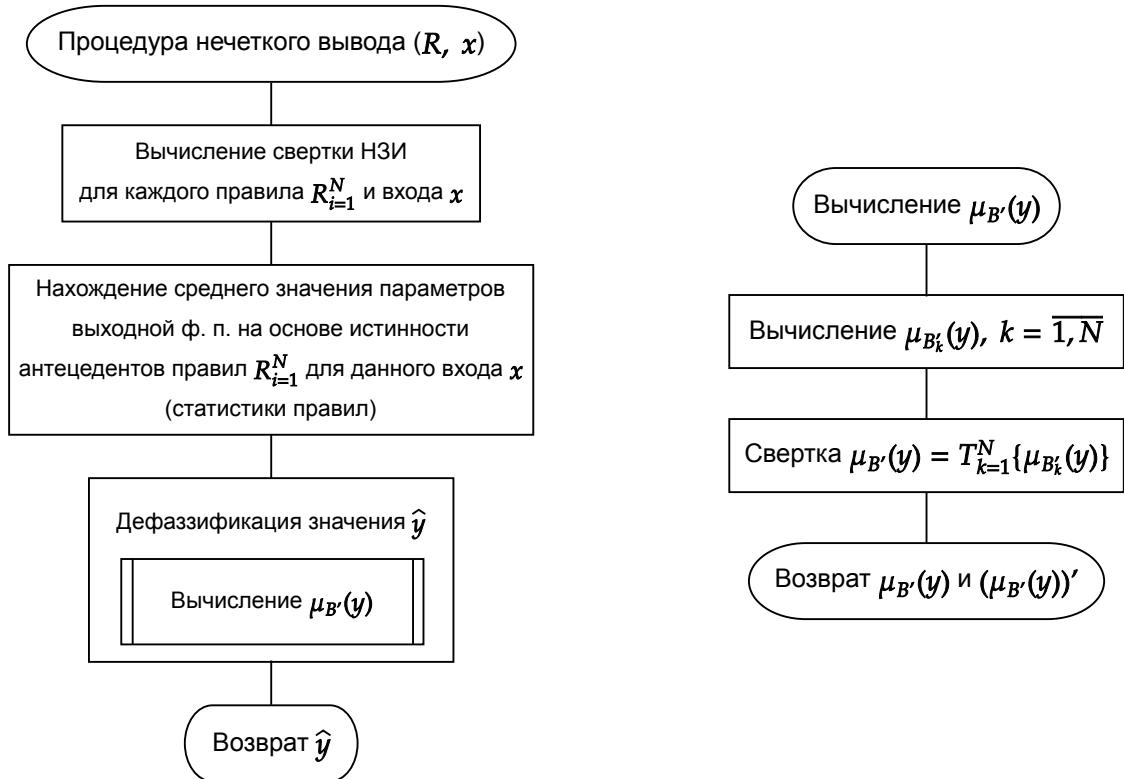


Рисунок 3.1 — Блок схема процедуры нечеткого логического вывода на основе нечеткого значения истинности.

При выполнении процедуры нечеткого вывода для набора входных данных можно выполнять процедуру вывода по каждому входному экземпляру независимо, то есть параллельно. Сам алгоритм нечеткого вывода не предусматривает наличия промежуточной информации, которой бы нужно было обмениваться между вычислениями по другим экземплярам входных данных. Тогда, поскольку алгоритм нечеткого вывода для каждого экземпляра входных данных можно реализовать при ограниченном небольшом объеме входных и промежуточных данных, имеет смысл разместить эти данные в памяти представляющей наибольшую скорость доступа к данным, которая в технологии CUDA соответствует разделяемой памяти внутри CUDA-блока, а сами вычисления над этими данными также реализовать внутри этого CUDA-блока.

Время нечеткого вывода может быть сокращено за счет уменьшения времени работы отдельного CUDA-блока путем увеличения количества параллельных CUDA-нитей. Тогда планировщик потокового мультипроцессора

(*streaming multiprocessor*) сможет распределять инструкции между доступными арифметико-логическими модулями, обеспечивая *instruction pipelining*, или сократить амортизированное время задержки (*latency*) при выполнении длительной инструкции подгрузки данных из глобальной памяти графического процессора. Однако с другой стороны слишком большое количество нитей внутри CUDA-блока упрется в ограниченное количество регистровой памяти и ограниченное количество арифметико-логических модулей внутри потокового мультипроцессора (распределемых между нитями 4-мя планировщиками на большинстве версий наборов вычислительных возможностей (*compute capabilities*)). Ограничность объема разделяемой памяти на один потоковый мультипроцессор также может ограничить предельное число CUDA-блоков, размещаемых в одном потоке мультипроцессора. Согласно документации CUDA, устройства с набором вычислительных возможностей версии 7.5 (и некоторых более низких версий) максимальный объем разделяемой памяти на одном потоковом мультипроцессоре равен 64 Кб, а на устройствах с набором вычислительных возможностей более высокой версии — предусматривается 100 - 164 Кб разделяемой памяти. Рациональное использование только необходимого объема разделяемой памяти позволит планировщику на графическом процессоре поставить на исполнение сразу несколько CUDA-блоков на один потоковый мультипроцессор.

Суммарный объем разделяемой памяти, необходимый для работы процедуры нечеткого вывода для одного входного экземпляра, складывается из объемов памяти используемых в этом выводе данных. Основной вклад в этот объем делают компоненты данных:

- параметры ф. п. N правил при n входах (2 параметра для гауссовой ф-и): $(\text{sizeof float}) \times (n + 1) \times N \times 2$;
- параметры ф. п. входного экземпляра данных при n входах (2 параметра для гауссовой ф-и): $(\text{sizeof float}) \times (n + 1) \times 2$;
- значения свертки НЗИ для N правил при размере расчетной сетки равному D_{ftv} : $(\text{sizeof float}) \times D_{ftv} \times N$ и промежуточные данные:
 - индексы координат расчетной сетки с максимальным значением НЗИ каждому из N правил и n входов: $(\text{sizeof char}) \times n \times N \times 2$;
 - максимальные значения НЗИ на текущей итерации свертки НЗИ по каждому из N правил и n входов: $(\text{sizeof float}) \times n \times N \times 2$;

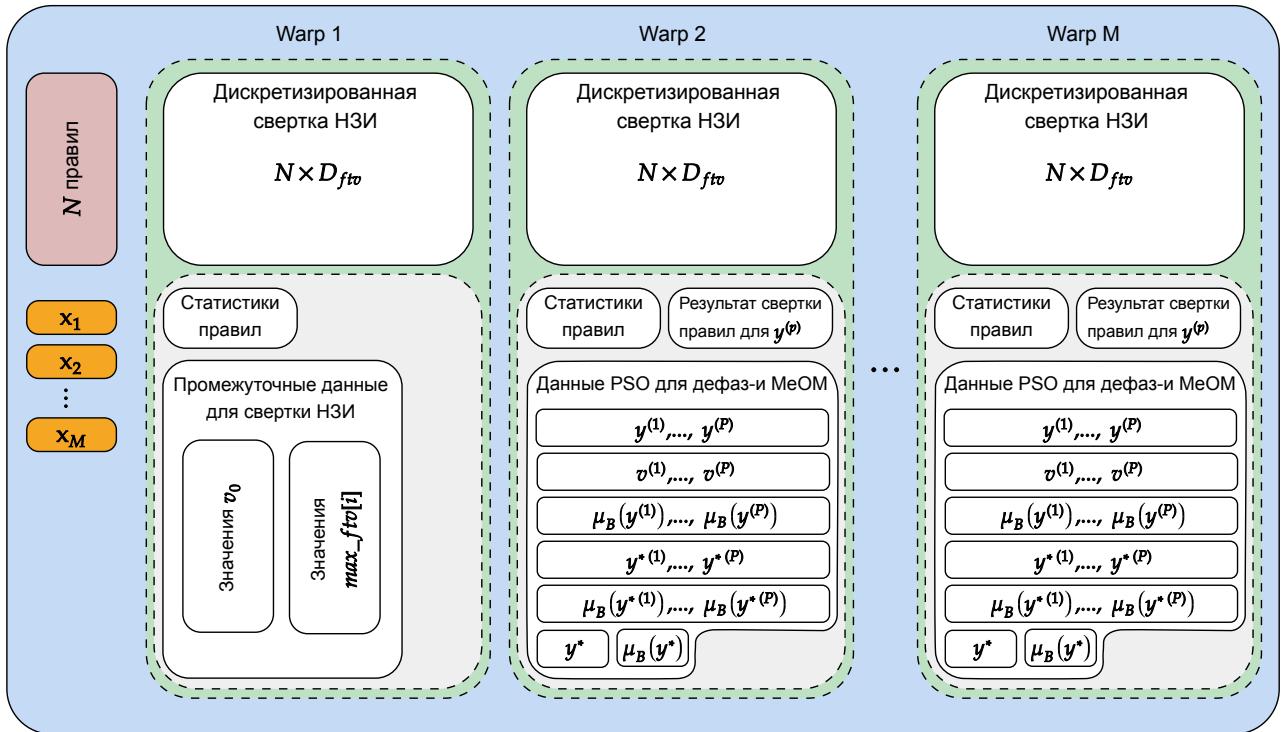


Рисунок 3.2 — Карта разделяемой памяти одного CUDA-блока нечеткого логического вывода на основе нечеткого значения истинности при использовании дефазификации МеОМ.

- популяции из P координат $y^{(p)}$, скоростей изменения координат $v^{(p)}$, значений выходной ф. п. $\mu_B(y^{(p)})$, координат локальных оптимумов $y^{*(p)}$, значений выходной ф. п. в локальных оптимумах $\mu_B(y^{*(p)})$ (5 популяций): (`sizeof float`) $\times P \times 5$.

Последний пункт приведен для случая использования *MeOM* в качестве метода дефазификации.

При нечетком выводе для одного входного экземпляра в одном CUDA-блоке, значимая доля вычислительного времени будет потрачена на загрузку базы правил из глобальной памяти в разделяемую память, а, в случае когда данные одного CUDA-блока будут занимать значимую долю доступной разделяемой памяти, число «подменных» CUDA-блоков на один потоковый процессор будет низким. Лучшим решением будет реализация «долгоживущих» CUDA-блоков, каждый из которых занимает большую часть разделяемой памяти. В таком случае база правил подгружается в разделяемую память один раз, а нечеткий вывод производится сразу для пакета экземпляров входных данных. Карта организации разделяемой памяти внутри одного такого CUDA-блока изображена на рисунке 3.2.

При достаточном размере пакета данных, обрабатываемых внутри одного CUDA-блока, нечеткий вывод по отдельному экземпляру выгодно реализовать внутри отдельного набора из 32-х нитей (*warp*), которые исполняются единым пакетом нитей вычисления. Такой способ организации вычислений позволит практически полностью избавиться от барьерной синхронизации между нитями всего CUDA-блока с использованием `__syncthreads()`, а для реализации эффективной свертки внутри набора из 32-х нитей CUDA предоставляет набор *intrinsic*-функций — `__shfl()`, `__shfl_down()`, `__shfl_up()` и `__shfl_xor()`.

Из описанного ранее функционала библиотеки Kokkos для размещения и доступа к данным в разделяемой памяти CUDA-блока крайне удобно использовать `Kokkos::View` с опорой на широко эксплуатируемую в среде C++ разработчиков идиому RAII (Resource acquisition is initialization — получение некоторого ресурса неразрывно совмещается с инициализацией, а освобождение — с уничтожением объекта), которая позволяет избавиться от необходимости «ручного» вычисления адресов и смещений, располагаемых в разделяемой памяти, программных объектов, что особенно актуально при переиспользовании сегментов разделяемой памяти, занимаемых временными данными, в дальнейших шагах алгоритма.

Для удобства ...

```
using ScratchSpace =
    typename Kokkos::Cuda::scratch_memory_space;
template <typename DataType>
using ScratchView =
    Kokkos::View<DataType,
        ScratchSpace,
        Kokkos::MemoryTraits<Kokkos::Unmanaged>>;
```

Для выделения памяти с помощью `Kokkos::View` требуется получить дескриптор разделяемой памяти текущего CUDA-блока с помощью вызова `Kokkos::TeamHandleConcept<>::team_shmem()`, который описывает текущее состояние разделяемой памяти данного CUDA-блока.

Поскольку вычисление свертки НЗИ для всех правил нечеткой системы составляет половину сложности алгоритма нечеткого вывода и в последствии будет неоднократно использоваться при агрегации правил, необходимо обеспечить единоразовое вычисление функции принадлежности свертки НЗИ (или ее

аппроксимации), в результате которого будут известны значения последней на всей ее области определения.

3.1 Вычисление нечеткого значения истинности посредством дискретизации

При реализации вычисления НЗИ, когда функции принадлежности нечетких множеств A и A' заданы гауссовыми функциями, может возникнуть сложность, когда ф. п. НЗИ вырождается в близкий к некоторой асимптоте (горизонтальной или вертикальной) вид. Для изучения условий возникновения таких ситуаций перепишем формулу (1.2) ниже и обозначим ее составные компоненты через α , β , $\varphi(v)$:

$$\exp \left(- \left(\underbrace{\frac{a_1 - a_2}{b_2}}_{\alpha} \pm \underbrace{\frac{b_1}{b_2}}_{\beta} \underbrace{\sqrt{-\ln v}}_{\varphi(v)} \right)^2 \right).$$

Тогда вычисление значения (1.2) можно представить в виде композиции функций

$$\mu_{CP(A,A')}(v) = \exp(-z(v)^2), \quad (3.1)$$

и

$$z(v) = \alpha \pm \beta \varphi(v). \quad (3.2)$$

Для иллюстрации условий возникновения описанных вырождений функции (1.2) графики функций (3.1) и (3.2) при $\alpha = \beta = 1$ приведены на рисунках 3.3 и 3.4 соответственно. Из рисунка 3.4 видно, что вырождение выражения (3.1) в близкий к асимптоте вид возможно при очень больших или очень малых значениях α и β . Данное наблюдение проиллюстрировано на рисунке 3.5.

В ситуациях, изображенных на рисунках 3.5а, 3.5б, 3.5г и 3.5д, функция принадлежности НЗИ имеет своей асимптотой вертикальную прямую $v = v_0$. При реализации вычисления НЗИ с использованием расчетной сетки, связанная с тем, что ф. п. НЗИ в точке может быть вообще не определена (рис. 3.5а и 3.5д) или ф. п. НЗИ в точке v_0 имеет значение сильно отличающееся от значений в

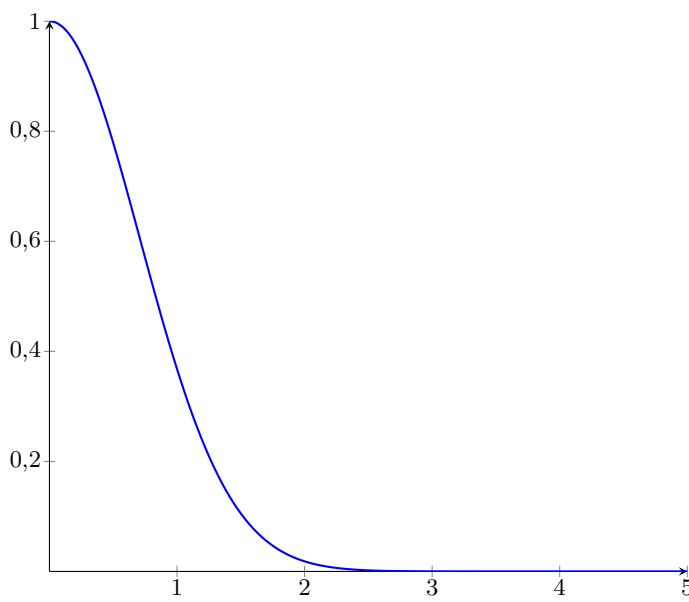


Рисунок 3.3 — График функции $\exp(-z^2)$

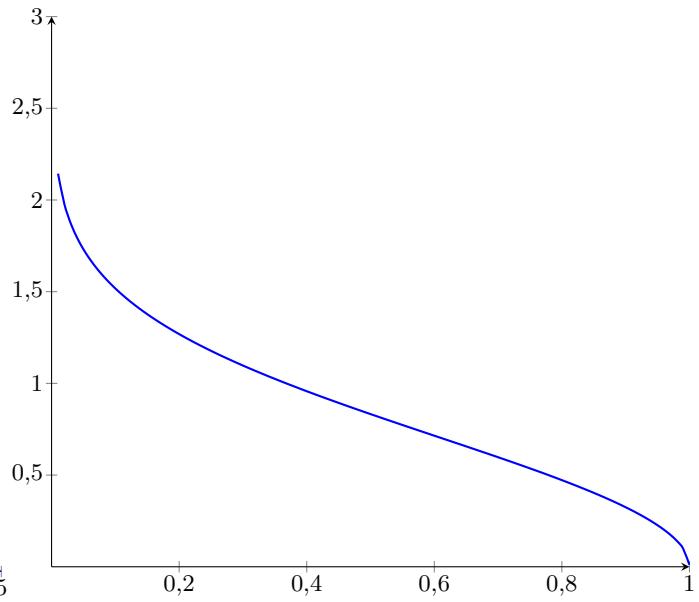


Рисунок 3.4 — График функции $\varphi(v) = \sqrt{-\ln v}$

ближайших точках расчетной сетки v_1 и v_2 при $v_0 \in (v_1, v_2)$, из-за чего значение НЗИ в бесконечно малой окрестности точки v_0 будет упущено.

Из рисунка 3.3 видно, что максимальное значение выражения (3.1), равное 1, достигается при $z = 0$, а из рисунка 3.4 видно, что функция (3.2) обязательно пересекает ось абсцисс, то есть максимальное значение ф. п. НЗИ всегда равно 1. Тогда для решения проблемы «просеивания» значения $\mu_{CP(A,A')}(v_0)$ сквозь точки расчетной сетки можно вычислить координату точки v_0 , в которой функция (1.2) принимает свое максимальное значение, а затем положить значение в ближайшей к v_0 точке расчетной сетки равным этому максимуму, то есть значению 1. Для этого выразим аналитически значение v_0 , в котором (1.2) принимает значение 1.

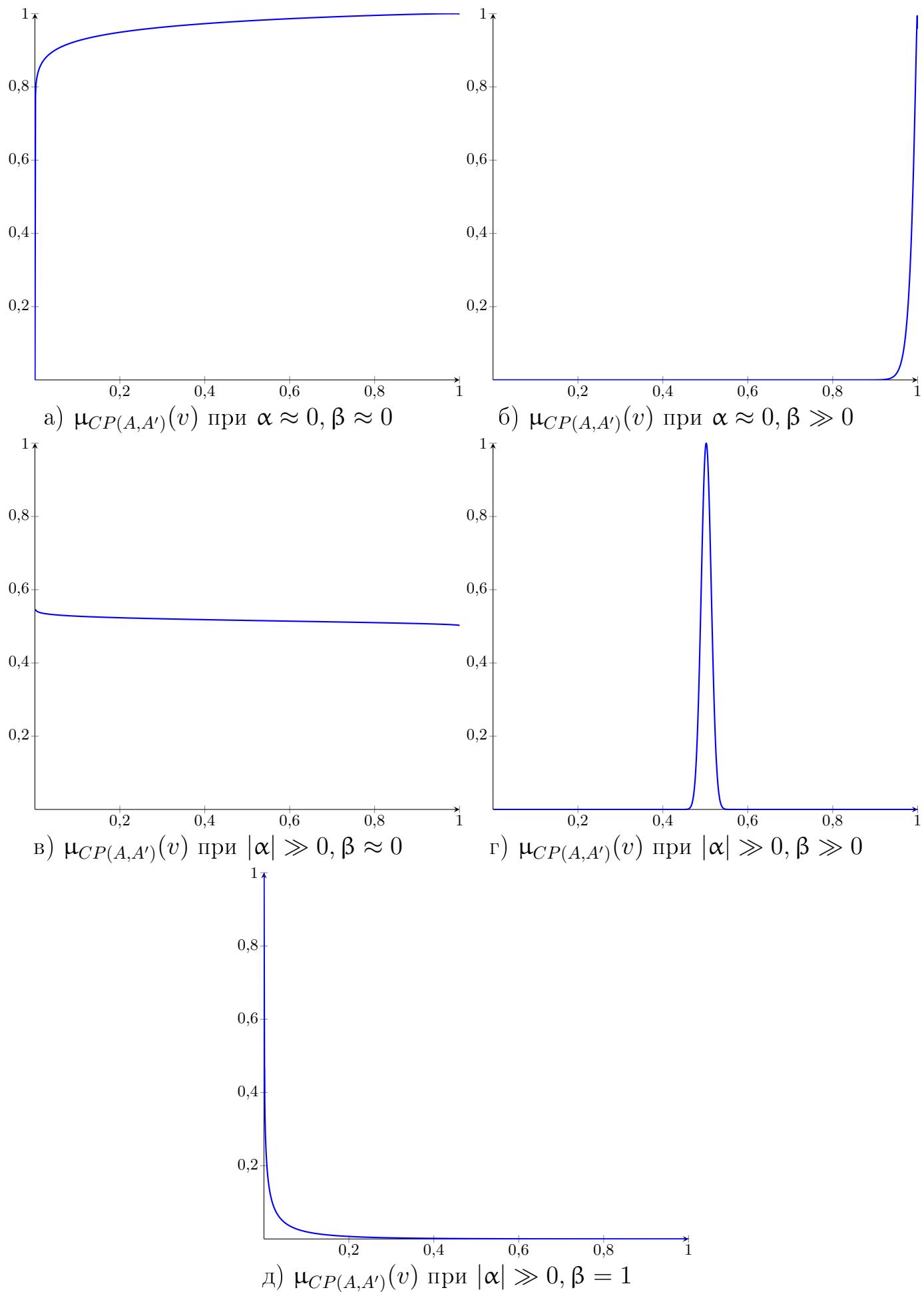


Рисунок 3.5 — Случаи вырождения ф. п. НЗИ в близкий к асимптоте вид.

$$\begin{aligned}
& \exp \left(- \left(\frac{(a_1 - a_2) \pm b_1 \sqrt{-\ln v_0}}{b_2} \right)^2 \right) = 1 \\
& \exp \left(- \left(\frac{(a_1 - a_2) \pm b_1 \sqrt{-\ln v_0}}{b_2} \right)^2 \right) = \exp(0) \\
& \frac{(a_1 - a_2) \pm b_1 \sqrt{-\ln v_0}}{b_2} = 0 \\
& \pm b_1 \sqrt{-\ln v_0} = a_1 - a_2 \\
& \sqrt{-\ln v_0} = \pm \frac{a_1 - a_2}{b_1} \\
& \ln v_0 = - \left(\frac{a_1 - a_2}{b_1} \right)^2 \\
& v_0 = \exp \left(- \left(\frac{a_1 - a_2}{b_1} \right)^2 \right) \quad (3.3)
\end{aligned}$$

Внедрение выражения (3.3) в алгоритм вычисления НЗИ позволит обеспечить соответствие определению дискретизированного представления ф. п. НЗИ. Строго говоря ф. п. на рисунке определена в точке v_0 и не требует... НЗИ с функцией принадлежности, имеющей горизонтальную асимптоту, как на рисунке 3.5в может вызвать сложность при построении алгоритма вычисления НЗИ на использовании метода градиентного спуска вместо дискретизации с фиксированным шагом расчетной сетки.

Вычисление нечеткого значения истинности для каждого правила по каждому входу отдельно предварительным этапом приведет к линейному увеличению объема необходимой памяти. Поскольку вычисление НЗИ является довольно легкой операцией, лучшим решением будет встраивание вычисления НЗИ по каждому входу в процедуру свертки НЗИ.

Формула (??) предлагает построение процедуры свертки НЗИ в виде дерева попарных сверток, для сохранения промежуточных результатов которых потребуется выделение объема памяти, также имеющего фактически линейный порядок зависимости от количества входов нечеткой системы. Возвращаясь к формуле (??) свертки НЗИ по входам в [] предложен параллельный алгоритм свертки НЗИ на расчетной сетке с постоянным шагом. Данный алгоритм итеративно вычисляет значение свертки НЗИ в каждой точки расчетной сетки продвигаясь от точки в пространстве истинности 1 к точке 0. Для каждого

правила отдельно потребуется объем памяти, необходимый для сохранения значений свертки НЗИ в каждой точке расчетной сетки и для сохранения текущего максимального значения НЗИ по каждому входу на данной итерации алгоритма.

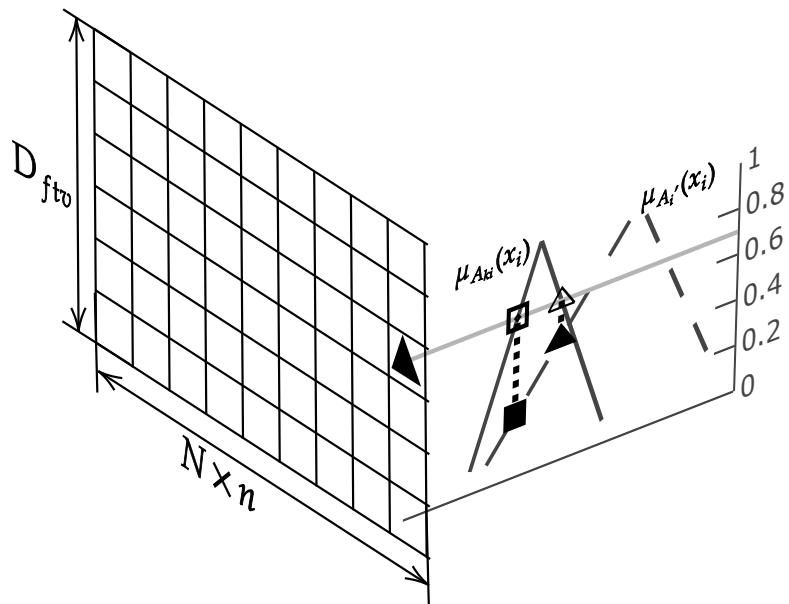
Преимуществом такого подхода представления НЗИ в программе является возможность быстрого нахождения значения функции принадлежности НЗИ в некоторой точке пространства истинности. Пара ближайших к этой точке позиций в массиве значений НЗИ на расчетной сетке определяется на основе значения фиксированного шага расчетной сетки. Для вычисления непосредственно значения НЗИ используется линейная интерполяция для найденной пары.

Поскольку при небольшом количестве входных переменных нечеткой системы большие участки расчетной сетки ф. п. свертки НЗИ соответствуют ф. п. НЗИ по одному из входов, при том же объеме памяти, необходимой для хранения результата сверки НЗИ, можно перейти к переменному шагу расчетной сетки с увеличением сложности функции аппроксимации участков ф. п. свертки НЗИ, принадлежащий ф. п. НЗИ одного из входов. Схема вычислений такого алгоритма схожа со схемой алгоритма при использовании постоянного шага расчетной сетки, а на каждой итерации определяется точка ...

Ввиду необходимости использования большой размерности расчетной сетки для получения высокой точности дискретизации для размещения результата свертки НЗИ потребуется существенный объем разделяемой памяти. Например, при использовании для задания вычисленных значений вещественных чисел одинарной точности (4 байта) и размерности расчетной сетки равной 100 сохранение результата свертки НЗИ в нечеткой системе с 50 правилами в базе правил потребуется $4 \text{ байта} \times 100 \times 50 = 20000$ байт разделяемой памяти.

3.2 Алгоритм свертки НЗИ при $T_1 = \min$ и T_3 - неубывающая по всем аргументам

При дискретизированном вычисление ф. п. свертки НЗИ в некоторой точке расчетной сетки $v_j \in [0,1]$ потребуется просматривать значения НЗИ в отрезке $[v_j, 1]$, то есть имеет вычислительную сложность $O(D_{ftv})$, где D_{ftv} — число точек расчетной сетки ф. п. НЗИ. Таким образом вычисление свертки НЗИ



по формуле (2.12) может быть распараллелено по точкам расчетной сетки, а нахождение $\tau_{A_k|A'}(v_j)$ использует операцию свертки, которая в имеет ограниченную возможность распараллеливания, но в целом требует большого количества повторных вычислений значений $\tau_{A_{ki}|A'_i}(v_k), v_k \in [v_j, 1]$. В [Karatach2024] предложен алгоритм вычисления свертки НЗИ сразу по всем входам с использованием техники динамического программирования, имеющей линейную зависимость от размера расчетной сетки D_{ftv} . Алгоритм приведен на алг. 1.

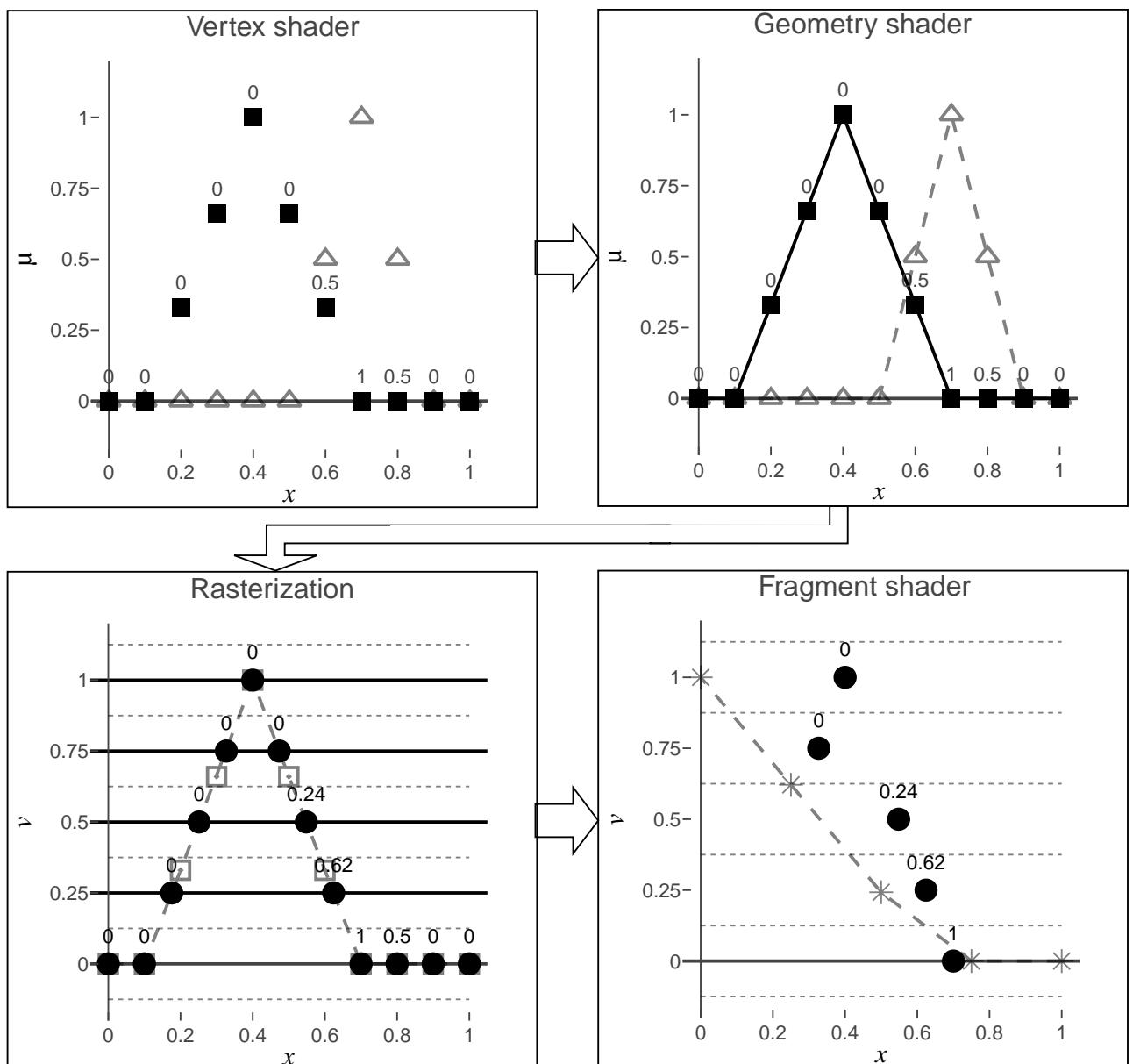
Данный алгоритм итеративно вычисляет значения $\tau_{A_k|A'}(v_j)$, продвигаясь от точки $v_j = 1$ к $v_j = 0$. При этом распараллеливаются вычисления по n входам. В i -х элементах массива \max_ftv хранятся максимальные значения $\tau_{A_{ki}|A'_i}(v_k)$ для j -й итерации, что избавляет от необходимости поиска этих значений на каждой итерации. Также, поскольку на j -й итерации значение $\tau_{A_k|A'}(v_j)$ выражается из значений НЗИ по входам в точках $v_{ki}, i = \overline{1, n}$, то, с учетом ограничения алгоритма $T_1 = \min$ и согласно выражению из формулы (2.10)

$$\underset{i=\overline{1, n}}{\text{T}_1} v_{ki} = v_j,$$

для одного из значений v_{ki} необходимо выполнение $V_{ki} = v_j$. Для этого в алг. 1 используется максимальное значение $\tau_{A_{ki}|A'_i}(v_j), i = \overline{1, n}$, когда ни по одному входу не обнаружено нового максимума ф. п. НЗИ в точке v_j .

Для лучшего понимания работы алгоритма на каждой итерации изображена на рисунке 3.2.

При



3.3 Реализация дефазификации

Производительность дефазификации по упрощенной схеме метода COG может быть увеличена путем масштабирования вычислений внутри блока по центрам выходной лингвистической переменной.

Поскольку используемые в данной работе в качестве функций принадлежности гауссовые функции являются гладкими во всей области определения выходной лингвистической переменной, для точного нахождения максимального значения ф. п. выходного нечеткого множества при дефазификации по методу среднего максимума (*MeOM*) можно использовать алгоритм градиент-

Algorithm 1 Алгоритм свертки НЗИ при $T_1 = \min$ и $T_3(a, b) \geq T_3(c, d)$ если $a > c$ или $b > d$

Require: $ftv_i, i = \overline{1, n}$

$max_ftv[i] = 0;$

for $v_j = 1 \dots 0$ **do**

$s \leftarrow \{ftv_i[v_j] \mid ftv_i[v_j] \geq max_ftv[i]\};$

$max_ftv[i] \leftarrow \max(max_ftv[i], ftv_i[v_j]);$

$v_max \leftarrow \max_i \{ftv_i[v_j]\}, v_max_index \leftarrow \arg \max_i \{ftv_i[v_j]\};$

if $s = \emptyset$ & $i = v_max_index$ **then**

$r[i] \leftarrow v_max;$

else

$r[i] \leftarrow max_ftv[i];$

end if

$ftv'[v_j] \leftarrow \underset{i}{T_3} \{r[i]\};$

end for

return ftv'

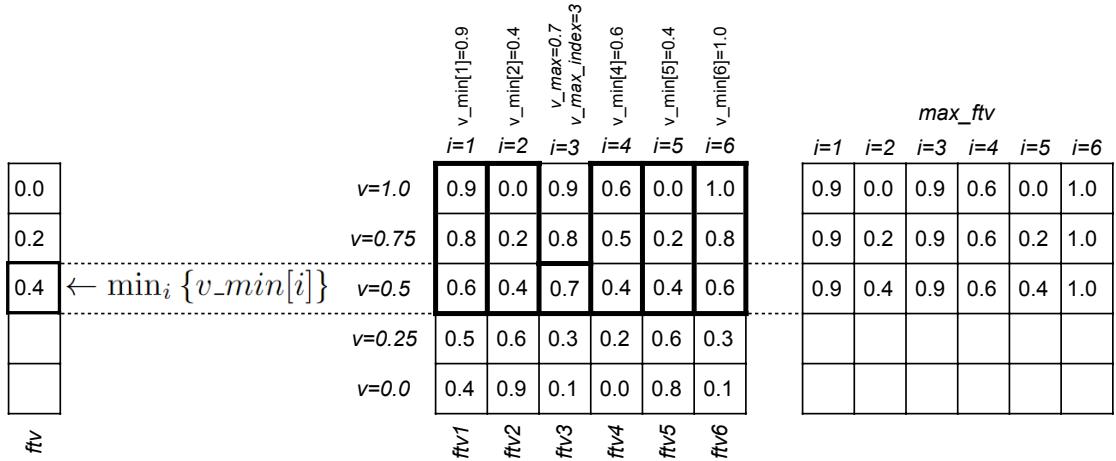


Рисунок 3.6 — Иллюстрация работы алгоритма свертки НЗИ для нескольких входов для расширенной \tilde{T} -нормы при $T_1 = \min$ и $T_3 = \min$.

ногого спуска. В ситуации когда вычисление значения агрегированной по всем правилам выходной функции принадлежности и ее производных в некоторой точке выходного пространства является вычислительно сложной процедурой, целесообразно будет использовать *метод Ньютона второго порядка* для более быстрого нахождения точки максимума выходной ф. п.:

$$y^{(k+1)} = y^{(k)} - \alpha \frac{\mu'_{B'}(y^{(k)})}{\mu''_{B'}(y^{(k)})}, \quad (3.4)$$

где α - коэффициент шага градиента.

Сложность может составить вопрос инициализации начальной координаты точки $y^{(0)}$, ведь в отдаленных от точки максимума выходной ф. п. $\mu_{B'}(y^{(0)})$ соответствующая ее минимуму гауссова функция может принять горизонтальный вид с нулевыми производными вне границ отрезка $[a_{B'} - 4b_{B'}, a_{B'} + 4b_{B'}]$. Для получения начального приближения можно использовать вычислительно более простой метод дефазификации, например, дефазификацию по методу среднего центра (СА). Также стоит добавить в формулу (3.4) штраф за отдаление от точки \hat{y}_{CA} с весовым параметром учета штрафа λ , а также учитывать штраф при низких значениях $\mu_{B'}(y^{(k)})$. С помощью параметра λ и номера итерации k можно обеспечить плавный сдвиг вклада в шаг итерации от штрафа до значения градиента с ростом числа прошедших итераций. Тогда формула (3.4) примет вид:

$$y^{(k+1)} = y^{(k)} + \alpha \left(\mu_{B'}(y^{(k)})(1 - \lambda) \left(-\frac{\mu'_{B'}(y^{(k)})}{\mu''_{B'}(y^{(k)})} \right) + (1 - \mu_{B'}(y^{(k)}))\lambda(\hat{y}_{CA} - y^{(k)}) \right),$$

$$\lambda = e^{-k\gamma},$$

где хорошая сходимость алгоритма показывается при выборе для параметра γ значения равного коэффициенту шага α .

Для преодоления проблемы локальных максимумов выходной функции принадлежности (которые ожидаемо возникнут при использовании t -нормы минимума или произведения для агрегации гауссовых функций) и сглаживания линии градиента, распространенной практикой является добавление момента $v^{(k)}$ в алгоритм градиентного спуска:

$$v^{(k+1)} = \omega v^{(k)} + (\mu_{B'}(y^{(k)})(1 - \lambda) \left(-\frac{\mu'_{B'}(y^{(k)})}{\mu''_{B'}(y^{(k)})} \right) + (1 - \mu_{B'}(y^{(k)}))\lambda(\hat{y}_{CA} - y^{(k)})),$$

$$y^{(k+1)} = y^{(k)} + \alpha v^{(k+1)}, \quad (3.5)$$

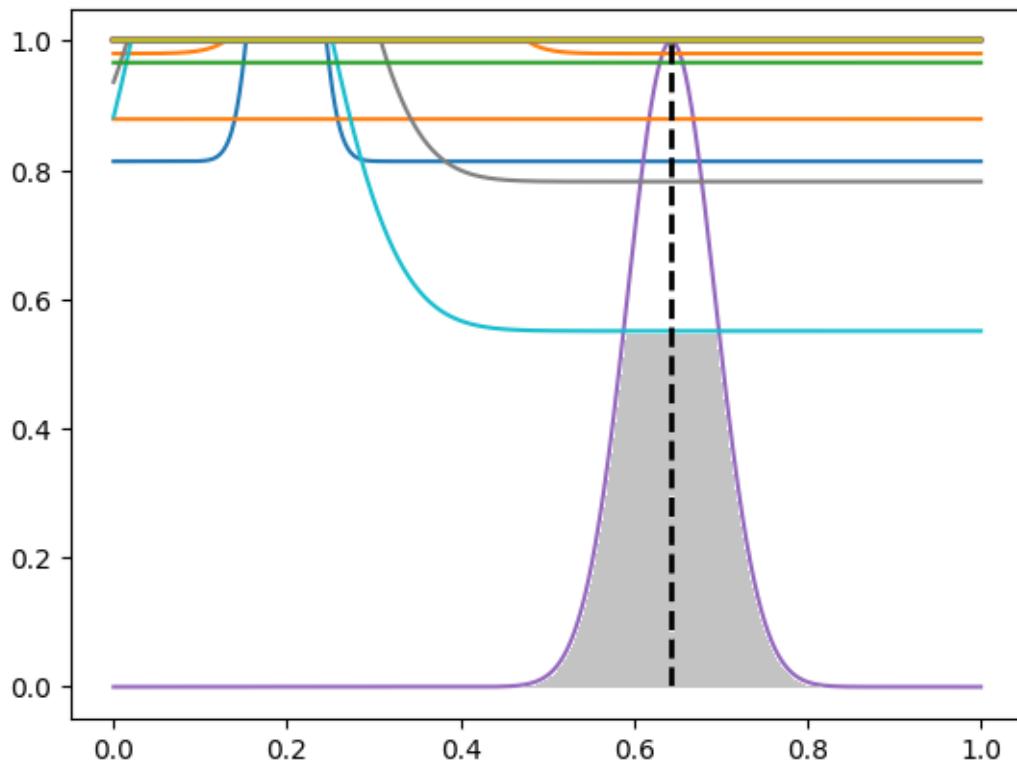


Рисунок 3.7 — Ситуация необходимости нахождения среднего максимума при дефазификации агрегированной функции принадлежности $\mu_{B'}(y)$.

где ω — параметр скользящего среднего значения момента.

Необходимость усреднения максимумов при использовании метода дефазификации Mean of Maximum возникает в ситуации изображенной на рисунке 3.7, где истинное значение y нарисовано пунктирной линией. Вычислить среднее значение y можно как выборочное среднее. Получить такую выборку точек можно посредством многократной оптимизации описанном выше методом. Как и в случае с упрощенной схемой дефазификации каждая такая оптимизация может происходить в отдельной группе нитей (warps) за счет масштабирования одного CUDA-блока.

Для этого можно использовать один из методов глобальной оптимизации, например, *Gradient-aware Particle Swarm Optimization*. Метод Particle Swarm Optimization (PSO) обеспечивает синхронизацию информации о текущей глобальной точке оптимума на данной итерации внутри популяции точек, что сокращает число итераций до сходимости к максимальному значению выходной функции принадлежности. С учетом (3.5) для данного метода формулы

шага оптимизации можно составить следующим образом:

$$\begin{aligned}
 v_i^{(k+1)} &= \omega v_i^{(k)} + \\
 &+ \alpha_l \left(r_l \left(y_i^{best} - y_i^{(k)} \right) + (1 - r_l) \left(-\frac{\mu'_{B'}(y_i^{(k)})}{\mu''_{B'}(y_i^{(k)})} \right) \right) + \\
 &+ \alpha_g \left(r_g \mu_{B'}(y^{best}) (y^{best} - y) + (1 - r_g) (1 - \mu_{B'}(y^{best})) (\hat{y}_{CA} - y^{(k)}) \right), \\
 y^{(k+1)} &= y^{(k)} + v^{(k+1)},
 \end{aligned} \tag{3.6}$$

где α_l и α_g — коэффициенты шага в направлении текущей точки локального и глобального оптимума, r_l и r_g — случайно сгенерированные числа в диапазоне $[0,1]$, параметризующие движение в сторону текущих точек локального и глобального оптимума алгоритма PSO между движением в сторону роста градиента и точки начального приближения \hat{y}_{CA} соответственно.

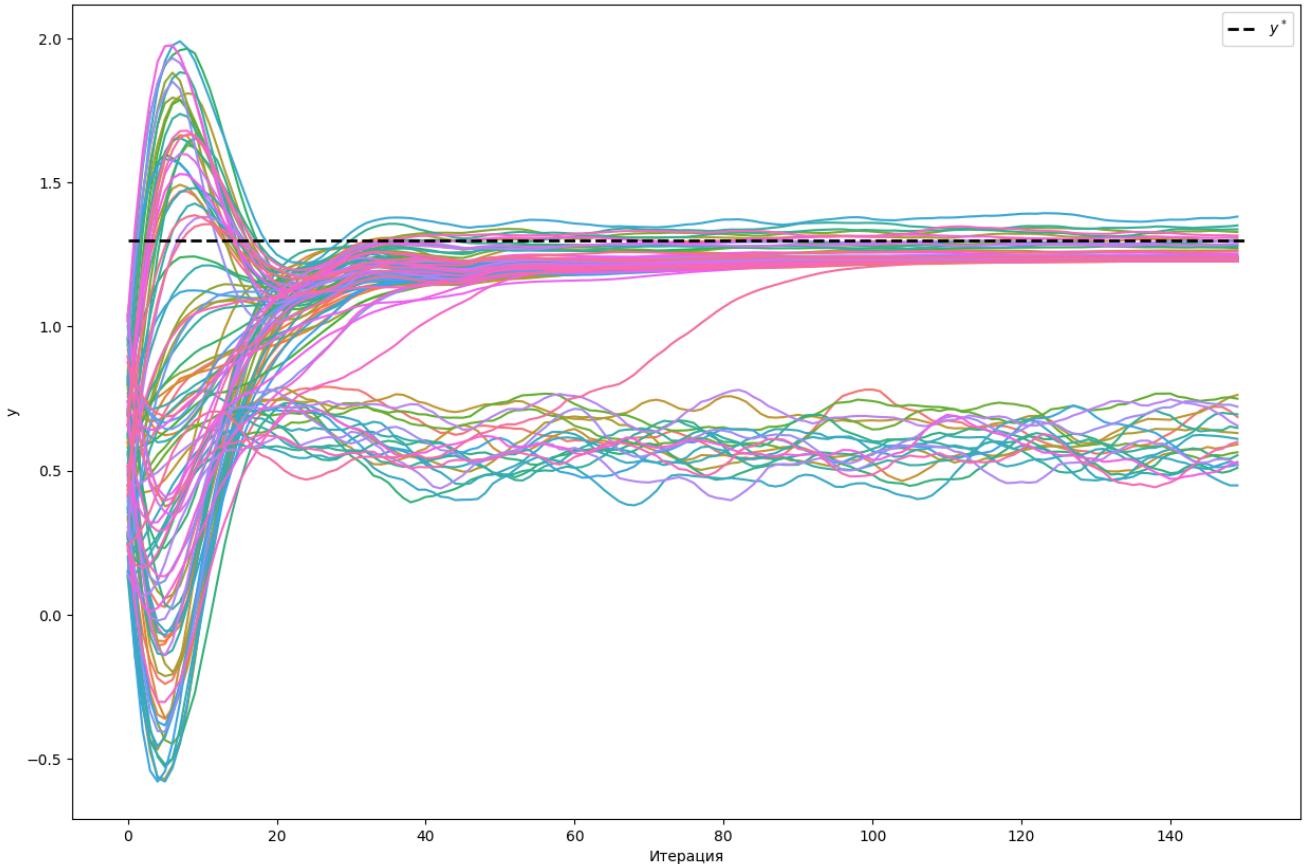


Рисунок 3.8 — Динамика движения точек популяции при работе алгоритма глобальной оптимизации Gradient-aware PSO.

Компонента после коэффициента α_l в формуле (3.6) обеспечить разнообразие среди точек популяции, достигших максимума $\mu_{B'}(y_i^{(k)})$. График работы

такого алгоритма PSO приведен на рисунке 3.8. Как видно из него — большинство точек популяции сосредоточились возле точки оптимума.

Для реализации дефазификации по *методу центра тяжести* можно использовать алгоритм оптимизации из предыдущего метода, сохраняя координаты $y_i^{(k)}$ и значения $\mu_{B'}(y_i^{(k)})$ на каждой итерации, а затем для вычисления формулы (2.22) использовать численное интегрирование, например, методом Симпсона.

3.4 Ускорение вывода за счет фильтрации ближайших правил

Как описано в разделе ??, на основе рисунка 2.4 и графика истинности отмечается отсутствие необходимости включать в процедуру вывода для данного входа весь набор правил из базы правил, поскольку в случае отсутствия пересечения входных нечетких множеств и нечетких множеств антецедента, центроида НЗИ будет около 0, а ф. п. отношения импликации тогда примет значение равное 1 независимо от значения ф. п. консеквента. При использовании для сверки правил t -нормы из свойства $T(a, 1) = a$ вытекает отсутствие необходимости включать такие правила в процедуру свертки правил, что способно значительно сократить время выполнения процедуры нечеткого вывода, особенно при использовании методов дефазификации, предполагающих множество сверток правил при нахождения $\mu_{B'}(y)$ для оптимизацию значения \hat{y} .

Для этого предлагается включать в процедуру вывода K правил с наибольшим значением истинности по отношению к данному входу. Поскольку вычисление НЗИ в свою очередь является вычислительно сложной процедурой, в качестве эвристики предлагается использовать одну из мер расстояния между нечеткими множествами. В качестве меры совместимости нечетких множеств $\mu_{A_1}(x)$ и $\mu_{A_2}(x)$ можно использовать дополнение до единицы *нормированное Евклидово расстояние* гауссовых функций принадлежности $g_1(x; a, b)$ и $g_2(x; a, b)$ этих нечетких множеств, которое выражается формулой:

$$\begin{aligned}
 \rho(A_1, A_2) = ||g_1 - g_2|| &= \sqrt{\int_{-\infty}^{\infty} (g_1(x) - g_2(x))^2 dx} \\
 &= \sqrt{\int_{-\infty}^{\infty} g_1^2 + \int_{-\infty}^{\infty} g_2^2 - 2 \int_{-\infty}^{\infty} g_1^2 g_2^2} \\
 \int_{-\infty}^{\infty} g_1^2(x) dx &= b_1 \sqrt{\pi} \\
 \int_{-\infty}^{\infty} g_2^2(x) dx &= b_2 \sqrt{\pi} \\
 \int_{-\infty}^{\infty} g_1^2(x) g_2^2(x) dx &= \sqrt{\frac{2\pi b_1^2 b_2^2}{b_1^2 + b_2^2}} \exp\left(-\left(\frac{(a_1 - a_2)^2}{2(b_1^2 + b_2^2)}\right)\right)
 \end{aligned}$$

Для многомерного случая мера совместности:

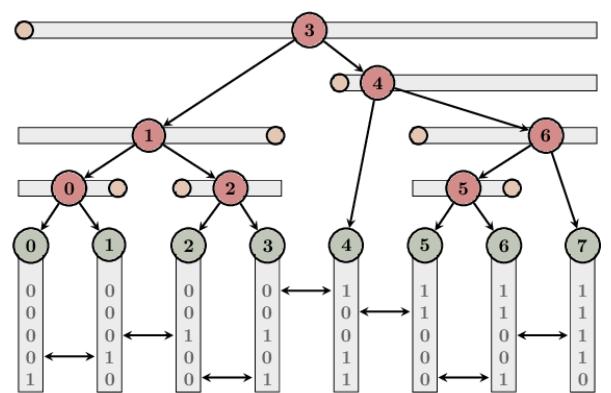
$$\rho(\mathbf{A}_1, \mathbf{A}_2) = 1 - \sqrt{\frac{1}{n} \sum_{i=1}^n \rho(A_{1i}, A_{2i})}$$

где n — размерность нечетких множеств A_1 и A_2 .

Помимо более простого способа оценки схожести входных н. м. и н. м. антецедента следует обеспечить эффективный способ нахождения K правил с наибольшей схожестью.

x: 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
y: 0 000	000000 000001	000100 000101	010000 010001	010100 010101	010100 010101	010100 010101	010100 010101
1 001	000010 000011	000110 000111	010010 010011	010110 010111	010110 010111	010110 010111	010110 010111
2 010	001000 001001	001100 001101	011000 011001	011100 011101	011100 011101	011100 011101	011100 011101
3 011	001010 001011	001110 001111	011010 011011	011100 011101	011110 011111	011110 011111	011110 011111
4 100	100000 100001	100100 100101	100101 100101	110000 110001	110001 110001	110100 110101	110101 110101
5 101	100010 100011	100110 100111	100111 100111	110010 110011	110011 110011	110110 110111	110111 110111
6 110	101000 101001	101100 101101	101101 101101	111000 111001	111001 111001	111100 111101	111101 111101
7 111	101010 101011	101110 101111	101111 101111	111010 111011	111011 111011	111110 111111	111111 111111

(a) z-curve



(b) apetrei

Рисунок 3.9 — Две иллюстрации: графика z-curve и apetrei.

В [prokopenko2024revisingapetreisboundingvolume] авторами библиотеки ArborX предложен алгоритм для эффективного построения иерархии

ограничивающих объемов. Данная структура данных используется для эффективного поиска ближайшей точки в n -мерном пространстве. Алгоритм выполняет построение структуры поиска за $O(N \times \log(N))$, где N - количество точек в исходном наборе. Логика поиска места в иерархии ограничивающих объемов для каждой отдельной точки помещена в отдельную нить параллельных вычислений, после чего остается только логарифмическая компонента сложности, соответствующая восходящему обходу промежуточного дерева и помещения точки в подобранный узел **с использованием атомарной операции**. В основе алгоритма лежит группировка близко расположенных точек для более быстрого подбора соответствующего ограничивающего объема за счет проецирования точки из n -мерного пространства на Z -кривую посредством вычисления кода Мортона для каждой точки. Согласно эвристике, лежащие рядом на Z -кривой точки располагаются достаточно близко и исходном n -мерном пространстве. Кроме того в статье описан модифицированный алгоритм Апетрея [], обеспечивающий обход иерархии на этапе поиска ближайших точек без необходимости помещения цепочки пройденных родительских узлов в стек за счет передачи информации об альтернативных узлах-кандидатах из родительских узлов в дочерние. Такой подход значительно снижает количество потребляемой памяти для сохранения промежуточной информации при выполнении этапа поиска.

Однако при таком подходе не получится

3.5 Реализация алгоритма построения базы правил

Для оценки предельного качества предложенного метода нечеткого вывода важно обеспечить метод нахождения оптимальной структуры правил, при которой нечеткая модель достигает максимального качества моделирования набора данных. Оптимизация базы правил позволяет оценить непосредственно качество моделирования, исключая неоптимальность извлечения знаний из набора данных, как при построении базы правил на основе эталонов. Поскольку композиция операций описанной в работе нейро-нечеткой системы не является гладкой функцией (например, при использовании t -нормы \min), для выполнения оптимизации не подходят методы градиентного спуска. В такой си-

туации следует использовать методы глобальной оптимизации, представителем которых являются семейство алгоритмов оптимизации на основе роевого интеллекта. Одним из них является метаэвристический алгоритм *метод роя частиц* (*particle swarm optimization, PSO*) [Kennedy1995].

В отличие от градиентного спуска, PSO не нуждается в вычислении производных ошибки. Частицы в PSO используют как собственный опыт, так и опыт соседей. Это позволяет алгоритму избегать застревания в локальных минимумах. В контексте оптимизации вектора параметров функций принадлежности базы правил большой размерности (часто, сотни параметров), PSO хорошо выполняет оптимизацию по всем размерностям одновременно. При работе алгоритм сперва исследует широкое пространство решений, а затем сосредоточиться на его уточнении, что также позволяет бороться с попаданием в локальный оптимум. Основные параметры — количество частиц, вес инерции ω и коэффициенты влияния локального φ_l и глобального φ_g оптимума. Частицы работают независимо, а вычисление функции приспособленности можно выполнять параллельно. Данный метод часто применяется для оптимизации нейро-нечетких систем в литературе.

В выполненной реализации для утилизации параллелизма графического ускорителя основная часть вычислений была перенесена внутрь CUDA-ядер. Например, вычисление функции приспособленности по каждой точке популяции производилось в отдельном CUDA-блоке. Для этого популяции частиц, вектора скорости и соответствующих оценки функции приспособленности были размещены в памяти графического устройства, а копирование в память хоста минимизировано до объема данных, необходимых для формирования аргументов CUDA-ядер шагов данного алгоритма.

3.6 Реализация обертки в Python-модуль

Для ускорения процесса отладки и тестирования, выполненная на языке C++ (CUDA) реализация нечеткой модели собирается компилятором NVCC в динамически подключаемую библиотеку (*.so (*shared object*) файл в Linux). Реализованная на языке С часть Python-обертки библиотеки выполняет поиск

файла библиотеки и символов в нем уже при запуске использующей эту библиотеку программы.

На сегодняшний день язык программирования Python занял позицию отраслевого стандарта в научных исследованиях и задачах анализа данных. Это обеспечено прежде всего удобным синтаксисом и легкостью интеграции большим многообразием высокопроизводительных библиотек научных вычислений, реализованных на языках с прямым управлением аппаратными ресурсами. Среди инструментов обеспечивающих реализацию механизма FFI в языке Python простым в использовании является инструмент Cython. Этот компилируемый метаязык расширяет парадигму Python, вводя статическую типизацию и структуры данных, характерные для системного программирования, что обеспечивает прозрачное описание C++ API. Современные инструменты пакетирования, включая обновленные версии setuptools, реализуют стандартизованные механизмы (PEP-517) для автоматизации включения Cython-компонентов в Python-пакеты.

3.7 Выводы

1. Процедура нечеткого вывода может быть представлена композицией независимых вычислений и их сверток, что делает уместным применение параллельных вычислений для организации нечеткого вывода. Эффективность реализации параллельных вычислений в технологии CUDA обеспечивается высоким уровнем задействованности различных аппаратных ресурсов на графическом ускорителе. Для исключения простоя при обращении к памяти, все необходимые для вычислений по отдельным входным экземплярам данные помещаются в разделяемую память CUDA-блоков. С целью минимизации времени на загрузку одной и той же базы правил, вместо легких взаимозаменяемых планировщиком CUDA-блоков, упор сделан на использование «долгоживущих» крупных CUDA-блоков, обрабатывающие одновременно несколько входных экземпляров. Вычисления по для каждого входного экземпляра данных вписаны в атомарный набор из 32-х вычислительных нитей (warp), из-за чего нет необходимости делать явную

- барьерную синхронизацию, а для свертки внутри `warp`-а предусмотрены специальные *intrinsic*-функции.
2. Значение НЗИ в точке v может быть выражено аналитически из гауссовых функций принадлежности н. м. антецедента A и входного н. м. A' . Согласно формулам (??) процедура дефазификация требует вычисление расширенной свертки нечеткого значения истинности в различных точках области определения НЗИ, тогда как сложность вычисление свертки НЗИ линейно зависит от количества входов нечеткой системы, а свертка в точке v зависит от значений НЗИ в области $[v, 1]$. Для решения этих ограничений выполняется кусочно-линейная аппроксимация функции свертки НЗИ на расчетной сетке заданной размерности. Из-за сложности вычисления, нахождение аппроксимации свертки НЗИ выполняется один раз по каждому правилу. Для решения проблемы «просеивания» максимальных значений НЗИ сквозь точки расчетной сетки, определяется ближайшая точка р. с. к точке с максимальным значением НЗИ. В главе предложен алгоритм свертки НЗИ по всем входам сразу.
 3. Реализованы дефазификации по методу центра тяжести (CoG) и среднего максимума (МеOM). Предусмотрена реализация дефазификации по упрощенной схеме CoG, которая может обеспечить меньшую вычислительную сложность и достаточную точность оценки выходного значения в некоторых задачах. Для задач, в которых требуется полноценная оценка выходного нечеткого множества дефазификация реализована с использованием численных методов. Дефазификация МеOM использует глобальную оптимизацию с учетом значения градиента функции принадлежности выходного н. м. (алгоритм Gradient-aware PSO) для поиска точки максимума. Дефазификация по методу GoG дополнительно выполняет численное интегрирования методом Симпсона.
 4. На основании отмеченной в разделе ?? возможности упрощения вычислений процедуры нечеткого вывода в случае удаленных антецедентов или консеквентов в реализацию нечеткого вывода внедрена оптимизация. Суть оптимизации состоит в отборе для данного вектора входных нечетких множеств правил с наибольшим пересечением антецедентов.
 5. Для удобства использования реализация нечеткой модели на языке CUDA/C++ обернута в Python-модуль с помощью инструмента инте-

грации кода на C/C++ в Python — Cython. Реализован класс нечеткой модели, предоставляющий *scikit-learn*-совместимый интерфейс обучения и вывода, а также возможность сериализации/десериализации обученной модели в файловую систему для облегчения воспроизведимости экспериментов. Интерфейс класса позволяет передавать данные в виде `numpy.ndarray`-объектов — наиболее распространенный способ в задачах научных вычислений на Python.

Глава 4. Применение разработанной нечеткой модели для прогнозирования временных рядов в задачах экономики и финансов

Прогнозирование временных рядов в экономике и финансах играет ключевую роль как инструмент анализа и предсказания динамики последовательно изменяющихся данных. Оно служит основой для принятия обоснованных решений в условиях неопределенности, обеспечивая оценку будущих тенденций, рисков и возможностей. Это позволяет субъектам экономической и финансовой деятельности — от индивидуальных инвесторов до государственных структур — оптимизировать управление ресурсами, минимизировать потери и формировать долгосрочные стратегии.

Стратегическое планирование и бюджетирование (обоснование финансовых планов, оптимизация ресурсного распределения), Управление рисками и инвестиционные решения

В государственных и корпоративных бюджетах прогнозы временных рядов используются для оценки объёмов доходов и расходов, планирования денежных потоков и определения ключевых показателей эффективности. У компаний прогнозирование сезонных и трендовых колебаний продаж помогает корректировать производственные мощности, склады и персонал, тем самым минимизируя издержки и снижая риск дефицита или перепроизводства. Прогнозы временных рядов служат основой для разработки моделей оптимального распределения активов и стратегий ребалансировки портфеля в зависимости от ожидаемых рыночных движений.

Одномерные модели оправданы, когда целевой ряд обладает высокой автокорреляцией, слабо зависит или отсутствует информация о значениях внешних факторов, ограниченны вычислительные и временные ресурсы для оценки взаимосвязей измерений. Примеры: краткосрочное прогнозирование инфляции, прогнозирование индекса S&P 500. Многомерные методы необходимы, когда влияние других временных рядов или внешних предикторов существенно для точности прогноза, доступна качественная информация об этих переменных или когда доступна информация о их будущих значениях, а также когда необходимо исследовать влияние «шока» в одной переменной на остальные. Примеры: оценка влияния нефтяных шоков на экономику Нигерии.

Область и ситуации в которой применима разработанная модель (авто-регрессионная)

Для прогнозирования используются модели

4.1 Задача прогнозирования стоимости ценных бумаг Тайваньской биржи

[Sadaei2016]

Taiwan Capitalization Weighted Stock Index (TWSE) - популярный набор данных стоимости акций биржи ТАИЕХ, используемый для оценки качества прогнозирования временных рядов. Датасет предоставляется официальным сайтом Набор данных составлен из ежедневных значений цены индекса в момент закрытия биржи в промежутке от 2001 до 2022 года.



Рисунок 4.1 — Тренировочный и тестовый отрезок значений временного ряда TWSE и среднеквадратичное отклонение гауссовой ф. п. в каждой точке.

Для оценки качества были отложены последние 500 окон временных точек, а для построения базы правил использовались предшествующие им 1000 окон. Отрезки исходного временного ряда для тестовый и тренировочный выбирались таким образом, чтобы диапазон значений в тестовом отрезке был включен в диапазон тренировочного отрезка. Фаззификация временного ряда

состояла в построении гауссовой функции принадлежности в каждой точке временного ряда. Центры гауссовой функции принадлежности равны значениям временного ряда в каждой точке. Значение среднеквадратичного отклонения находится согласно способу описанному в разделе ?? главы ??: **уточнить**.

Рисунок 4.2 — Влияние комбинации значений гиперпараметров на значение оптимизируемой метрики.

В рамках данной задачи горизонт прогнозирования задается равным 1. В качестве функции приспособленности использовалась метрика **NDEI**.

Для оценки предельных возможностей непосредственно нечеткой логической модели в данной задаче параметры α, \dots подбирались с помощью байесовской оптимизации, реализованной в Python-библиотеке Optuna [Optuna2024; akiba2019optuna]. Диапазоны значений этих параметров, в пределах которых производился поиск оптимальной комбинации, приведены в таблице. Для сокращения длительности байесовой оптимизации размер популяции выбирался равным количеству правил в базе правил, а число итераций ролевого алгоритма подбора базы правил — 50. Для алгоритма Gradient-aware PSO в методе дефазификации MeOM размер популяции составляет 30 особей, а количество итераций — 100. График значений оптимизируемой метрики для различных комбинаций значений гиперпараметров нечеткой модели в параллельных координатах изображен на рисунке 4.1. Из него можно отметить несколько наблюдений. При значение параметра лага $p = 5$, значении коэффициента среднеквадратичного отклонения функций принадлежности для фазифицированных входных данных $\varphi = 0.1$ и количестве правил в базе правил $N = 50$ обеспечивается лучшее значение целевой метрики. Также лучшее качество прогнозирования на тренировочном отрезке временного ряда достигается при использовании нечеткой импликации Лукасевича и метода дефазификации — *MeOM*.

Рисунок 4.3 — Влияние комбинации значений гиперпараметров на время построения базы правил.

На рисунке 4.1 график изображает зависимость времени оптимизации базы правил нечеткой системы при тех же комбинациях гиперпараметров, что и на рисунке 4.1.

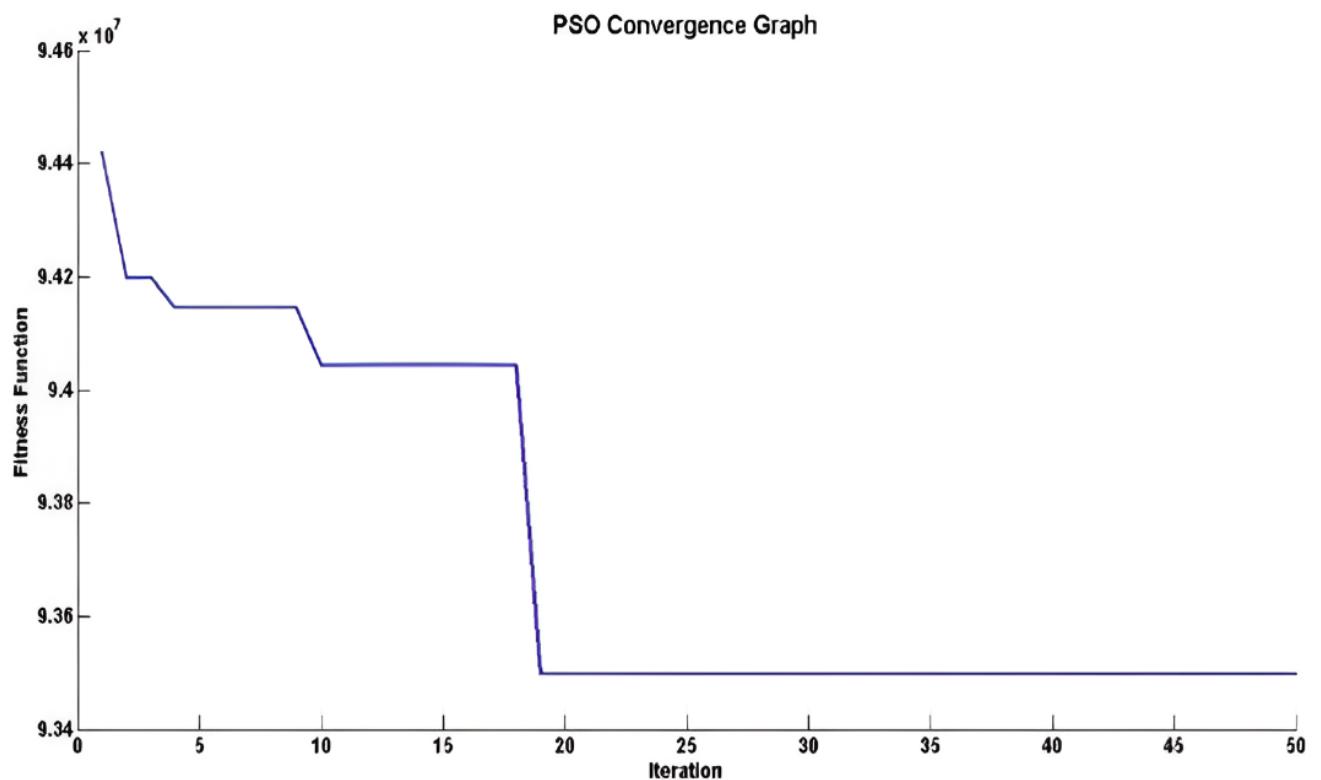


Рисунок 4.4 — Графики изменений максимального, среднего и лучшего по популяции значений оптимизируемой метрики.

Динамика максимального, среднего и лучшего по популяции значений оптимизируемой метрики в процессе PSO-оптимизации базы правил для подобранный выше комбинации гиперпараметров изображена на рисунке 4.1.

4.5

На оборудовании ????? данная реализация имеет следующую производительность

Конфигурация вычислительной системы: центральный процессор — AMD Ryzen 9 5900HX (8 ядер, 16 потоков, базовая частота 3.3 ГГц, кэш L3 16 МБ); оперативная память — 32 ГБ DDR4-3200 (двухканальный режим); графический процессор — NVIDIA GeForce RTX 3080 Laptop GPU с 16 ГБ видеопамяти (CUDA 12.1, compute capability — 8.6).

Таблица 1 — Сравнение показателей качества прогнозирования временного ряда из набора данных TWSE (TAIEX).

Модель	Время обучения, сек.	Время вывода, сек.	NDEI	MAPE	MAE
MLP	—	—	0.241711	0.0114034	—
Decision Tree	—	—	0.282676	0.0124152	—
SVM	—	—	0.537621	0.0241796	—
ANFIS (Мамдани)	—	—	0.563	0.465	—
ANFIS (Такаги-Сугено)	—	—	0.501	0.398	—
NSFLS-FTV (упрощенная схема)	1023	5.98	0.421	0.281	—
NSFLS-FTV	2746	12.84	0.251	0.132	—



Рисунок 4.5 — Демонстрация предсказанных и фактических значений в наборе данных TWSE.

4.2 Задача прогнозирования активности клиента по банковскому продукту

Разработанная модель нечеткого логического вывода также была применена для прогнозирования оборотов юридических лиц по эквайрингу. Эквайринг — это банковская услуга, позволяющая юридическим лицам принимать безналичные платежи от своих клиентов с помощью банковских карт и других платежных систем, таких как мобильные платежи (например, Mir Pay, Google Pay, Apple Pay) или QR-коды. Банк-эквайер предоставляет терминал, обрабатывает транзакции и переводит средства на счет юридического лица. Существует несколько основных видов эквайринга, каждый из которых предназначен для разных форматов бизнеса: торговый эквайринг — самый распространенный вид, при котором в торговой точке устанавливается POS-терминал для приема карт (подходит для онлайн-магазинов, кафе и других предприятий, где оплата происходит при личном контакте с клиентом); интернет-эквайринг — позволяет принимать платежи на сайтах, в мобильных приложениях и мессенджерах, для чего на сайт или в приложение интегрируется специальная платежная форма; мобильный эквайринг — используются мобильные mPOS-терминалы, которые являются удобным решением для курьерских служб, такси, выездной торговли; оплата по QR-коду — клиент сканирует QR-код с помощью своего мобильного устройства и подтверждает оплату в банковском приложении. Банк-эквайер взимает комиссию с каждой транзакции. Размер комиссии зависит от оборота юрлица, условий банка и типа используемой платёжной системы.

Прогнозирование объема транзакций по банковским продуктам позволяет подбирать более оптимальные условия обслуживания для клиента и принимать более точные решения по нему в различных ситуациях. Прогнозные данные могут использоваться для различных целей:

- 1. Оценка кредитоспособности.** Прогноз оборота дает более точное и своевременное представление о финансовом состоянии компании, чем годовая отчетность. Если банк видит, что оборот клиента стабильно падает, это может быть ранним и главным сигналом о грядущих проблемах с погашением кредитов и возникновению просрочек.
- 2. Повышение прибыльности и оптимизация ценообразования.** Для клиентов с прогнозируемым высоким и стабильным оборотом банк

может предложить более низкую комиссию по эквайрингу, чтобы удержать их и выиграть в конкурентной борьбе. И наоборот, для клиентов с нестабильным или падающим прогнозируемым оборотом можно сохранить стандартный или повышенный тариф. Анализ позволяет выявить компании с высоким потенциалом роста. Такие клиенты — главные кандидаты на углубление сотрудничества.

- 3. Своевременные предложения.** Если банк прогнозирует резкий рост оборота у клиента (например, сезонный всплеск у магазина подарков перед Новым годом), ему можно проактивно предложить краткосрочный кредит на пополнение оборотных средств.
- 4. Продажа дополнительных продуктов.** Растущему бизнесу могут потребоваться зарплатные проекты, корпоративные карты, новые расчетные счета или инвестиционные услуги. Прогноз позволяет сделать релевантное предложение в нужный момент.
- 5. Предотвращение оттока.** Если прогноз показывает стагнацию или падение оборота, менеджер может связаться с клиентом, чтобы выяснить причины и предложить решения (например, кредитные каникулы или реструктуризацию долга), тем самым повышая лояльность.

Кроме того, банк может использовать агрегированные данные по всем корпоративным клиентам для анализа общей картины состояния рынка. Банк может видеть, какие отрасли экономики растут, а какие находятся в упадке, основываясь на совокупном обороте своих клиентов в этих секторах. Эта информация может использоваться для корректировки кредитной политики и маркетинговой стратегии.

Динамика такого показателя как объем транзакций по эквайрингу во многом сопоставима со значениями в недавнем прошлом. Временная последовательность составленная из таких данных почти всегда имеет автокорреляцию — статистическую взаимосвязь между значениями ряда в разные моменты времени. **особенно в случае клиентов ММБ** Показатели активности клиента по другим банковским продуктам или в других банках, а также влияние внешних экономических факторов, как правило в высокой степени скоррелированы и отражены в динамике изменений моделируемого показателя, либо же наоборот — вообще не дают новой информации об изменениях во временной последовательности. Это означает, что качество авторегрессионного моделирования является вполне показательным достигаемого качества моделирования в данной задаче.

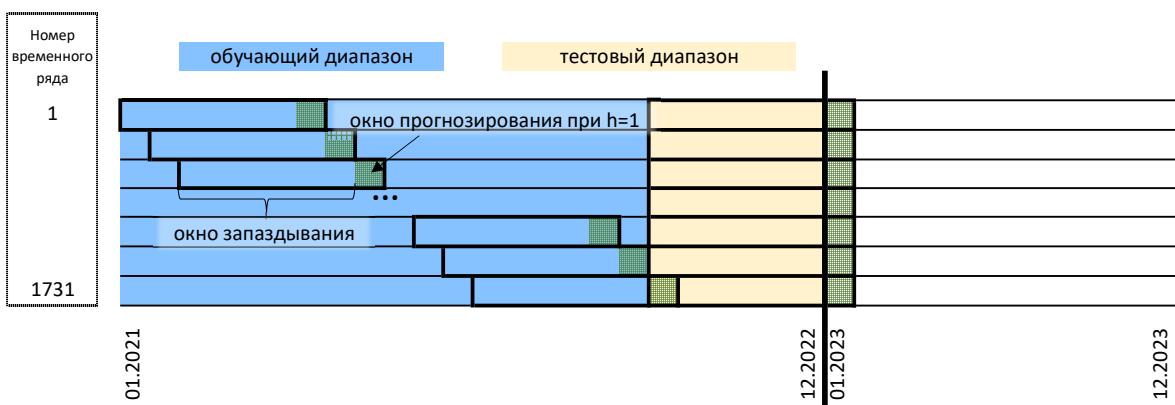


Рисунок 4.6 — Схема разбиения временных последовательностей на тренировочные и тестовые периоды.

Прогнозирование активности клиента по банковскому продукту (эквайринг) выполнялось на 1 и 3 следующих месяца.

Модель прогнозирования временных рядов строит прогноз с использованием значений только целевой переменной — объема оборота юридических лиц по эквайрингу (авторегрессия). Набор данных составлен из помесячных значений временных рядов за период с 01-01-2021 по 31-12-2023 для 2500 уникальных клиентов. Из набора данных были удалены временные ряды, состоящие только из нулевых значений, то есть клиенты без активности по продукту. В результате осталось 1731 временных рядов.

Схема разбиения временных рядов на тренировочный и тестовый отрезки изображена на рисунке.

Для оценки качества прогнозирования была выбрана целевая метрика МАЕ — данная метрика легко интерпретируема для данной задачи.

Был оценена производительность разработанной модели при разном размере лагового окна **3, 5, 7, 11**. Наблюдается линейное увеличение времени нечеткого вывода при линейном увеличении количества входов.

Лучшее качество прогнозирования наблюдалось при размере лагового окна — 6 точек.

Итоговые результаты качества прогнозирования по метрике МАЕ приведены в таблице 2.

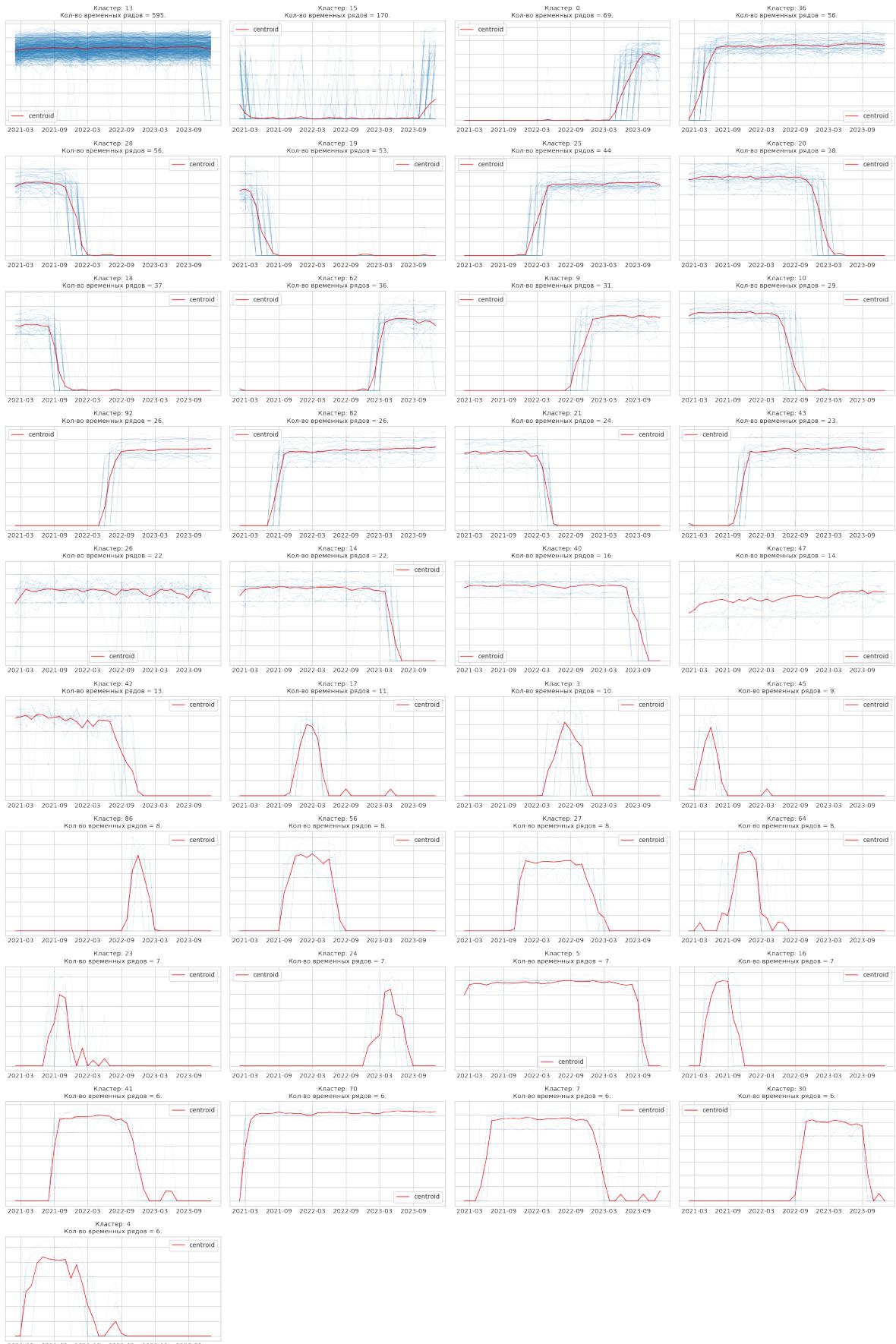


Рисунок 4.7 – Примеры кластеров временных рядов транзакционной активности различных клиентов по эквайрингу

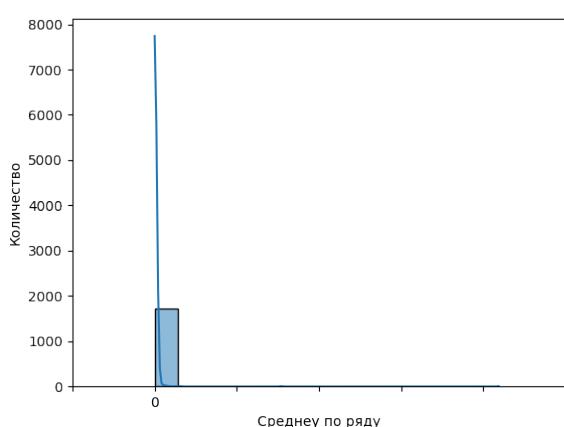


Рисунок 4.8 — Гистограмма значений оборота по эквайрингу.

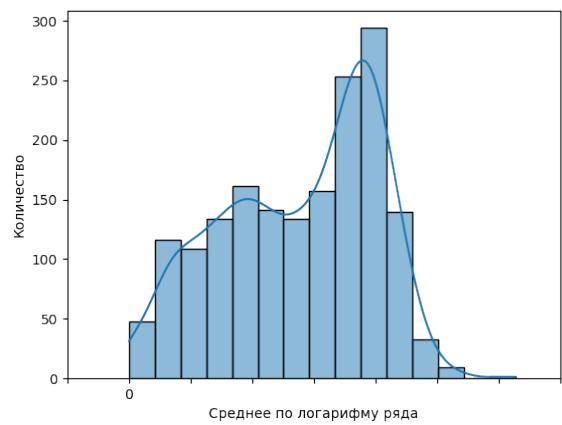


Рисунок 4.9 — Гистограмма логарифмированных значений оборота по эквайрингу.

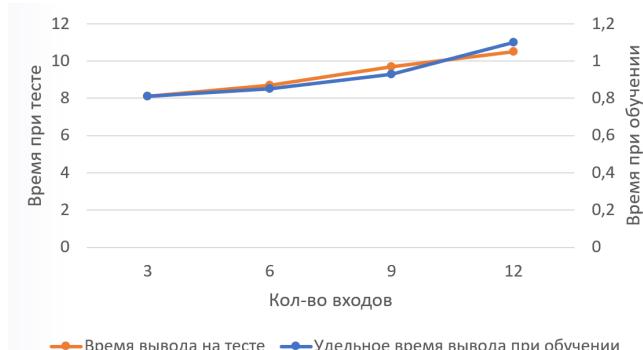


Рисунок 4.10 — Время нечеткого вывода для различных размеров лагового окна.

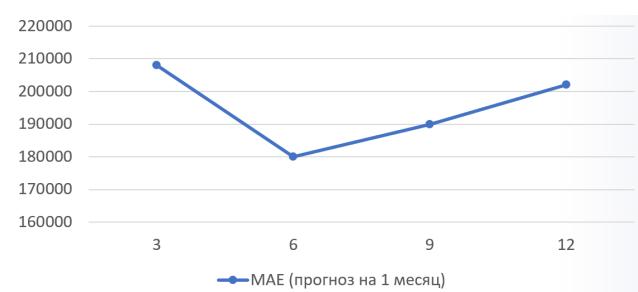


Рисунок 4.11 — Зависимость метрики качества прогнозирования от размера лагового окна.

Таблица 2 — Значения метрики MAE данной и альтернативных моделей.

Модель	Mean Absolute Error (MAE)	
	Среднее за 3 мес.	Сумма за 3 мес.
Наивная (среднее по ряду)	266792.479	389250.227
ARIMA	299906.518	437563.611
Градиентный бустинг	243800.592	355705.064
Нейросетевая модель	205415.312	299705.238
NSFLS+FTV	192929.211	260213.891

4.3 Выводы

1. Разработанная нечеткая модель прогнозирования временных рядов успешно применена для решения задач в экономике и финансах, в том числе для прогнозирования индекса TWSE и объема транзакций по банковскому продукту (эквайринг).
2. Проведённые эксперименты показали, что предложенная модель обеспечивает сопоставимое или лучшее качество прогнозирования по сравнению с классическими и нейросетевыми подходами, что подтверждается результатами по метрикам MAE и NDEI.
3. Модель демонстрирует устойчивую производительность при различных размерах лагового окна и обладает приемлемым временем вычисления, что позволяет использовать её для обработки больших массивов временных рядов.
4. Гибкость настройки гиперпараметров и возможность оптимизации структуры базы правил обеспечивают адаптацию модели к различным типам временных рядов и задачам прогнозирования.
5. Практическая применимость модели подтверждена на реальных данных, что свидетельствует о её эффективности для поддержки принятия решений в экономических и финансовых приложениях.

Заключение

Основные результаты работы заключаются в следующем. **Основные результаты диссертационного исследования**

1. Проведен анализ актуальных проблем методов нечеткого вывода с использованием несинглтонной фазификации. Приведено описание понятия нечеткого значения истинности. Обозначена актуальность оценки качества нечеткого моделирования при использовании нечеткого вывода логического типа с использованием несинглтонной фазификации на задаче прогнозирования временных рядов, описана адаптивная процедура оценки неопределенности и приведен адаптированный алгоритм метода роя частиц для подбора параметров базы правил.
2. Разработан и теоретически обоснован новый способ нечеткого логического вывода на основе нечеткого значения истинности, позволяющий оценить нечеткую степень совместимости входных нечетких множеств и н. м. антецедента правила по каждом входу независимо. Получен новый вид нечетких правил «Если истинно, то B_k ». Теоретически и экспериментально было показано, что сложность вычисления свертки НЗИ $O(|V| \cdot n)$ и сложность нечеткого вывода на основе НЗИ $O(|V| \cdot |Y|)$ полиномиально зависят от числа входов n .
3. Предложен эффективный параллельный алгоритмы вычисления свертки НЗИ для нечетких систем с несколькими входами, обеспечивающий сложность свертки НЗИ $O(|V| \times \log(n))$. Выполнена параллельная реализация нечеткого логического вывода с использованием технологии CUDA. Разработаны оптимизации по организации памяти и вычислений, обеспечивающие масштабируемость и эффективность при обработке больших массивов данных. Дефазификация по методу среднего максимума (МеOM) реализована на основе метода оптимизации Gradient-aware PSO.
4. Разработанный метод нечеткого вывода на основе НЗИ адаптирован к задаче мультиклассовой классификации, где при использовании метода дефазификации по **обеспечивается упрощение**.
5. На наборе данных Mackey-Glass для нечеткой модели на основе логического типа вывода и несинглтонной фазификации достигнута точность

прогнозирования временных рядов по метрике sMAPE — 8% при размере базе правил — 30 правил. Это значительно превосходит точность прогнозирования по этой метрике в том же эксперименте при использовании синглтонной фазификации ($\approx 40\%$) и имеет значительно меньшее количество правил при сопоставимой точности прогнозирования $\approx 10\%$ для вывода типа Мамдани с несинглтонной фазификацией (184 правил).

6. Программная реализация разработанной нечеткой модели была применена для решения задачи прогнозирования помесячных объемов транзакций безналичных платежей корпоративных клиентов банка. Достигнут прирост точности прогнозирования в 12%/5% по метрике RMAE при прогнозировании на один/три месяца в сравнении с лучшей моделью — многослойный перцентрон.

Перспективы дальнейших исследований

Поскольку при нечетком выводе частой является ситуация, когда срабатывает лишь небольшая часть правил, тогда время вывода можно значительно сократить, выполняя дефазификацию $\mu_{B'}(y)$ лишь для ограниченного набора самых релевантных входу правил. Это особенно актуально в практических задачах, в т. ч. с большим количеством входов, где требуется большое количество правил для покрытия большинства комбинаций термов. Предлагается для отбора релевантных правил использовать расстояние между нечеткими отношениями.

Словарь терминов

Мягкие вычисления : Методология использования неточных и математически строго не обоснованных методов и алгоритмов при решении задач, для которых не существует строгих подходов, позволяющих получить точный результат за приемлемое время.

Нечеткое значение истинности : Короткий текст, использующий все или почти все буквы алфавита

Список рисунков

1.1 Схема системы нечеткого вывода с использованием синглтонной и несинглтонной фазификации	14
1.2 Сравнение областей активации правил при переходе от синглтонной фазификации к несинглтонной и при увеличении ширины окрестности погрешности $\sigma_{A'}$	15
1.3 Сравнение формы функций принадлежности выходных нечетких множеств для подхода Мамдани	17
1.4 Сравнение формы функций принадлежности выходных нечетких множеств для логического подхода	18
1.5 Значения лингвистической переменной «истинность»	19
1.6 Пример вычисления нечеткого значения истинности	20
1.7 Иллюстрация случая истинного отношения высказываний	21
1.8 Иллюстрация случая ложного отношения высказываний	22
1.9 Иллюстрация случая абсолютно истинного отношения высказываний	23
1.10 Иллюстрация случая абсолютно ложного отношения высказываний .	24
1.11 Иллюстрация случая квαιстиинного отношения высказываний . . .	24
2.1 Сравнение классической схемы нечеткого вывода и схемы нечеткого вывода на основе НЗИ	37
2.2 Схема нейро-нечеткой системы с использованием дефазификации по методу среднего центра для S - и R -импликаций	38
2.3 Схема нейро-нечеткой системы с использованием дефазификации по методу центра тяжести	41
2.4 Пример нечетких множеств, удовлетворяющих условию $\mu_{B_k}(y_r) = 0$ для $y \neq r$	42
3.1 Блок схема процедуры нечеткого логического вывода на основе нечеткого значения истинности.	49
3.2 Карта разделяемой памяти одного CUDA-блока нечеткого логического вывода на основе нечеткого значения истинности при использовании дефазификации МеОМ.	51
3.3 График функции $\exp(-z^2)$	54
3.4 График функции $\varphi(v) = \sqrt{-\ln v}$	54

3.5	Случаи вырождения ф. п. НЗИ в близкий к асимптоте вид.	55
3.6	Иллюстрация работы алгоритма свертки НЗИ для нескольких входов для расширенной \tilde{T} -нормы при $T_1 = \min$ и $T_3 = \min$	60
3.7	Ситуация необходимости нахождения среднего максимума при дефазификации агрегированной функции принадлежности $\mu_{B'}(y)$. . .	62
3.8	Динамика движения точек популяции при работе алгоритма глобальной оптимизации Gradient-aware PSO.	63
3.9	Две иллюстрации: графика z-curve и apetrei.	65
4.1	Тренировочный и тестовый отрезок значений временного ряда TWSE и среднеквадратичное отклонение гауссовой ф. п. в каждой точке.	72
4.2	Влияние комбинации значений гиперпараметров на значение оптимизируемой метрики.	73
4.3	Влияние комбинации значений гиперпараметров на время построения базы правил.	73
4.4	Графики изменений максимального, среднего и лучшего по популяции значений оптимизируемой метрики.	74
4.5	Демонстрация предсказанных и фактических значений в наборе данных TWSE.	75
4.6	Схема разбиения временных последовательностей на тренировочные и тестовые периоды.	78
4.7	Примеры кластеров временных рядов транзакционной активности различных клиентов по эквайрингу	79
4.8	Гистограмма значений оборота по эквайрингу.	80
4.9	Гистограмма логарифмированных значений оборота по эквайрингу. . .	80
4.10	Время нечеткого вывода для различных размеров лагового окна. . .	80
4.11	Зависимость метрики качества прогнозирования от размера лагового окна.	80

Список таблиц

1	Сравнение показателей качества прогнозирования временного ряда из набора данных TWSE (TAIEX)	75
2	Значения метрики MAE данной и альтернативных моделей.	80

Приложение А

Текст программного кода Python-модуля нечеткого вывода с использованием технологии CUDA

```

cdef extern from "tsfuzzy.hpp":
    cdef enum class ImplType:
        LUKASIEWICZ,
        REICHENBACH,
        YAGER,
        KLEENE_DIENES
    cdef enum class DefuzMethod:
        COG,
        COG_SIMPLE,
        MEOM
    void fuzzy_fit_impl(
        const float *points_means, const float *points_stds, const unsigned
        → points_size, const unsigned point_dims,
        float *rules_means, float *rules_std, const unsigned rules_count,
        const ImplType impl_type, const DefuzMethod defuz_method,
        const unsigned ftv_discretization_size, const unsigned
        → defuz_pso_population_size, const unsigned defuz_pso_iter_count,
        const unsigned rules_pso_population_size, const unsigned
        → rules_pso_iterations_count
    )
    void fuzzy_predict_impl(
        const float *points_means, const float *points_stds, float *predictions, const
        → unsigned points_size, const unsigned point_dims,
        const float *rules_means, const float *rules_std, const unsigned rules_count,
        const ImplType impl_type, const DefuzMethod defuz_method,
        const unsigned ftv_discretization_size, const unsigned
        → defuz_pso_population_size, const unsigned defuz_pso_iter_count
    )

import numpy as np
cimport numpy as cnp
import pandas as pd

cnp.import_array()

```

Листинг А.1 Модуль-обертка на языке Cython

```

cdef class TsFuzzy:

    cdef ImplType impl_type
    cdef DefuzMethod defuz_method
    cdef int ftv_discretization_size
    cdef int defuz_pso_population_size
    cdef int defuz_pso_iters_count
    cdef int rules_pso_population_size
    cdef int rules_pso_iterations_count
    cdef cnp.ndarray rules_means
    cdef cnp.ndarray rules_stds

    def __init__(
        self,
        rules_count: int, y_dims: int,
        impl_type: Literal["lukasiewicz", "reichenbach", "yager", "kleene_dienes"] =
        ↪ "lukasiewicz",
        defuz_method: Literal["cog", "cog_simple", "meom"] = "meom",
        ftv_discretization_size: int = 11, defuz_pso_population_size: int = 30,
        ↪ defuz_pso_iters_count: int = 100,
        rules_pso_population_size: int = 50, rules_pso_iterations_count: int = 150
    ):
        impl_type_map = {"lukasiewicz": ImplType.LUKASIEWICZ, "reichenbach":
        ↪ ImplType.REICHENBACH, "yager": ImplType.YAGER, "kleene_dienes":
        ↪ ImplType.KLEENE_DIENES}
        defuz_method_map = {"cog": DefuzMethod.COG, "cog_simple":
        ↪ DefuzMethod.COG_SIMPLE, "meom": DefuzMethod.MEOM}
        try:
            self.impl_type = impl_type_map[impl_type]
        except KeyError:
            raise ValueError("Invalid impl_type; expected 'lukasiewicz' or 'goguen'.")
        try:
            self.defuz_method = defuz_method_map[defuz_method]
        except KeyError:
            raise ValueError("Invalid defuz_method; expected 'cog', 'cog_simple', or
            ↪ 'meom'.")
        self.ftv_discretization_size = ftv_discretization_size
        self.defuz_pso_population_size = defuz_pso_population_size
        self.defuz_pso_iters_count = defuz_pso_iters_count
        self.rules_pso_population_size = rules_pso_population_size
        self.rules_pso_iterations_count = rules_pso_iterations_count
        self.rules_means = np.empty((rules_count, y_dims), dtype=np.float32)
        self.rules_stds = np.empty((rules_count, y_dims), dtype=np.float32)

    def fit(self, points_means_, points_stds_):

```

```

    assert points_means_.shape == points_stds_.shape
    cdef cnp.ndarray[cnp.float32_t, ndim=2, mode='c'] points_means =
        → np.ascontiguousarray(points_means_, dtype=np.float32)
    cdef cnp.ndarray[cnp.float32_t, ndim=2, mode='c'] points_stds =
        → np.ascontiguousarray(points_stds_, dtype=np.float32)
    cdef int points_size = points_means.shape[0]
    cdef int points_dims = points_means.shape[1]
    fuzzy_fit_impl(
        <const float*>points_means.data, <const float*>points_stds.data,
        → points_size, points_dims,
        <float*>self.rules_means.data, <float*>self.rules_stds.data,
        → self.rules_means.shape[0],
        self.impl_type, self.defuz_method, self.ftv_discretization_size,
        → self.defuz_pso_population_size, self.defuz_pso_iters_count,
        self.rules_pso_population_size, self.rules_pso_iterations_count,
    )

def predict(self, points_means_, points_stds_):
    assert points_means_.shape == points_stds_.shape
    cdef cnp.ndarray[cnp.float32_t, ndim=2, mode='c'] points_means =
        → np.ascontiguousarray(points_means_, dtype=np.float32)
    cdef cnp.ndarray[cnp.float32_t, ndim=2, mode='c'] points_stds =
        → np.ascontiguousarray(points_stds_, dtype=np.float32)
    cdef int points_size = points_means.shape[0]
    cdef int points_dims = points_means.shape[1]
    cdef cnp.ndarray[cnp.float32_t, ndim=1, mode='c'] predictions =
        → np.empty(points_size, dtype=np.float32)
    fuzzy_predict_impl(
        <const float*>points_means.data, <const float*>points_stds.data,
        → <float*>predictions.data, points_size, points_dims,
        <const float*>self.rules_means.data, <const float*>self.rules_stds.data,
        → self.rules_means.shape[0],
        self.impl_type, self.defuz_method, self.ftv_discretization_size,
        → self.defuz_pso_population_size, self.defuz_pso_iters_count,
    )
    return predictions

def __getstate__(self):
    return {
        "impl_type": int(self.impl_type),
        "defuz_method": int(self.defuz_method),
        "ftv_discretization_size": self.ftv_discretization_size,
        "defuz_pso_population_size": self.defuz_pso_population_size,
        "defuz_pso_iters_count": self.defuz_pso_iters_count,
        "rules_means": np.asarray(self.rules_means),
    }

```

Листинг А.2 Реализация нечеткого вывода на языке CUDA C++ с использованием библиотек Kokkos и ArborX

```

    "rules_stds": np.asarray(self.rules_stds)
}

def __setstate__(self, state):
    from tsfuzzy import ImplType, DefuzMethod
    self.impl_type = ImplType(state["impl_type"])
    self.defuz_method = DefuzMethod(state["defuz_method"])
    self.ftv_discretization_size = state["ftv_discretization_size"]
    self.defuz_pso_population_size = state["defuz_pso_population_size"]
    self.defuz_pso_iters_count = state["defuz_pso_iters_count"]
    self.rules_means = np.asarray(state["rules_means"], dtype=np.float32)
    self.rules_stds = np.asarray(state["rules_stds"], dtype=np.float32)

@staticmethod
def fuzzify(
    y,
    wma = None, sample_std = 1.0
):
    # Compute the difference series d with length len(y)-1.
    d = pd.Series(y[1:] - y[:-1]) / np.sqrt(2)
    y = pd.Series(y)
    # Calculate deviations from d using ewm.
    computed_deviations = sample_std * d.ewm(alpha=0.7, adjust=False,
        min_periods=1).std()
    # Prepend the first computed deviation value so that deviations has the same
    # length as y.
    deviations = pd.concat([pd.Series([computed_deviations.iloc[0]]),
        computed_deviations]).reset_index(drop=True)
    # Compute centers. Using min_periods=1 ensures that initial values are filled
    # with the first computed mean.
    if wma is not None:
        centers = y.ewm(alpha=1 - wma, adjust=True, min_periods=1).mean()
    else:
        centers = y
    return centers, deviations

```

Листинг реализации нечеткого вывода на языке CUDA C++ с использованием библиотек Kokkos и ArborX ??.

```

template <typename RealT>
struct metrics

```

```

{
    std::size_t count = 0;
    RealT y_true_mean = 0;
    RealT y_true2_mean = 0;
    RealT mae = 0;
    RealT mape = 0;
    RealT mse = 0;
public:
    KOKKOS_INLINE_FUNCTION
    void operator()(RealT y_true, RealT y_pred)
    {
        y_true_mean = (y_true + count * y_true_mean) / (count+1);
        y_true2_mean = (y_true * y_true + count * y_true2_mean) / (count+1);
        mae = (Kokkos::abs(y_true - y_pred) + count * mae) / (count+1);
        mape = (Kokkos::abs(y_true - y_pred) / y_true + count * mape) / (count+1);
        mse = (Kokkos::pow(y_true - y_pred, 2) + count * mse) / (count+1);
        ++count;
    }

    KOKKOS_INLINE_FUNCTION
    RealT rmse() const { return Kokkos::sqrt(mse); }
    KOKKOS_INLINE_FUNCTION
    RealT ndei() const { return rmse() / std(); }
    KOKKOS_INLINE_FUNCTION
    RealT er2() const { return mse / var(); }

private:
    KOKKOS_INLINE_FUNCTION
    RealT var() const { return y_true2_mean - Kokkos::pow(y_true_mean, 2); }
    KOKKOS_INLINE_FUNCTION
    RealT std() const { return Kokkos::sqrt(var()); }
};

template <unsigned DIM, typename SomeMemoryTraits>
void fuzzy_infer(const Kokkos::View<ArborX::Point<DIM>*, Kokkos::OpenMP::array_layout,
    Kokkos::CudaSpace> &points_means_device,
    const Kokkos::View<ArborX::Point<DIM>*, Kokkos::OpenMP::array_layout,
    Kokkos::CudaSpace> &points_stds_device,
    Kokkos::View<float**, Kokkos::CudaSpace, SomeMemoryTraits>
    &predictions_population_device,
    const Kokkos::View<ArborX::Point<DIM>**, Kokkos::CudaSpace,
    SomeMemoryTraits>& rules_means_population_device,
    const Kokkos::View<ArborX::Point<DIM>**, Kokkos::CudaSpace,
    SomeMemoryTraits>& rules_stds_population_device,
    
```

```

    const Kokkos::Random_XorShift64_Pool<Kokkos::Cuda> &warp_random_pool,
    const Kokkos::Random_XorShift64_Pool<Kokkos::Cuda>
        ↪ &thread_random_pool,
    const ImplType impl_type,
    const DefuzMethod defuz_method,
    const unsigned ftv_discretization_size,
    const unsigned defuz_pso_population_size,
    const unsigned defuz_pso_iterations_count,
    const unsigned block_replication_count = 1) {

const bool debug = (bool)get_env<int>("DEBUG", 0);
const int DEBUG_FTV_RULE_IDX = get_env<int>("DEBUG_FTV_RULE_IDX", 0);
const int DEBUG_V_STEP = get_env<int>("DEBUG_V_STEP", 50);
const int DEBUG_SAMPLE_IDX = get_env<int>("DEBUG_SAMPLE_IDX", 10000);

const unsigned rules_population_size = rules_means_population_device.extent(0);
const unsigned rules_count = rules_means_population_device.extent(1);
const unsigned warp_count = (44000 / (
    4 * (DIM*rules_count*2 + rules_count*ftv_discretization_size + rules_count*(DIM-1)
    ↪ + rules_count*3 + defuz_pso_population_size*5)
    + 1 * (rules_count*(DIM-1))
    + 1000
));
assert(warp_count >= 1);

// dprintf("rules_population_size = %u, block_replication_count = %u, warp_count =
// → %u, rules_count = %u, ftv_discretization_size = %u, defuz_pso_population_size =
// → %u\n", rules_population_size, block_replication_count, warp_count, rules_count,
// → ftv_discretization_size, defuz_pso_population_size);
Kokkos::TeamPolicy<Kokkos::Cuda> team_policy(rules_population_size *
    ↪ block_replication_count, 32 * warp_count);
team_policy.set_scratch_size(0, Kokkos::PerTeam(44000));

auto fuzzy_infer_block_lambda = [=] __device__ (const
    ↪ Kokkos::TeamPolicy<Kokkos::Cuda>::member_type &team) {
    // if (team.league_rank() == 0 && team.team_rank() == 0) {
    //     dprintf("gridDim = [%d %d %d] blockDim = [%d %d %d]\n",
    //         gridDim.x, gridDim.y, gridDim.z, blockDim.x, blockDim.y, blockDim.z);
    // }

    const int pso_idx = team.league_rank() / block_replication_count;
    const int warp_idx = team.team_rank() / 32;
    const int lane_idx = team.team_rank() % 32;
    auto warp_gen = warp_random_pool.get_state(team.league_rank() * warp_count +
        ↪ warp_idx);
}

```

```

auto thread_gen = thread_random_pool.get_state(team.league_rank() *
→ team.team_size() + team.team_rank());

ScratchView<ArborX::Point<DIM>*> rules_means(team.team_shmem(), rules_count);
ScratchView<ArborX::Point<DIM>*> rules_stds(team.team_shmem(), rules_count);
// if (team.league_rank() == 0 && team.team_rank() == 0)
//   printf("rules_count = %d, rules_means.extent(0) = %d, rules_stds.extent(0) =
→ %d\n", (int)rules_count, (int)rules_means.extent(0),
→ (int)rules_stds.extent(0));
Kokkos::parallel_for(Kokkos::TeamThreadRange(team, rules_count), [=] __device__
→ (int j) {
    rules_means(j) = rules_means_population_device(pso_idx, j);
    rules_stds(j) = rules_stds_population_device(pso_idx, j);
});
__syncthreads();

ScratchView<ArborX::Point<DIM>*> points_means(team.team_shmem(), warp_count),
→ points_stds(team.team_shmem(), warp_count);

// dprintf("league_rank: %d, team_rank: %d, pso_idx: %d, warp_idx: %d, lane_idx:
→ %d, block_replication_count: %u, warp_count: %u\n",
//       team.league_rank(), team.team_rank(), pso_idx, warp_idx, lane_idx,
→ block_replication_count, warp_count);
auto team_shmem_backup = team.team_shmem();
for (int sample_idx = (team.league_rank() % block_replication_count) * warp_count +
→ warp_idx;
     sample_idx < points_means_device.extent(0);
     sample_idx += block_replication_count * warp_count)
{
    const_cast<ScratchSpace&>(team.team_shmem()) = team_shmem_backup;
    for (int i = lane_idx; i < DIM; i += 32) {
        points_means(warp_idx)[i] = points_means_device(sample_idx)[i];
        points_stds(warp_idx)[i] = points_stds_device(sample_idx)[i];
    }
    const auto point_means = Kokkos::subview(points_means, warp_idx);
    const auto point_stds = Kokkos::subview(points_stds, warp_idx);
    // if (lane_idx == 0) {
    //   printf("sample_idx = %d, warp_idx = %d, point = {[%f %f], [%f %f], [%f %f],
→ [%f %f], [%f %f]}\n",
    //         sample_idx,
    //         warp_idx,
    //         point_means()[0], point_stds()[0],
    //         point_means()[1], point_stds()[1],
    //         point_means()[2], point_stds()[2],
    //         point_means()[3], point_stds()[3],

```

```

//  point_means()[4], point_stds()[4]
//  // point_means()[5], point_stds()[5]
//  );
// }
_syncwarp();

using Kokkos::abs;
using Kokkos::exp;
using Kokkos::pow;
using Kokkos::sqrt;
using Kokkos::log;
using Kokkos::round;

ScratchView<float***> rules_reduced_ftv(team.team_shmem(), warp_count,
→ rules_means.extent(0), ftv_discretization_size);
// auto reduce_ftv_team_shmem_backup = team.team_shmem();

{
    ScratchView<unsigned char***> rules_ftv_core_step(team.team_shmem(),
→ warp_count, rules_means.extent(0), DIM-1);
    ScratchView<float***> rules_max_ftv(team.team_shmem(), warp_count,
→ rules_means.extent(0), DIM-1);
    for (int rule_idx = lane_idx; rule_idx < rules_means.extent(0); rule_idx += 32)
→ {
        auto reduced_ftv = Kokkos::subview(rules_reduced_ftv, warp_idx, rule_idx,
→ Kokkos::ALL); // TODO // FIXME
        auto ftv_core_step = Kokkos::subview(rules_ftv_core_step, warp_idx, rule_idx,
→ Kokkos::ALL);
        auto max_ftv = Kokkos::subview(rules_max_ftv, warp_idx, rule_idx,
→ Kokkos::ALL);

        const auto &rule_means = rules_means(rule_idx);
        const auto &rule_stds = rules_stds(rule_idx);

        for (int dim = 0; dim < DIM-1; ++dim) {
            float a = rule_means[dim], b = rule_stds[dim];
            float c = point_means()[dim];
            ftv_core_step(dim) = min(max((int) round(exp(-pow((a-c)/b, 2)) *
→ (ftv_discretization_size-1)), 0), ftv_discretization_size-1);
            // if (sample_idx == DEBUG_SAMPLE_IDX && rule_idx == DEBUG_FTV_RULE_IDX)
            //     dprintf("rule %d, dim = %d, [%f %f %f %f], ftv_core_step = %d\n",
→ rule_idx, dim, a, b, c, point_stds()[dim], (int)ftv_core_step(dim));
            max_ftv(dim) = 0.f;
        }
    }
}

```

```

// Here task is to provide discretization shape most similar to the each of
→ FTV shape cases
for (int v_step = ftv_discretization_size; v_step > 0; ) {
    --v_step;
    float v = (float)v_step / (ftv_discretization_size-1);
    float v_max_ftv = 0.f, v_max_ftv_dim;
    bool is_growth_present = false;
    for (int dim = 0; dim < DIM-1; ++dim) {
        const float a = rule_means[dim], b = rule_stds[dim];
        const float c = point_means()[dim], d = point_stds()[dim];
        float ftv = (v_step == ftv_core_step(dim))
            ? 1.f
            : Kokkos::max(exp(-pow((a - c) + b * sqrt(-log(v))) / d, 2)),
            → exp(-pow((a - c) - b * sqrt(-log(v))) / d, 2));
    }

    // if (rule_idx == DEBUG_FTV_RULE_IDX && v_step == DEBUG_V_STEP)
    //   printf("rule %d, v_step = %d, v = %f, dim = %d, ftv = %f\n",
    → rule_idx, v_step, v, dim, ftv);
    if (ftv > max_ftv(dim)) {
        max_ftv(dim) = ftv;
        is_growth_present = true;
    }
    if (ftv > v_max_ftv) {
        v_max_ftv = ftv;
        v_max_ftv_dim = dim;
    }
}
float ftv_reduced = 1.f;
for (int dim = 0; dim < DIM-1; ++dim) {
    if (!is_growth_present && (dim == v_max_ftv_dim)) {
        ftv_reduced = Kokkos::min(ftv_reduced, v_max_ftv);
    } else {
        ftv_reduced = Kokkos::min(ftv_reduced, max_ftv(dim));
    }
}
if (rule_idx == DEBUG_FTV_RULE_IDX && sample_idx == DEBUG_SAMPLE_IDX) {
    dprintf("rule %d, v_step = %d, v = %f, max_ftv = { %f %f %f %f %f },
    → v_max_ftv = %f, v_max_ftv_dim = %d, ftv_reduced = %f\n",
    rule_idx, v_step, v, max_ftv(0), max_ftv(1), max_ftv(2), max_ftv(3),
    → max_ftv(4),
    v_max_ftv, (int)v_max_ftv_dim, ftv_reduced);
}
reduced_ftv(v_step) = ftv_reduced;
}
}

```

```

}

// const_cast<ScratchSpace>(team.team_shmem()) = reduce_ftv_team_shmem_backup;
// __syncthreads();

ScratchView<float*> y_means(team.team_shmem(), warp_count);
ScratchView<float*> y_stds(team.team_shmem(), warp_count);

// ScratchView<float*> y_mean_numer(team.team_shmem(), warp_count);
// ScratchView<float*> y_std_numer(team.team_shmem(), warp_count);
// ScratchView<float*> y_denom(team.team_shmem(), warp_count);
// y_mean_numer(warp_idx) = 0;
// y_std_numer(warp_idx) = 0;
// y_denom(warp_idx) = 0;
// __syncthreads();
// Kokkos::parallel_reduce(
//   Kokkos::TeamThreadMDRange(team, warp_count, rules_means.extent(0)),
float y_mean_numer = 0.f, y_std_numer = 0.f, y_denom = 0.f;
for (int rule_idx = lane_idx; rule_idx < rules_means.extent(0); rule_idx += 32) {
    const auto &rule_stds = rules_stds(rule_idx);
    const auto &rule_means = rules_means(rule_idx);
    const auto reduced_ftv = Kokkos::subview(rules_reduced_ftv, warp_idx, rule_idx,
→ Kokkos::ALL);

    float v_numerator = 0.f, v_denominator = 0.f;
    const float h = 1.f / (ftv_discretization_size-1);
    for (int j = 0; j < ftv_discretization_size; j += 3) {
        const float v1 = j / (ftv_discretization_size-1);
        const float v2 = (j+1) / (ftv_discretization_size-1);
        const float v3 = (j+2) / (ftv_discretization_size-1);
        const float v4 = (j+3) / (ftv_discretization_size-1);

        const float f1 = reduced_ftv(j);
        const float f2 = reduced_ftv(j + 1);
        const float f3 = reduced_ftv(j + 2);
        const float f4 = reduced_ftv(j + 3);

        // Center of gravity v value using Simpson's rule
        v_numerator += 3*h/8 * (v1*f1 + 3*v2*f2 + 3*v3*f3 + v4*f4);
        v_denominator += 3*h/8 * (f1 + 3*f2 + 3*f3 + f4);
    }
    const float v_cog = v_numerator / v_denominator + 0.001;

    // for (int j = 0; j < ftv_discretization_size; ++j) printf("%d %d %f\n", i, j,
→ reduced_ftv(j));
}

```

```

// if (sample_idx == DEBUG_SAMPLE_IDX) printf("rule %d: v_cog = %f v_numer = %f
→   v_denom = %f\n", i, v_cog, v_numer, v_denom);
y_mean_numer += rule_means[DIM-1] * v_cog;
y_std_numer += rule_stds[DIM-1] * v_cog;
y_denom += v_cog;
}

::cuda_intra_warp_reduction(y_mean_numer, Kokkos::Sum<float,
→   ScratchSpace>(y_mean_numer));
::cuda_intra_warp_reduction(y_std_numer, Kokkos::Sum<float,
→   ScratchSpace>(y_std_numer));
::cuda_intra_warp_reduction(y_denom, Kokkos::Sum<float, ScratchSpace>(y_denom));

if (lane_idx == 0) y_means(warp_idx) = (y_denom == 0.f) ? point_means()[DIM-2] :
→   y_mean_numer / y_denom;
if (lane_idx == 0) y_stds(warp_idx) = (y_denom == 0.f) ? point_stds()[DIM-2] :
→   y_std_numer / y_denom;
__syncwarp();

ScratchView<float*> out_firelevels(team.team_shmem(), warp_count);
ScratchView<float*> out_f1s(team.team_shmem(), warp_count);
ScratchView<float*> out_f2s(team.team_shmem(), warp_count);
auto compute_out_firelevel = [=] __device__ (float y, float &out_f1, float
→   &out_f2) {
    auto get_ftv_rule_cross = [=](const int rule_idx, const float gauss_y0, float
→   &f0_max_value, float &f0_max_v) {
        const auto reduced_ftv = Kokkos::subview(rules_reduced_ftv, warp_idx,
→   rule_idx, Kokkos::ALL);

        for (int v_step = 0; v_step < ftv_discretization_size-1; ++v_step) {
            const float v1 = (float)v_step / (ftv_discretization_size-1);
            const float v2 = (float)(v_step+1) / (ftv_discretization_size-1);
            const float f0_cp1 = reduced_ftv(v_step);
            const float f0_cp2 = reduced_ftv(v_step+1);
            float f0_impl1 = 0.f, f0_impl2 = 0.f;

            switch (impl_type) {
                case ImplType::LUKASIEWICZ:
                    f0_impl1 = Kokkos::min(1.f, 1.f - v1 + gauss_y0);
                    f0_impl2 = Kokkos::min(1.f, 1.f - v2 + gauss_y0);
                    break;
                case ImplType::REICHENBACH:
                    f0_impl1 = Kokkos::min(1.f, 1.f - v1 + v1 * gauss_y0);
                    f0_impl2 = Kokkos::min(1.f, 1.f - v2 + v2 * gauss_y0);
                    break;
                case ImplType::KLEENE_DIENES:

```

```

f0_impl1 = Kokkos::max(1.f - v1, gauss_y0);
f0_impl2 = Kokkos::max(1.f - v2, gauss_y0);
break;
}

if ((f0_impl1 - f0_cp1) * (f0_impl2 - f0_cp2) <= 0.f) {
    // const float v = v1 + (v2 - v1) * (f0_impl1 - f0_cp1) / ((f0_impl2 -
    // f0_cp2) - (f0_impl1 - f0_cp1));
    const float v = (f0_cp1 - f0_cp2 - f0_impl1 + f0_impl2 == 0.f)
        ? (v1 + v2) / 2
        : (f0_cp1*v2 - f0_cp2*v1 - f0_impl1*v2 + f0_impl2*v1) / (f0_cp1 -
        // f0_cp2 - f0_impl1 + f0_impl2);
    float f0 = 0.f;
    switch (impl_type) {
        case ImplType::LUKASIEWICZ:
            f0 = Kokkos::min(1.f, 1.f - v + gauss_y0);
            break;
        case ImplType::REICHENBACH:
            f0 = Kokkos::min(1.f, 1.f - v + v * gauss_y0);
            break;
        case ImplType::KLEENE_DIENES:
            f0 = Kokkos::max(1.f - v, gauss_y0);
            break;
    }
    // if (rule_idx == DEBUG_FTV_RULE_IDX && sample_idx == DEBUG_SAMPLE_IDX)
    //     printf("[rule %d] v_step = %d, v = %f, f0 = %f, f0_cp1 = %f, f0_cp2
    //     = %f, f0_impl1 = %f, f0_impl2 = %f\n",
    //     rule_idx, v_step, v, f0, f0_cp1, f0_cp2, f0_impl1, f0_impl2);
    if (f0 > f0_max_value) {
        f0_max_value = f0;
        f0_max_v = v;
    }
}
}
};

Kokkos::ValLocScalar<float, int> min_out_firelevel = {100, -1};
for (int rule_idx = lane_idx; rule_idx < rules_means.extent(0); rule_idx += 32)
{
    const auto &rule_means = rules_means(rule_idx);
    const auto &rule_stds = rules_stds(rule_idx);
    const float out_mu = rule_means[DIM-1], out_sigma = rule_stds[DIM-1];
    const float gauss_y0 = exp(-pow((y - out_mu) / (2*out_sigma), 2));

    float f0_max_value = 0.f, f0_max_v = -1;
    get_ftv_rule_cross(rule_idx, gauss_y0, f0_max_value, f0_max_v);
}

```

```

    if (f0_max_value < min_out_firelevel.val) {
        min_out_firelevel.val = f0_max_value;
        min_out_firelevel.loc = rule_idx;
    }
}

auto min_loc_reducer = ::TeamMinLoc<float, int>(min_out_firelevel);
::cuda_intra_warp_reduction(min_out_firelevel, min_loc_reducer);
if (lane_idx == 0) {
    out_firelevels(warp_idx) = min_out_firelevel.val;
    const int rule_idx = min_out_firelevel.loc;
    const auto &rule_means = rules_means(rule_idx);
    const auto &rule_stds = rules_stds(rule_idx);
    const float out_mu = rule_means[DIM-1], out_sigma = rule_stds[DIM-1];
    const float gauss_y0 = exp(-pow((y - out_mu) / (2*out_sigma), 2));
    float f0_max_value, f0_max_v;
    get_ftv_rule_cross(rule_idx, gauss_y0, f0_max_value, f0_max_v);
    out_f1s(warp_idx) = -(y - out_mu) / (2*out_sigma*out_sigma *
        (f0_max_v+0.001));
    out_f2s(warp_idx) = (out_mu*out_mu - 2*out_mu*y - 2*out_sigma*out_sigma +
        y*y) / (4*pow(out_sigma, 4)*(f0_max_v+0.001));
}
--syncwarp();

out_f1 = out_f1s(warp_idx);
out_f2 = out_f2s(warp_idx);
return out_firelevels(warp_idx);
};

float y, out_f1, out_f2;
float o = -1.f;
switch (defuz_method) {
    case DefuzMethod::COG_SIMPLE: {
        float y_cog_numerator = 0.f, y_cog_denom = 0.f;
        for (int k = 0; k < rules_means.extent(0); ++k) {
            const auto &y_k_center = rules_means(k)[DIM-1];
            const float out_firelevel = compute_out_firelevel(y_k_center, out_f1,
                out_f2);
            y_cog_numerator += y_k_center * out_firelevel;
            y_cog_denom += out_firelevel;
        }
        y = (y_cog_denom == 0.f) ? y_means(warp_idx) : y_cog_numerator / y_cog_denom;
// if (team.team_rank() == 0) printf("[sample_id = %d] y_mean=%f\n", sample_idx,
        // y_mean);
        break;
    }
}

```

```

}

case DefuzMethod::COG: {
    const unsigned n_it = defuz_pso_iterations_count;
    const float lr = 0.1f;
    const unsigned population_size = defuz_pso_population_size;
    const float y_lower_bound = y_means(warp_idx) - 10.f * y_stds(warp_idx),
    ↳ y_upper_bound = y_means(warp_idx) + 10.f * y_stds(warp_idx);
    ScratchView<float*> v_population(team.team_shmem(), population_size);
    ScratchView<float**> y_populations(team.team_shmem(), n_it, population_size);
    ScratchView<float**> out_firelevel_populations(team.team_shmem(), n_it-1,
    ↳ population_size);
    Kokkos::parallel_for(Kokkos::TeamThreadRange(team, population_size), [=]
    ↳ __device__ (int j) {
        auto &mutable_gen =
            ↳ const_cast<std::remove_const_t<decltype(thread_gen)>&>(thread_gen);
        v_population(j) = 0.f;
        y_populations(0, j) = mutable_gen.frand(y_lower_bound, y_upper_bound);
    });
    __syncwarp();
    for (int y_it = 1; y_it < n_it; ++y_it) {
        const float d = exp(-y_it*lr);
        for (int j = 0; j < population_size; ++j) {
            const float out_firelevel = compute_out_firelevel(y_populations(y_it-1,
            ↳ j), out_f1, out_f2);
            out_firelevel_populations(y_it-1, j) = out_firelevel;
            v_population(j) = (1.f-lr) * v_population(j) +
                (out_firelevel*(1.f-d)*(-out_f1 / (out_f2+1e-6f)) +
                ↳ (1.f-out_firelevel)*d*(y_means(warp_idx)-y_populations(y_it-1,
                ↳ j)));
            if (sample_idx == DEBUG_SAMPLE_IDX && lane_idx == 0)
                dprintf("y_it = %2d j=%d, y = %f, out_firelevel = %f, out_f1 = %f,
                ↳ out_f2 = %f, v = %f, d = %f\n",
                y_it, j, y_populations(y_it-1, j), out_firelevel_populations(y_it-1,
                ↳ j), out_f1, out_f2, v_population(j), d);
            y_populations(y_it, j) = y_populations(y_it-1, j) + lr * v_population(j);
        }
    }
    __syncthreads();
    ScratchView<float*> y_values(y_populations.data(), (n_it-1) *
    ↳ population_size);
    ScratchView<float*> out_firelevel_values(out_firelevel_populations.data(),
    ↳ (n_it-1) * population_size);
    Kokkos::Experimental::sort_by_key_team(team, y_values, out_firelevel_values);
    ScratchView<float> y_cog_numer(team.team_shmem(), 1),
    ↳ y_cog_denom(team.team_shmem(), 1);
}

```

```

y_cog_numer() = 0.f;
y_cog_denom() = 0.f;
// Kokkos::parallel_reduce(
//   Kokkos::TeamThreadRange(team, (n_it-1)*population_size/2),
//   [=] __device__ (int i, auto &y_numerator, auto &y_denom) {
__syncthreads();
// if (team.team_rank() == 0)
    float y_numerator = 0.f, y_denom = 0.f;
for (int i = 0; i < (n_it-1)*population_size; ++i) {
    const float h = (y_values(i+1) - y_values(i)) / 2.f;
    if (h == 0.f) continue;
    const float y1 = y_values(i) + h;
    const float f0 = out_firelevel_values(i);
    const float f1 = compute_out_firelevel(y1, out_f1, out_f2);
    const float f2 = out_firelevel_values(i+1);
    const float g0 = y_values(i) * f0;
    const float g1 = y1 * f1;
    const float g2 = y_values(i+1) * f2;
    y_numerator += h/3.f * (g0 + 4.f*g1 + g2);
    y_denom += h/3.f * (f0 + 4.f*f1 + f2);
    // y_cog_numer() += y_numerator;
    // y_cog_denom() += y_denom;
}
__syncthreads();
y_cog_numer() = y_numerator;
y_cog_denom() = y_denom;
__syncthreads();
// printf("y_cog_numer = %f, y_cog_denom = %f\n", y_cog_numer(),
// → y_cog_denom());
// }, y_cog_numer(), y_cog_denom());
y = (y_cog_denom() == 0.f) ? y_means(warp_idx) : y_cog_numer() /
→ y_cog_denom();
break;
}

case DefuzMethod::MEOM: {
    const float lr = 0.01f;
    const float lr_p = 0.03f, lr_g = 0.03f;
    const float y_lower_bound = y_means(warp_idx) - 10.f * y_stds(warp_idx),
    → y_upper_bound = y_means(warp_idx) + 10.f * y_stds(warp_idx);
    const unsigned population_size = defuz_pso_population_size;
    ScratchView<float**> y_population(team.team_shmem(), warp_count,
    → population_size);
    ScratchView<float**> v_population(team.team_shmem(), warp_count,
    → population_size);
}

```

```

ScratchView<float**> out_firelevel_population(team.team_shmem(), warp_count,
→ population_size);
ScratchView<float**> best_y_population(team.team_shmem(), warp_count,
→ population_size);
ScratchView<float**> best_out_firelevel_population(team.team_shmem(),
→ warp_count, population_size);
ScratchView<float*> best_y(team.team_shmem(), warp_count);
ScratchView<float*> best_out_firelevel(team.team_shmem(), warp_count);
for (int j = lane_idx; j < population_size; j += 32) {
    auto &mutable_gen =
        const_cast<std::remove_const_t<decltype(thread_gen)>&>(thread_gen);
    y_population(warp_idx, j) = mutable_gen.frand(y_lower_bound,
→ y_upper_bound);
    const auto delta = y_upper_bound - y_lower_bound;
    v_population(warp_idx, j) = 0.5f*mutable_gen.frand(-delta, delta);
    best_out_firelevel_population(warp_idx, j) = -1.f;
}
best_y(warp_idx) = y_means(warp_idx);
best_out_firelevel(warp_idx) = 0.f;
__syncwarp();
// y = y_mean;
y = 0.643016;
float out_firelevel;
float v = 0.f;
for (int y_it = 0; y_it < defuz_pso_iterations_count; ++y_it) {
    const float d = exp(-y_it*lr_g);
    for (int j = 0; j < population_size; ++j) {
        out_firelevel_population(warp_idx, j) = out_firelevel =
            compute_out_firelevel(y_population(warp_idx, j), out_f1, out_f2);
        if (out_firelevel > best_out_firelevel_population(warp_idx, j)) {
            best_out_firelevel_population(warp_idx, j) = out_firelevel;
            best_y_population(warp_idx, j) = y_population(warp_idx, j);
            if (out_firelevel > best_out_firelevel(warp_idx)) {
                best_out_firelevel(warp_idx) = out_firelevel;
                best_y(warp_idx) = y_population(warp_idx, j);
            }
        }
    }
    for (int j = 0; j < population_size; ++j) {
        const float y = y_population(warp_idx, j);
        const float out_firelevel = out_firelevel_population(warp_idx, j);
        // v = 0.8f * v + ((out_firelevel)*(1.f-d)*(out_f1 / (abs(out_f2) +
→ 1e-6f)) + (1.f - out_firelevel)*(d)*(y_mean - y));
        const float rp = warp_gen.frand(0.f, 1.f), rg = warp_gen.frand(0.f, 1.f);
        // const float rd = gen.frand(0.f, 1.f), rm = gen.frand(0.f, 1.f);
    }
}

```

```

    const float rd = 1.f - rp, rm = 1.f - rg;
    v_population(warp_idx, j) = 0.8f * v_population(warp_idx, j) + lr_p *
    ↪ ((rd)*(out_f1 / (abs(out_f2) + 1e-6f)) +
    ↪ rp*(best_y_population(warp_idx, j) - y)) +
    lr_g * ((1.f-best_out_firelevel(warp_idx))*(rm)*d*(y_means(warp_idx) -
    ↪ y) + best_out_firelevel(warp_idx)*rg*(best_y(warp_idx) - y));
    // if (sample_idx == DEBUG_SAMPLE_IDX && team.team_rank() == 0)
    //   dprintf("y_it = %2d, y = %f, out_firelevel = %f, out_f1 = %f, out_f2
    ↪ = %f, v = %f, d = %f\n", y_it, y, out_firelevel, out_f1, out_f2, v,
    ↪ d);
    // y += lr * v;
    // if (sample_idx == DEBUG_SAMPLE_IDX && lane_idx == 0)
    //   dprintf("y_it = %2d, j = %d, y = %f, out_firelevel = %f, out_f1 =
    ↪ %f, out_f2 = %f, v = %f, d = %f\n", y_it, j, y_population(j),
    ↪ out_firelevel_population(j), out_f1, out_f2, v_population(j), d);
    y_population(warp_idx, j) += v_population(warp_idx, j);
}

float max_out_firelevel_value = 0.f, max_out_firelevel_mean_y = 0.f,
    ↪ max_out_firelevel_mean_y2 = 0.f;
int max_out_firelevel_count = 0;
for (int j = 0; j < population_size; ++j) {
    const float y = y_population(warp_idx, j);
    const float out_firelevel = out_firelevel_population(warp_idx, j);
    if (out_firelevel > max_out_firelevel_value) {
        max_out_firelevel_value = out_firelevel;
        max_out_firelevel_mean_y = y;
        max_out_firelevel_mean_y2 = y * y;
        max_out_firelevel_count = 1;
    } else if (out_firelevel == max_out_firelevel_value) {
        max_out_firelevel_mean_y += y;
        max_out_firelevel_mean_y2 += y * y;
        ++max_out_firelevel_count;
    }
}
max_out_firelevel_mean_y /= max_out_firelevel_count;
o = max_out_firelevel_value;
y = max_out_firelevel_mean_y;
break;
}
default:
    y = y_means(warp_idx);
}

predictions_population_device(pso_idx, sample_idx) = y;

```

```

// if (lane_idx == 0) dprintf("[sample_idx = %3d] y_mean=%f y_std=%f y_pred=%f
→ y_true=%f o=%f\n", sample_idx, y_means(warp_idx), y_stds(warp_idx), y,
→ point_means()[DIM-1], o);
}

warp_random_pool.free_state(warp_gen);
thread_random_pool.free_state(thread_gen);
};

// auto get_kernel_func = [=] {
//   auto parallel_for = Kokkos::Impl::ParallelFor(fuzzy_infer_block_lambda,
→ team_policy);
//   using LaunchBounds = typename decltype(parallel_for)::LaunchBounds;
//   return Kokkos::Impl::CudaParallelLaunch<decltype(parallel_for),
→ LaunchBounds>::get_kernel_func();
// };
// Kokkos::Impl::CudaInternal::singleton().cuda_func_set_attribute_wrapper(
//   get_kernel_func(), cudaFuncAttributeMaxDynamicSharedMemorySize, 98300);

// Kokkos::print_configuration(std::cout, true);

Kokkos::parallel_for(team_policy, fuzzy_infer_block_lambda);
}

extern "C"
void fuzzy_fit_impl(
  const float *points_means, const float *points_stds, const unsigned points_size,
  → const unsigned point_dims,
  float *rules_means, float *rules_stds, const unsigned rules_count,
  const ImplType impl_type, const DefuzMethod defuz_method,
  const unsigned ftv_discretization_size, const unsigned defuz_pso_population_size,
  → const unsigned defuz_pso_iter_count,
  const unsigned rules_pso_population_size, const unsigned rules_pso_iterations_count
)
{
  assert(point_dims == DIM);

  const bool debug = (bool)get_env<int>("DEBUG", 0);

  Kokkos::InitializationSettings settings;
  settings.set_num_threads(1);
  settings.set_device_id(0);
  if (!Kokkos::is_initialized()) {
    Kokkos::initialize(settings);
  }
}

```

```

}

// cudaStream_t stream;
// cudaStreamCreate(&stream);
// Kokkos::push_finalize_hook([stream] { cudaStreamDestroy(stream); });
Kokkos::Serial serial;
Kokkos::OpenMP openmp;
Kokkos::Cuda cuda;

using ExecutorSpace = Kokkos::Cuda;
using CudaMemorySpace = ExecutorSpace::memory_space;

assert(points_size >= DIM);

Kokkos::View<const Point<DIM>*, Kokkos::HostSpace,
    Kokkos::MemoryTraits<Kokkos::Unmanaged>> points_means_host((const
    → Point<DIM>*)points_means, points_size);
Kokkos::View<const Point<DIM>*, Kokkos::HostSpace,
    Kokkos::MemoryTraits<Kokkos::Unmanaged>>
    → points_stds_host(Kokkos::view_wrap((const Point<DIM>*)points_stds), points_size);
Kokkos::View<Point<DIM>*, Kokkos::HostSpace, Kokkos::MemoryTraits<Kokkos::Unmanaged>>
    → rules_means_host(Kokkos::view_wrap((Point<DIM>*)rules_means), rules_count);
Kokkos::View<Point<DIM>*, Kokkos::HostSpace, Kokkos::MemoryTraits<Kokkos::Unmanaged>>
    → rules_stds_host(Kokkos::view_wrap((Point<DIM>*)rules_stds), rules_count);

auto points_means_device = Kokkos::create_mirror_view_and_copy(CudaMemorySpace{},
    → points_means_host);
auto points_stds_device = Kokkos::create_mirror_view_and_copy(CudaMemorySpace{},
    → points_stds_host);
if (debug) std::cout << "size: " << points_means_host.size() << ", " <<
    → points_means_device.size() << std::endl;
if (debug) std::cout << typeid(points_means_device).name() << std::endl;

int shared_mem_per_block = 0;
cudaDeviceGetAttribute(&shared_mem_per_block, cudaDevAttrMaxSharedMemoryPerBlock, 0);
if (debug) std::cout << "shared_mem_per_block = " << shared_mem_per_block <<
    → std::endl;
// __global__ void
    → Kokkos::Impl::cuda_parallel_launch_local_memory<Kokkos::Impl::ParallelFor<Kokkos::Impl::
    → Kokkos::CudaSpace>, float, true>,
    → Kokkos::RangePolicy<Kokkos::Device<Kokkos::Cuda, Kokkos::CudaSpace>,
    → Kokkos::IndexType<long> > >, Kokkos::Device<Kokkos::Cuda, Kokkos::CudaSpace> >(

```

```

// Kokkos::Impl::ParallelFor<Kokkos::Impl::ViewValueFunctor<Kokkos::Device<Kokkos::Cuda,
// Kokkos::CudaSpace>, float, true>,
// Kokkos::RangePolicy<Kokkos::Device<Kokkos::Cuda, Kokkos::CudaSpace>,
// Kokkos::IndexType<long> > > const&);
// cudaFuncSetAttribute(
//   Kokkos::Impl::cuda_parallel_launch_local_memory<
//     Kokkos::Impl::ParallelFor<
//       Kokkos::Impl::ViewValueFunctor<
//         Kokkos::Device<Kokkos::Cuda, Kokkos::CudaSpace>, float, true>,
//         Kokkos::RangePolicy<Kokkos::Device<Kokkos::Cuda, Kokkos::CudaSpace>,
//         Kokkos::IndexType<long> >
//       >,
//       Kokkos::Device<Kokkos::Cuda, Kokkos::CudaSpace>
//     >
//   >

// Compute points std
float points_mean = 0.f, points_std = 0.f;
for (int i = 0; i < points_size; ++i) { points_mean += points_means_host(i)[0]; }
points_mean /= points_size;
for (int i = 0; i < points_size; ++i) { points_std +=
  Kokkos::pow(points_means_host(i)[0] - points_mean, 2); }
points_std = Kokkos::sqrt(points_std / points_size);
dprintf("points_mean = %f, points_std = %f\n", points_mean, points_std);

// auto gauss_distance = [](float a_mean, float a_std, float b_mean, float b_std) {
//   const float a_area = a_std * Kokkos::sqrt(Kokkos::numbers::pi_v<float>);
//   const float b_area = b_std * Kokkos::sqrt(Kokkos::numbers::pi_v<float>);
//   const float ab_area =
//     Kokkos::sqrt(2*Kokkos::numbers::pi_v<float> * Kokkos::pow(a_std * b_std, 2) /
//     (a_std*a_std + b_std*b_std))
//   * Kokkos::exp(-Kokkos::pow(a_mean - b_mean, 2) / (2 * (a_std*a_std +
//     b_std*b_std)));
//   return Kokkos::sqrt(a_area + b_area - 2 * ab_area);
// };

// const unsigned ftv_discretization_size =
//   get_env<unsigned>("FTV_DISCRETIZATION_SIZE", 51);
// const unsigned defuz_pso_population_size =
//   get_env<unsigned>("DEFUZ_PSO_POPULATION_SIZE", 100);
const unsigned block_replication_count = get_env<unsigned>("BLOCK_REPLICATION_COUNT",
  1);

#if 0

```

```

Kokkos::View<ArborX::Point<DIM>*, CudaMemorySpace>
→ rules_means_device("rules_means_device", rules_count);
Kokkos::View<ArborX::Point<DIM>*, CudaMemorySpace>
→ rules_stds_device("rules_stds_device", rules_count);
// One-to-one rule generation: use every training data point as a rule.
Kokkos::parallel_for(
    Kokkos::RangePolicy<Kokkos::Cuda>(0, points_size),
    KOKKOS_LAMBDA(const int idx) {
        for (int dim = 0; dim < DIM; ++dim) {
            rules_means_device(idx)[dim] = points_means_device(idx)[dim];
            // rules_stds_device(idx)[dim] = 0.01f;
            rules_stds_device(idx)[dim] = points_stds_device(idx)[dim];
        }
    }
);
Kokkos::deep_copy(rules_means_host, rules_means_device);
Kokkos::deep_copy(rules_stds_host, rules_stds_device);
Kokkos::fence();

Kokkos::View<float**, Kokkos::HostSpace> test_host("test_host", 2, 3);
printf("test_host.strides = [%ld, %ld]\n", test_host.stride_0(),
→ test_host.stride_1());
Kokkos::View<float**, Kokkos::CudaSpace> test_device("test_device", 2, 3);
printf("test_device.strides = [%ld, %ld]\n", test_device.stride_0(),
→ test_device.stride_1());

{
    assert(rules_count == points_size);

    Kokkos::View<float*, CudaMemorySpace> predictions_device("predictions_device",
→ points_size);

    Kokkos::View<Point<DIM>**, CudaMemorySpace,
→ Kokkos::MemoryTraits<Kokkos::Unmanaged>> rules_means_population_device(
        rules_means_device.data(), 1, rules_means_device.extent(0));
    Kokkos::View<Point<DIM>**, CudaMemorySpace,
→ Kokkos::MemoryTraits<Kokkos::Unmanaged>> rules_stds_population_device(
        rules_stds_device.data(), 1, rules_stds_device.extent(0));
    Kokkos::View<float**, CudaMemorySpace, Kokkos::MemoryTraits<Kokkos::Unmanaged>>
→ predictions_population_device(
        predictions_device.data(), 1, predictions_device.extent(0));

    // Initialize random pools.
    Kokkos::Random_XorShift64_Pool<Kokkos::Cuda> warp_random_pool(42);
    warp_random_pool.init(42, rules_means_device.extent(0) * block_replication_count);
}

```

```

Kokkos::Random_XorShift64_Pool<Kokkos::Cuda> thread_random_pool(42);
thread_random_pool.init(42, rules_means_device.extent(0) * block_replication_count
→ * 32);

// Call fuzzy_infer with DIM set to 5.
fuzzy_infer<DIM>(
    points_means_device,
    points_stds_device,
    predictions_population_device,
    rules_means_population_device,
    rules_stds_population_device,
    warp_random_pool,
    thread_random_pool,
    impl_type,
    defuz_method,
    ftv_discretization_size,
    defuz_pso_population_size,
    defuz_pso_iter_count,
    block_replication_count
);

{
    auto predictions_host = Kokkos::create_mirror_view_and_copy(Kokkos::HostSpace{}, 
→ predictions_device);
    metrics<float> m;
    for (int i = 0; i < points_size; ++i)
    {
        float trueVal = points_means_host(i)[DIM-1];
        float predVal = predictions_host(i);
        m(trueVal, predVal);
    }
    std::cout << "MAE: " << m.mae
        << "\nRMSE: " << m.rmse()
        << "\nNDEI: " << m.ndei()
        << "\nER2: " << m.er2() << std::endl;
}
}

#else
const unsigned pso_population_size = get_env<unsigned>("PSO_POPULATION_SIZE",
→ rules_pso_population_size);
const int pso_iterations_count = get_env<int>("PSO_ITERATIONS_COUNT",
→ rules_pso_iterations_count);
Kokkos::View<ArborX::Point<DIM>**, CudaMemorySpace, Kokkos::MemoryManaged>
→ rules_means_population_device("rules_means_population_device",
→ pso_population_size, rules_count);

```

```

Kokkos::View<ArborX::Point<DIM>**, CudaMemorySpace, Kokkos::MemoryManaged>
    ↵ rules_stds_population_device("rules_stds_population_device", pso_population_size,
    ↵ rules_count);
Kokkos::View<ArborX::Point<DIM>**, CudaMemorySpace>
    ↵ rules_means_v_population_device("rules_means_v_population_device",
    ↵ pso_population_size, rules_count);
Kokkos::View<ArborX::Point<DIM>**, CudaMemorySpace>
    ↵ rules_stds_v_population_device("rules_stds_v_population_device",
    ↵ pso_population_size, rules_count);
Kokkos::View<ArborX::Point<DIM>**, CudaMemorySpace>
    ↵ rules_means_best_population_device("rules_means_best_population_device",
    ↵ pso_population_size, rules_count);
Kokkos::View<ArborX::Point<DIM>**, CudaMemorySpace>
    ↵ rules_stds_best_population_device("rules_stds_best_population_device",
    ↵ pso_population_size, rules_count);
Kokkos::View<ArborX::Point<DIM>*, CudaMemorySpace>
    ↵ rules_means_best_device("rules_means_best_device", rules_count);
Kokkos::View<ArborX::Point<DIM>*, CudaMemorySpace>
    ↵ rules_stds_best_device("rules_stds_best_device", rules_count);
Kokkos::View<float**, CudaMemorySpace, Kokkos::MemoryManaged>
    ↵ predictions_population_device("predictions_population_device",
    ↵ pso_population_size, points_size);
Kokkos::View<float*, Kokkos::HostSpace>
    ↵ population_scores_host("population_scores_host", pso_population_size);
auto population_scores_device = Kokkos::create_mirror(CudaMemorySpace{}, 
    ↵ population_scores_host);
Kokkos::View<float*, Kokkos::HostSpace>
    ↵ population_best_scores_host("population_best_scores_host", pso_population_size);
Kokkos::deep_copy(population_best_scores_host, 1000000000.f);
float best_score = 10000000.f;

Kokkos::Random_XorShift64_Pool<> pso_random_pool(/* seed = */ 42);
pso_random_pool.init(42, pso_population_size * rules_count);
Kokkos::Random_XorShift64_Pool<> warp_random_pool(/* seed = */ 42);
warp_random_pool.init(42, pso_population_size * points_size);
Kokkos::Random_XorShift64_Pool<> thread_random_pool(/* seed = */ 42);
thread_random_pool.init(42, pso_population_size * points_size * 32);

Kokkos::parallel_for(Kokkos::RangePolicy<Kokkos::Cuda>(0, pso_population_size *
    ↵ rules_count), KOKKOS_LAMBDA(int idx) {
    auto pso_gen = pso_random_pool.get_state(idx);
    unsigned i = idx / rules_count;
    unsigned j = idx % rules_count;
    for (int dim = 0; dim < DIM; ++dim) {

```

```

rules_means_population_device(i, j)[dim] = pso_gen.frand(points_mean - 10.f *
    ↵ points_std, points_mean + 10.f * points_std);
rules_stds_population_device(i, j)[dim] = pso_gen.frand(0.f, 10.f * points_std);
rules_means_v_population_device(i, j)[dim] = pso_gen.frand(-10.f * points_std,
    ↵ 10.f * points_std);
rules_stds_v_population_device(i, j)[dim] = pso_gen.frand(-10.f * points_std,
    ↵ 10.f * points_std);
}

});

auto tqdm = tq::trange(pso_iterations_count);
for (int pso_it : tqdm) {
// for (int i = 0; i < rules_count; ++i) {
//   dprintf("rule %3d:", i);
//   for (int dim = 0; dim < DIM; ++dim) {
//     dprintf(" [%f %f]", rules_means_population_host(0, i)[dim],
// → rules_stds_population_host(0, i)[dim]);
//   }
// }
fuzzy_infer<DIM>(
    points_means_device,
    points_stds_device,
    predictions_population_device,
    rules_means_population_device,
    rules_stds_population_device,
    warp_random_pool,
    thread_random_pool,
    impl_type,
    defuz_method,
    ftv_discretization_size,
    defuz_pso_population_size,
    defuz_pso_iter_count,
    block_replication_count
);
Kokkos::fence();

// Calculate RMSE for each pso_idx
Kokkos::parallel_for(Kokkos::RangePolicy<Kokkos::Cuda>(0, pso_population_size),
    ↵ KOKKOS_LAMBDA(const int p) {
    metrics<float> m;
    for (int j = 0; j < points_size; ++j) {
        float pred = predictions_population_device(p, j);
        float truth = points_means_device(j)[DIM-1];
        m(truth, pred);
    }
}

```

```

    population_scores_device(p) = m.ndei();
});

Kokkos::fence();
Kokkos::deep_copy(population_scores_host, population_scores_device);

for (int i = 0; i < pso_population_size; ++i) {
    if (population_scores_host(i) < population_best_scores_host(i)) {
        population_best_scores_host(i) = population_scores_host(i);
        Kokkos::deep_copy(Kokkos::subview(rules_means_best_population_device, i,
                                         Kokkos::ALL), Kokkos::subview(rules_means_population_device, i,
                                         Kokkos::ALL));
        Kokkos::deep_copy(Kokkos::subview(rules_stds_best_population_device, i,
                                         Kokkos::ALL), Kokkos::subview(rules_stds_population_device, i,
                                         Kokkos::ALL));
        if (population_best_scores_host(i) < best_score) {
            best_score = population_best_scores_host(i);
            Kokkos::deep_copy(rules_means_best_device,
                               Kokkos::subview(rules_means_best_population_device, i, Kokkos::ALL));
            Kokkos::deep_copy(rules_stds_best_device,
                               Kokkos::subview(rules_stds_best_population_device, i, Kokkos::ALL));
        }
    }
}

// dprintf("[%3d] %f best;", pso_it, best_score);
// for (int i = 0; i < pso_population_size; ++i) dprintf(" %f",
// → population_scores_host(i));
// dprintf("\n");
tqdm << best_score << " best;";
for (int i = 0; i < pso_population_size; ++i) tqdm << ' ' <<
→ population_scores_host(i);
tqdm << '\n';

const float lr_p = 0.5f, lr_g = 0.5f, w = 0.5f;
Kokkos::parallel_for(Kokkos::RangePolicy<Kokkos::Cuda>(0, pso_population_size *
                                         rules_count), KOKKOS_LAMBDA(int idx) {
    auto pso_gen = pso_random_pool.get_state(idx);
    unsigned i = idx / rules_count;
    unsigned j = idx % rules_count;
    for (int dim = 0; dim < DIM; ++dim) {
        {
            float rp = pso_gen.frand(), rg = pso_gen.frand();
            rules_means_v_population_device(i, j)[dim] = w *
                rules_means_v_population_device(i, j)[dim]
    }
}
}

```

```

+ lr_p * rp * (rules_means_best_population_device(i, j)[dim] -
    ↳ rules_means_population_device(i, j)[dim])
+ lr_g * rg * (rules_means_best_device(j)[dim] -
    ↳ rules_means_population_device(i, j)[dim]);
rules_means_population_device(i, j)[dim] +=
    ↳ rules_means_v_population_device(i, j)[dim];
}

{
    float rp = pso_gen.frand(), rg = pso_gen.frand();
    rules_stds_v_population_device(i, j)[dim] = w *
        ↳ rules_stds_v_population_device(i, j)[dim]
        + lr_p * rp * (rules_stds_best_population_device(i, j)[dim] -
            ↳ rules_stds_population_device(i, j)[dim])
        + lr_g * rg * (rules_stds_best_device(j)[dim] -
            ↳ rules_stds_population_device(i, j)[dim]);
    rules_stds_population_device(i, j)[dim] += rules_stds_v_population_device(i,
        ↳ j)[dim];
}
}

pso_random_pool.free_state(pso_gen);
});

}

Kokkos::deep_copy(rules_means_host, rules_means_best_device);
Kokkos::deep_copy(rules_stds_host, rules_stds_best_device);
#endif

// cudaStreamDestroy(stream);
}

extern "C"
void fuzzy_predict_impl(
    const float *points_means, const float *points_stds, float *predictions, const
    ↳ unsigned points_size, const unsigned point_dims,
    const float *rules_means, const float *rules_stds, const unsigned rules_count,
    const ImplType impl_type, const DefuzMethod defuz_method,
    const unsigned ftv_discretization_size, const unsigned defuz_pso_population_size,
    ↳ const unsigned defuz_pso_iter_count
)
{
    assert(point_dims == DIM);

Kokkos::InitializationSettings settings;
settings.set_num_threads(1);

```

```

settings.set_device_id(0);
if (!Kokkos::is_initialized()) {
    Kokkos::initialize(settings);
}
Kokkos::Serial serial;
Kokkos::OpenMP openmp;
Kokkos::Cuda cuda;

assert(points_size >= DIM);

Kokkos::View<const Point<DIM>*, Kokkos::HostSpace,
    Kokkos::MemoryTraits<Kokkos::Unmanaged>>
    points_means_host(Kokkos::view_wrap((const Point<DIM>*)points_means),
    points_size);
Kokkos::View<const Point<DIM>*, Kokkos::HostSpace,
    Kokkos::MemoryTraits<Kokkos::Unmanaged>>
    points_stds_host(Kokkos::view_wrap((const Point<DIM>*)points_stds), points_size);
Kokkos::View<const Point<DIM>*, Kokkos::HostSpace,
    Kokkos::MemoryTraits<Kokkos::Unmanaged>>
    rules_means_host(Kokkos::view_wrap((const Point<DIM>*)rules_means), rules_count);
Kokkos::View<const Point<DIM>*, Kokkos::HostSpace,
    Kokkos::MemoryTraits<Kokkos::Unmanaged>> rules_stds_host(Kokkos::view_wrap((const
    Point<DIM>*)rules_stds), rules_count);

auto points_means_device = Kokkos::create_mirror_view_and_copy(Kokkos::CudaSpace{}, 
    points_means_host);
auto points_stds_device = Kokkos::create_mirror_view_and_copy(Kokkos::CudaSpace{}, 
    points_stds_host);

// const unsigned ftv_discretization_size =
//     get_env<unsigned>("FTV_DISCRETIZATION_SIZE", 51);
// const unsigned defuz_pso_population_size =
//     get_env<unsigned>("DEFUZ_PSO_POPULATION_SIZE", 100);
const unsigned block_replication_count = get_env<unsigned>("BLOCK_REPLICATION_COUNT",
    1);

auto rules_means_device = Kokkos::create_mirror_view_and_copy(Kokkos::CudaSpace{}, 
    rules_means_host);
auto rules_stds_device = Kokkos::create_mirror_view_and_copy(Kokkos::CudaSpace{}, 
    rules_stds_host);
Kokkos::View<float*, Kokkos::HostSpace, Kokkos::MemoryTraits<Kokkos::Unmanaged>>
    predictions_host(predictions, points_size);
Kokkos::View<float*, Kokkos::CudaSpace> predictions_device("predictions_device",
    points_size);

```

```

{

Kokkos::View<Point<DIM>**, Kokkos::CudaSpace,
→ Kokkos::MemoryTraits<Kokkos::Unmanaged>> rules_means_population_device(
    rules_means_device.data(), 1, rules_means_device.extent(0));
Kokkos::View<Point<DIM>**, Kokkos::CudaSpace,
→ Kokkos::MemoryTraits<Kokkos::Unmanaged>> rules_stds_population_device(
    rules_stds_device.data(), 1, rules_stds_device.extent(0));
Kokkos::View<float**, Kokkos::CudaSpace, Kokkos::MemoryTraits<Kokkos::Unmanaged>>
→ predictions_population_device(
    predictions_device.data(), 1, predictions_device.extent(0));

// Initialize random pools.
Kokkos::Random_XorShift64_Pool<Kokkos::Cuda> warp_random_pool(42);
warp_random_pool.init(42, rules_means_device.extent(0) * block_replication_count);
Kokkos::Random_XorShift64_Pool<Kokkos::Cuda> thread_random_pool(42);
thread_random_pool.init(42, rules_means_device.extent(0) * block_replication_count
→ * 32);

// Call fuzzy_infer with DIM set to 5.
fuzzy_infer<DIM>(
    points_means_device,
    points_stds_device,
    predictions_population_device,
    rules_means_population_device,
    rules_stds_population_device,
    warp_random_pool,
    thread_random_pool,
    impl_type,
    defuz_method,
    ftv_discretization_size,
    defuz_pso_population_size,
    defuz_pso_iter_count,
    block_replication_count
);

Kokkos::deep_copy(predictions_host, predictions_device);

{
    metrics<float> m;
    for (int i = 0; i < points_size; ++i)
    {
        float trueVal = points_means_host(i)[DIM-1];
        float predVal = predictions_host[i];
        m(trueVal, predVal);
    }
}

```

```
    }

    std::cout << "MAE: " << m.mae
        << "\nRMSE: " << m.rmse()
        << "\nNDEI: " << m.ndei()
        << "\nER2: " << m.er2() << std::endl;
    }
}

Kokkos::parallel_for(Kokkos::RangePolicy(openmp, 0, points_size), KOKKOS_LAMBDA(int
    i) { assert(std::abs(predictions_host(i) != NAN)); });
}
```

Приложение Б

Свидетельства о государственной регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025663372

Среда для параллельного обобщенного нечеткого вывода на основе нечеткого значения истинности

Правообладатель: *федеральное государственное бюджетное образовательное учреждение высшего образования «Белгородский государственный технологический университет им. В.Г. Шухова» (RU)*

Авторы: *Синюк Василий Григорьевич (RU), Каратач Сергей Александрович (RU)*

Заявка № 2025661081

Дата поступления 05 мая 2025 г.

Дата государственной регистрации

в Реестре программ для ЭВМ 28 мая 2025 г.

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c10b300fb54f240f67070cc2026
Владелец Зубов Юрий Сергеевич

Ю.С. Зубов

Действителен с 10.07.2024 по 03.10.2025



РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025663464

**Библиотека моделирования временных рядов на основе
параллельного нечеткого вывода с использованием
нечеткого значения истинности**

Правообладатель: *федеральное государственное бюджетное
образовательное учреждение высшего образования
«Белгородский государственный технологический
университет им. В.Г. Шухова» (RU)*

Авторы: *Синюк Василий Григорьевич (RU), Каратач Сергей
Александрович (RU)*

Заявка № 2025660990

Дата поступления **05 мая 2025 г.**

Дата государственной регистрации

в Реестре программ для ЭВМ **28 мая 2025 г.**

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДСИГНОУ
Сертификат: 0692e7d9a63009f547240f670bcfa2026
Владелец: Зубов Юрий Сергеевич
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

