

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 и алгоритмов обработки временных
рядов в задачах их прогнозирования на основе нечетких
систем с учетом нечеткости входов**

Специальность 2.3.1 —
«Системный анализ, управление и обработка информации, статистика»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
кандидат технических наук, профессор
Синюк Василий Григорьевич

Белгород — 2025

Оглавление

Стр.

Введение	5
Глава 1. Анализ методов прогнозирования временных рядов в задачах управления с позиции адекватности учета уникальности свойств генерирующих их объектов	10
1.1 Роль прогнозирования в задачах управления и принятия решений	10
1.2 Прогнозирование по временным рядам для уникальных объектов	11
1.3 Условия адекватности учета неопределенности временных рядов статистическими методами	14
1.4 Анализ адекватности учета неопределенности временных рядов распространеными моделями прогнозирования	17
1.5 Оценка адекватности учета неопределенности временных рядов в подходах на основе теории нечетких множеств	21
1.6 Постановка задачи	22
1.7 Выводы по главе	23
Глава 2. Разработка метода прогнозирования временных рядов с использованием нечетких систем на основе правил	25
2.1 Задача нечеткого логического вывода	25
2.2 Применение предложенного метода вывода в нечеткой модели для задачи прогнозирования временных рядов	26
2.2.1 Несинглтонная фазификация	27
2.2.2 Адаптивная процедура несинглтонной фазификации временных последовательностей	30
2.2.3 Выбор схемы дефазификации.	35
2.3 Альтернативный метод нечеткого вывода с полиномиальной вычислительной сложностью	37
2.3.1 Нечеткое значение истинности	37
2.3.2 Вычисление нечеткого значения истинности, когда функции принадлежности высказываний задаются гауссовыми функциями	44

2.3.3	Метод вывода на основе НЗИ	44
2.3.4	Вывод логического типа на основе нечеткого значения истиинности	47
2.3.5	Свойства вывода типа Мамдани на основе нечеткого значения истиинности	49
2.4	Выводы по главе	49
Глава 3. Разработка алгоритмической реализации метода прогнозирования временных рядов на основе нечетких моделей		51
3.1	Параллельный алгоритм свертки НЗИ	51
3.2	Адаптация алгоритма PSO для построения базы правил.	53
3.3	Применение алгоритма оптимизации на основе метода роя частиц для вычисления дефазификации по среднему максимуму	55
3.4	Процедура метода прогнозирования временных рядов на основе разработанной нечеткой модели	60
3.5	Выводы по главе	64
Глава 4. Разработка информационной технологии прогнозирования временных рядов, прототипа ее программной реализации на основе нечеткой модели и оценка работоспособности		65
4.1	Библиотека с параллельной реализацией нечеткого вывода на основе технологии CUDA	65
4.1.1	Вычисление нечеткого значения истиинности посредством дискретизации	70
4.1.2	Реализация обертки в Python-модуль	74
4.2	Оценка работоспособности разработанного метода прогнозирования на основе нечеткого логического вывода относительно других типов нечетких моделей и с позиции вычислительной сложности	76
4.3	Оценка работоспособности разработанной информационной технологии прогнозирования на основе нечеткой модели относительно вероятностных подходов.	81

4.4 Выводы по главе	87
Заключение	89
Список литературы	90
Список рисунков	92
Список таблиц	95
Приложение А. Текст программного кода Python-модуля нечеткого вывода с использованием технологии CUDA	96
Приложение Б. Свидетельства о государственной регистрации программ для ЭВМ	125
Приложение В. Справка о практическом применении результатов диссертационной работы в организации	128

Введение

Процесс управления сводится к принятию решений и контролю за их реализацией. Принимаемые решения должны быть обоснованными. Методы обоснования решений входят в системы информационного обеспечения управления. Одним из таких методов является использование результатов анализа возможных исходов при различном воздействии на объект управления. Для анализа выполняется прогнозирование этих исходов. Прогнозирование часто осуществляется по временным рядам, которые генерируются при регистрации значений параметров некоторого контролируемого процесса. Обоснованность принимаемых на основе прогноза решений зависит от адекватности извлечения и учета неопределенности временных данных используемой моделью прогнозирования.

Решением задачи прогнозирования временных рядов занималось большое количество исследователей. В результате среди прочих выделился набор широко используемых моделей прогнозирования, таких как линейный предсказатель и модель Бокса-Дженкинса. Такие подходы используют для получения вероятностных оценок характеристик временных рядов свойство стационарности и статистическое усреднение по ансамблю характеристик объектов. Неопределенность при этом часто рассматривается как ошибки измерений, подчиненные конкретному закону распределения, то есть происходит «навязывание законов природе». Таким образом, из-за наложения этих ограничений вероятностные модели не совсем адекватны.

Альтернативным подходом может служить использование нечетких систем на основе правил с фаззификацией типа non-singleton разработанной Л. Заде теории нечетких множеств. Тогда как вероятностные модели основаны на усреднении характеристик временных рядов. Для задания функции принадлежности при этом не требуется иметь ансамбль временных рядов или свойство стационарности отдельного временного ряда.

Распространенные подходы нечеткого вывода, выработанные Э. Мамдани, П. Ларсеном, Т. Такаги, М. Сугено и Ю. Цукамато, используют четкие значения входов, полученные в результате фаззификации singleton для простоты реализации. Из-за такого упрощения теряется информация о неопределенности входных значений временных рядов. Использование фаззификации типа

non-singleton позволяет сохранить информацию о нечеткости входных данных, но приводит к экспоненциальной вычислительной сложности нечеткого логического вывода при многих выходах модели. Тогда, чтобы использовать нечеткий логический вывод при фазификации non-singleton, необходимо создать нечеткую модель с разработкой эффективных алгоритмов.

Таким образом, задача разработки информационной технологии прогнозирования на основе нечеткого вывода при фазификации типа non-singleton является актуальной.

Проблемы, связанные с нечетким выводом и нечетким моделированием в России изучались и прорабатывались А.Н. Аверкин, В.В Борисов, И.А. Ходашинский, А.В. Язенин, Н.Г. Ярушкина. За рубежом развитием нечеткой теории занимались Р. Angelov, D. Dubois, H. Ishibuchi, J.M. Mendel, L. Rutkowski, H. Tanaka, R.R. Yager, T. Yasukawa, L. Wang и др..

Объектом исследования являются информационная технология управления на основе анализа временных рядов.

Предметом исследования являются методы и алгоритмы прогнозирования временных рядов на основе нечеткого моделирования.

Целью данной работы является совершенствование информационного обеспечения управления на основе разработки информационной технологии и алгоритмов прогнозирования временных рядов с использованием нечетких систем, адекватно учитывающих уникальность объектов.

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

1. Анализ методов прогнозирования временных рядов в задачах управления с позиции адекватности учета уникальности свойств генерирующих их объектов.
2. Разработка метода прогнозирования временных рядов с использованием нечетких систем на основе правил.
3. Разработка алгоритмической реализации метода прогнозирования временных рядов на основе нечетких моделей.
4. Разработка информационной технологии прогнозирования временных рядов и прототипа ее программной реализации.
5. Оценка работоспособности программно-алгоритмической реализации на основе вычислительных экспериментов.

Научной новизной обладают следующие результаты диссертационного исследования:

1. Модель нечеткого вывода на основе нечеткого значения истинности (НЗИ), позволяющая адекватно учесть изменчивость характеристик входных данных.
2. Метод вывода для систем MISO-структуры логического типа и типа Мамдани на основе нечеткого правила «Если $нзи$ есть ИСТИННО, то y есть B », снижающий экспоненциальную сложность до полиномиальной.
3. Метод прогнозирования временных рядов на основе разработанной системы нечеткого вывода.
4. Алгоритмы реализации метода прогнозирования на основе нечеткого вывода с применением технологии параллельных вычислений.

Теоретическая значимость определяет принцип учета нечеткости входных данных, модель нечеткого вывода на этой основе и способы получения результата.

Практическая значимость: определяется возможностью построения систем прогнозирования временных рядов в задачах принятия решения с учетом неопределенности временных рядов на основе разработанных результатов. Применение разработанной информационной технологии позволяет осуществить прогнозирование временных рядов с изменчивыми характеристиками в задачах принятия решения. Теоретические результаты и разработанный прототип программного обеспечения используется в учебном процессе БГТУ им. В. Г. Шухова. Среди них две зарегистрированные программы ЭВМ:

1. Свидетельство №2025661081 «Среда для параллельного обобщенного нечеткого вывода на основе нечеткого значения истинности».
2. Свидетельство №2025660990 «Библиотека моделирования временных рядов на основе параллельного нечеткого вывода с использованием нечеткого значения истинности».

Разработанная информационная технология принята к использованию в ПАО Сбербанк.

Работа выполнена при участии в проекте РФФИ №20-07-00030 «Разработка высокопроизводительных методов интеллектуального анализа данных на основе нечёткого моделирования и создание компьютерной системы поддержки принятия решений для классификации и прогнозирования».

Методология и методы исследования. В работе использовалась методология принятия решений, методы теории временных рядов, теории нечетких множеств и машинного обучения.

Основные положения, выносимые на защиту:

1. Применение разработанной информационной технологии позволяет расширить системы управления для принятия обоснованных решений на основе временных рядов, описывающих уникальные объекты.
2. Разработанное алгоритмическое обеспечение и программная реализация, которая не предъявляет особых требований к архитектуре вычислительных сред.
3. Результаты проведенных вычислительные эксперименты иллюстрируют работоспособность информационной технологии.

Область исследования. Содержание диссертации соответствует следующим пунктам паспорта специальности 2.3.1. «Системный анализ, управление и обработка информации, статистика» по следующим направлениям исследований:

- п. 1 паспорта специальности: Теоретические основы и методы системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта.
- п. 4 паспорта специальности: Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта.

Достоверность полученных результатов обеспечивается корректностью математических преобразований, отсутствием противоречий с известными положениями теории и практики прогнозирования временных рядов, адекватным учетом нечеткость входных данных и иллюстрируется результатами вычислительных экспериментов о работоспособности информационной технологии, а также результатами в публикациях в рецензируемых журналах и на конференциях.

Апробация работы. Основные результаты работы докладывались на следующих конференциях:

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) 2023», 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 автора. Все результаты диссертационного исследования получены либо автором лично, либо при его непосредственном участии.

Публикации. Основные результаты по теме диссертации изложены в 11 печатных изданиях, 3 из которых изданы в журналах, рекомендованных ВАК, 3 – в периодических научных журналах, индексируемых Web of Science и Scopus, 5 – в тезисах докладов. Зарегистрированы 2 программы для ЭВМ.

Объем и структура работы. Диссертация состоит из введения, 4 глав, заключения и 3 приложений. Полный объём диссертации составляет 128 страниц, включая 37 рисунков и 1 таблицу. Список литературы содержит 14 наименований.

Глава 1. Анализ методов прогнозирования временных рядов в задачах управления с позиции адекватности учета уникальности свойств генерирующих их объектов

1.1 Роль прогнозирования в задачах управления и принятия решений

Прогнозирование является фундаментальным элементом информационного обеспечения управленческой деятельности, выступая критически важным инструментом поддержки принятия решений в различных сферах. Его основная функция заключается в научно обоснованном предвидении возможных тенденций развития управляемого объекта или процесса.

Принципиальное значение прогнозирования в управлении состоит в формировании предпосылок для принятия обоснованных решений. На этапе, предшествующем непосредственному планированию, прогнозирование позволяет определить реальность и целесообразность потенциальных стратегических направлений.

Ключевая роль прогнозирования проявляется в нескольких ключевых аспектах:

- Научное предвидение: Прогнозирование обеспечивает выявление тенденций и закономерностей развития исследуемых систем, что создает фундамент для стратегического планирования.
- Информационная поддержка: Процесс прогнозирования направлен на получение научно обоснованных вариантов изменения управляемого объекта во времени и пространстве.
- Антиципация рисков: Через прогнозирование возможно заблаговременное определение потенциальных негативных последствий и рисков при выборе управленческих решений. Многовариантность сценариев: Прогнозирование позволяет разрабатывать множественные сценарии развития событий, охватывающие различные потенциальные траектории.

Особую значимость прогнозирование приобретает в условиях высокой неопределенности и динамической изменчивости внешней среды. Оно транс-

формируется из простого инструмента предсказания в сложный механизм поддержки адаптивного управления.

В контексте принятия решений прогнозирование выступает не просто методом экстраполяции существующих тенденций, а сложным аналитическим инструментом, интегрирующим статистические методы, экспертные оценки и современные технологии обработки данных.

Таким образом, прогнозирование является не только вспомогательным, но и системообразующим элементом процесса управления, обеспечивающим научную обоснованность, стратегическую дальновидность и риск-ориентированность принимаемых решений.

1.2 Прогнозирование по временным рядам для уникальных объектов

Информационная система (ИС) — это взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации для достижения цели управления. Она автоматизирует и поддерживает бизнес-процессы, а также решает конкретные информационные задачи в рамках заданной предметной области.

Информационная система поддержки принятия решений часто использует *информационную технологию* прогнозирования для формирования материала для принятия обоснованных решений.

Информационная технология (ИТ) — это совокупность методов, процессов, программного обеспечения, оборудования и сетей, предназначенных для сбора, обработки, хранения, передачи и защиты данных и информации.

Относительно информационной системы, информационные технологии представляют собой конкретные инструменты (аппаратное обеспечение, программное обеспечение, средства коммуникации), с помощью которых строится и работает информационная система, как показано на рисунке 1.1. *Информационная технология прогнозирования* тогда является инструментом прогнозирования в информационной системе на основе поступающей на вход информации.

Информационная технология прогнозирования, как правило, представляет собой программную реализацию некоторого метода прогнозирования,

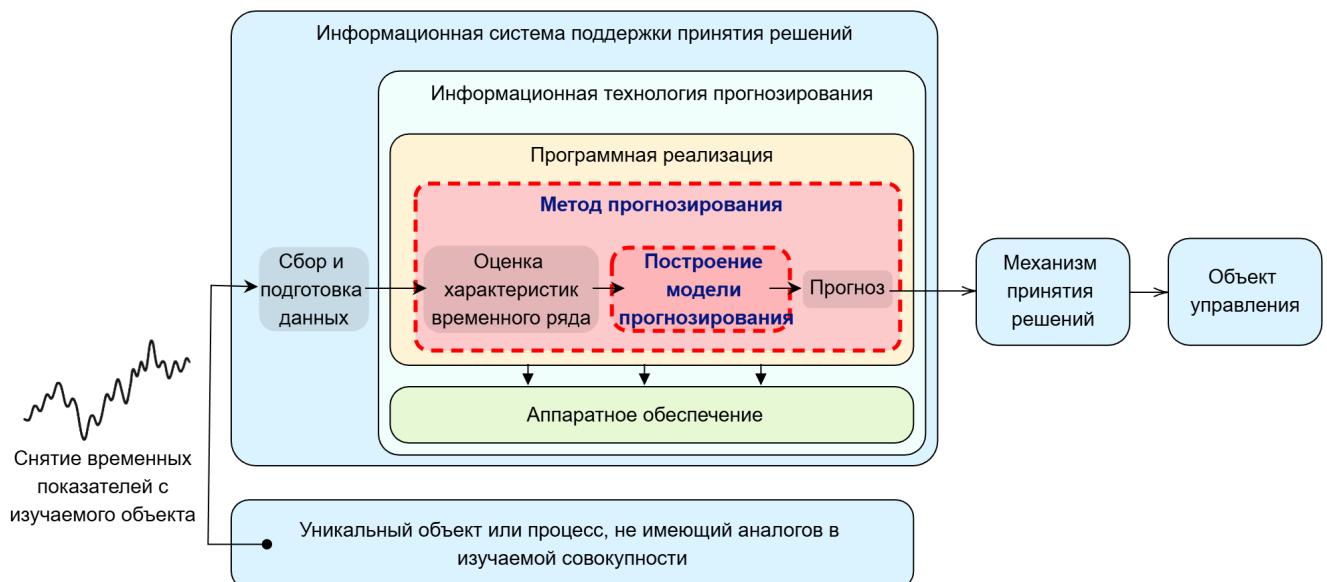


Рисунок 1.1 — Схема системы управления на основе информационной технологии прогнозирования.

опирающуюся на возможности аппаратной платформы (например, использующую технологии массивно-параллельных вычислений для распараллеливания процедуры прогноза).

Метод в машинном обучении — это алгоритм или способ построения модели, то есть процедура, набор правил или подход, по которым из обучающих данных формируется модель. Метод определяет, как будут анализироваться данные и как будет происходить обучение. Метод прогнозирования использует некоторую *модель* прогнозирования.

Модель — описание явления или процесса, достаточное для достижения цели решаемой задачи. Модель прогнозирования — описание протекающего во времени процесса, достаточное для получения прогноза требуемой точности.

Входные данные такой информационной системы включают *временной ряд*, то есть последовательность значений, составленную посредством фиксации измеренных показателей в определенном временном промежутке, отражающую развитие некоторого процесса или состояния объекта. Объект управления при этом может отличаться от источника генерации временных данных.

Часто производится снятие временных рядов показателей сразу с некоторой совокупности объектов или группы протекающих параллельно процессов. Но в отдельных случаях такие объекты или процессы являются *的独特的*, то есть не имеют ни одного полностью идентичного аналога в изучаемой совокупности. Такие объекты или процессы являются единственными в своем роде

или настолько индивидуализированными, что невозможно создать репрезентативную выборку из аналогичных объектов для классического статистического анализа. В контексте прогнозирования временных рядов это означает, что для этих объектов или процессов есть только один доступный временной ряд наблюдений, и невозможно опереться на «типичные» свойства популяции.

Далее приведены несколько примеров систем, являющихся уникальными объектами.

Пример 1. Гидроэлектростанция с уникальным географическим положением.

Рассмотрим пример временного ряда, описывающего дневной объём воды, проходящей через плотину конкретной гидроэлектростанции (ГЭС).

Поскольку каждая ГЭС имеет уникальное географическое расположение, климатические и гидрологические условия, невозможно найти другое предприятие с идентичными параметрами.

Пример 2. Пациент с уникальными физиологическими показателями.

Ещё более сложный случай — оценка воздействия лекарственного препарата на физиологические показатели конкретного человека.

Если усреднить эффект препарата по множеству пациентов, можно получить усреднённое значение, однако оно не отражает индивидуальную реакцию конкретного организма, поскольку каждый человек имеет свои особенности — генетику, метаболизм, историю болезни, образ жизни.

Более корректным подходом является измерение индивидуального отклика — наблюдение за изменениями показателей при многократном приёме одинаковой дозы лекарства одним пациентом в разные моменты времени. В этом случае формируется индивидуальный временной ряд реакции на препарат.

Пример 3. Корпоративные клиенты банка с уникальной бизнес-моделью в уникальной экономической среде.

Корпоративных клиентов банка можно рассматривать как уникальные системы, потому что каждый из них представляет собой сложную, многокомпонентную организацию со своими собственными параметрами, которые формируют уникальный «профиль» взаимодействия с банком.

Каждая компания имеет свою структуру управления, юридическую форму, подразделения, процессы принятия решений — это влияет на то, как она взаимодействует с банком, кто уполномочен проводить операции, как организованы финансовые потоки.

В отличие от физических лиц, компании отличаются видами деятельности, отраслевой спецификой, логистикой и цепочками поставок, сезонностью бизнеса.

1.3 Условия адекватности учета неопределенности временных рядов статистическими методами

В области технических наук понятие «адекватность» часто используется в контексте моделирования и описывает степень соответствия модели реальному объекту или процессу. Адекватность модели машинного обучения (или, в более общем смысле, любой модели) — это степень соответствия модели реальной системе или объекту, который она описывает, с учетом цели моделирования. Это понятие подразумевает, что модель должна совпадать с моделируемой системой в отношении поставленных задач, обеспечивая приемлемую точность и полезность для принятия решений.

В данной работе анализируется адекватность моделей прогнозирования с позиции адекватности учета неопределенности временных рядов посредством использования статистических характеристик согласно предусловиям применения вероятностных оценок. Можно выделить следующие условия корректного вычисления статистических характеристик для ансамбля объектов:

1. *Однородность совокупности наблюдений.* Все элементы ансамбля должны принадлежать одной и той же статистической совокупности, то есть представлять собой реализации одной случайной величины или одного стохастического процесса с неизменными параметрами распределения. Нарушение однородности приводит к несопоставимости выборок и некорректности оценок момента первого и второго порядков.
2. *Статистическая независимость или контролируемая структура зависимости.* Для корректного применения классических оценок математического ожидания и дисперсии, а также для справедливости стандартных асимптотических результатов (закон больших чисел, центральная предельная теорема), требуется независимость наблюдений. В случаях, когда зависимости неизбежны (например, присутствует автокорреляция), должна быть известна или оценима структура кор-

реляций с последующим учётом её влияния на дисперсию оценок и эффективный размер выборки.

3. *Асимптотическая нормальность оценок.* Согласно центральной предельной теореме, при достаточно большом числе независимых и одинаково распределённых наблюдений выборочное среднее стремится к нормальному распределению независимо от вида исходного закона. Минимально необходимое значение n определяется свойствами распределения; часто используемое практическое правило $n \geq 30$ является лишь приближённой эвристикой. Для получения оценки математического ожидания с заданной абсолютной погрешностью ε и доверительной вероятностью $1 - \alpha$ используется оценка размера ансамбля

$$n \approx \left(\frac{z_{1-\alpha/2}\sigma}{\varepsilon} \right)^2,$$

где σ — стандартное отклонение, а $z_{1-\alpha/2}$ — квантиль стандартного нормального распределения.

При ограниченном объёме ансамбля и отсутствии уверенности в нормальности распределения оценок иногда возможно использовать t-распределение (для приблизительно нормальных данных) либо непараметрические методы, такие как бутстрэп, позволяющие получать доверительные интервалы и характеристики точности без строгих предположений о распределении.

Если условия вычисления статистических характеристик по ансамблю не выполняются, ансамблевое усреднение заменяется временным (как часто делается в статистической физике, климатологии и теории сигналов).

Эргодичность временного ряда. Чтобы такая замена была корректной, необходимо, чтобы усреднение по времени было эквивалентно усреднению по ансамблю. Этим свойством обладают *эргодические процессы*.

Процесс называется *эргодическим*, если его временные средние совпадают со статистическими (математическими) средними:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T x_t = M(x_t)$$

с вероятностью 1.

Иными словами, эргодичность — это свойство, позволяющее оценивать характеристики всего стохастического процесса по одной достаточно длинной

реализации. Иногда это свойство называют *свойством хорошего перемешивания*, подразумевая, что значения ряда, взятые в разные моменты времени, образуют выборку, эквивалентную множеству значений, полученных при повторных экспериментах в один и тот же момент.

Для того чтобы стационарный процесс был эргодичным, достаточно, чтобы автоковариация быстро убывала с увеличением лага, то есть выполнялось условие:

$$\lim_{n \rightarrow \infty} \left(\frac{1}{n} \sum_{k=1}^n \gamma(k) \right) = 0,$$

где $\gamma(k)$ — функция автоковариации.

Формально эргодичность это свойство всего временного процесса. На практике часто говорят о *эргодичности реализации* временного ряда.

Свойство стационарности временных рядов. В анализе временных рядов важное значение имеют стационарные процессы, вероятностные свойства которых не изменяются во времени. Именно такие ряды чаще всего применяются для описания случайных (стохастических) составляющих наблюдаемых данных, поскольку позволяют делать устойчивые статистические выводы и строить предсказательные модели.

Временной ряд $x_t, t = 1, 2, \dots, n$, называется *строго стационарным* (или *стационарным в узком смысле*), если его совместное распределение вероятностей для любых n последовательных наблюдений x_1, x_2, \dots, x_n совпадает с распределением наблюдений $x_{1+k}, x_{2+k}, \dots, x_{n+k}$ при любом целом сдвиге k . Иными словами, закон распределения и его характеристики не зависят от конкретного момента времени. Следовательно, *математическое ожидание* и *среднее квадратическое отклонение* стационарного ряда постоянны:

$$M(x_t) = a = \text{const}, \quad \sigma(x_t) = \sigma = \text{const}.$$

Эти параметры можно оценить по наблюдениям x_t следующим образом:

$$\bar{x} * t = \frac{1}{n} \sum *t = 1^n x_t,$$

$$\sigma_t^2 = \frac{1}{n} \sum_{t=1}^n (x_t - \bar{x}_t)^2.$$

Одним из простейших примеров стационарного временного ряда является ***белый шум*** — последовательность некоррелированных случайных величин

с нулевым математическим ожиданием и постоянной дисперсией. В частности, ошибки e_t в классической линейной регрессионной модели считаются белым шумом, а при нормальном распределении — нормальным (гауссовским) белым шумом.

Стохастический процесс можно рассматривать как функцию двух аргументов: времени и случайности. Если случайность зафиксирована, мы получаем одну реализацию процесса — конкретную наблюдаемую последовательность.

Однако теоретические характеристики, такие как математическое ожидание $M(x_t)$, определяются как *усреднение по ансамблю* — то есть по множеству всех возможных реализаций процесса в данный момент времени. На практике же в нашем распоряжении обычно *только одна реализация*, и повторные наблюдения в один и тот же момент времени невозможны.

Поэтому при анализе временных рядов мы вынуждены заменять усреднение по ансамблю ***усреднением по времени***, рассчитывая средние значения и другие характеристики на основе одной реализации.

Стоит подчеркнуть, что не всякий стационарный процесс является эргодическим, хотя для практических целей наличие стационарности часто подразумевает эргодичность.

Стационарность гарантирует неизменность статистических характеристик во времени. Эргодичность позволяет использовать эти характеристики, вычисленные по одной реализации, как оценки теоретических (ансамблевых) значений.

Таким образом, одной стационарности недостаточно для корректной статистической обработки временных рядов — необходима также эргодичность, обеспечивающая достоверность оценок, полученных на основе наблюдаемых данных.

1.4 Анализ адекватности учета неопределенности временных рядов распространенными моделями прогнозирования

Линейный предсказатель. Каноническая форма линейного предсказателя вычисляет оценку будущего значения как линейную комбинацию прошлых на-

блудений. Модель представляет формулой свертки:

$$\hat{x}_k = \sum_{i=1}^p \beta_i x_{k-i},$$

где β_1, \dots, β_p — коэффициенты предсказателя (параметры фильтра), x_{k-i} — прошлые значения ряда, а \hat{x}_k — оценка (прогноз) значения x_k по прошлым наблюдениям.

Прогноз этой модели будет точным только тогда, когда удовлетворяют коэффициентам разностного уравнения, коэффициенты которого известны. К таким временным рядам относится, например, аддитивные комбинации синусоид с различными частотами. Однако, такие модели являются идеализацией, а значения их параметров (амплитуды, частоты и начальные фазы) заранее неизвестны и их надо оценивать, что снижает точность прогноза.

Модель «Бокса-Дженкинса» представляет собой классический и один из наиболее распространённых подходов к прогнозированию временных рядов на основе линейных стохастических моделей. Его основная идея заключается в том, что динамику наблюдаемых данных можно описать комбинацией авторегрессионных и скользящих средних компонент, параметры которых оцениваются по прошлым значениям ряда, а затем используются для построения прогноза. Метод был разработан Джорджем Боксом и Гвильямом Дженкинсом в 1970-е годы и с тех пор стал стандартом при анализе и моделировании стационарных временных последовательностей.

В основе подхода лежит представление о временном ряде как о реализации стационарного линейного стохастического процесса. Наиболее общая форма модели, применяемой в методе Бокса–Дженкинса, записывается как

$$\varphi(B)(1 - B)^d x_t = \theta(B)\varepsilon_t,$$

где (x_t) — значение временного ряда в момент времени t , B — оператор запаздывания ($Bx_t = x_{t-1}$), (ε_t) — белый шум с нулевым средним и постоянной дисперсией (σ^2), $(\varphi(B))$ и $(\theta(B))$ — многочлены операторов авторегрессии и скользящего среднего соответственно, а d — порядок дифференцирования, обеспечивающий стационарность. В зависимости от выбора параметров получаются частные случаи: $AR(p)$ — авторегрессионная модель, $MA(q)$ — модель скользящего среднего, $ARMA(p,q)$ — их комбинация и $ARIMA(p,d,q)$ — модель для нестационарных рядов, приведённых к стационарности посредством дифференцирования.

Построение прогностической модели по Боксу–Дженкинсу включает несколько последовательных этапов.

1. *Предобработка данных и обеспечение стационарности.* На этом шаге устраняются тренды и сезонные компоненты, так как метод применим только к стационарным рядам. Используются дифференцирование, логарифмирование или вычитание сезонных средних.
2. *Идентификация модели.* Здесь подбираются порядки p , d и q , описывающие структуру зависимости. Основные инструменты идентификации — автокорреляционная функция (ACF) и частичная автокорреляционная функция (PACF). Для модели AR(p) характерно затухание ACF и обрыв PACF после лага p , тогда как для MA(q) — наоборот. Эти статистические характеристики строятся на основе ковариационной структуры ряда и позволяют предположить тип модели.
3. *Оценка параметров.* После выбора структуры модели параметры (φ_i) и (θ_j) оцениваются методами максимального правдоподобия или наименьших квадратов. Полученные оценки минимизируют среднеквадратическую ошибку прогноза.
4. *Диагностика модели.* Проверяется адекватность полученной модели и предпосылки метода. Остатки $(\hat{\varepsilon}_t)$ должны представлять собой белый шум — без автокорреляции и со стабильной дисперсией. Для этого анализируется ACF остатков и проводится тест Льюнга–Бокса, основанный на статистике

$$Q = n(n+2) \sum_{k=1}^m \frac{\hat{\rho}_k^2}{n-k},$$

которая при справедливости гипотезы о белом шуме имеет приближённо распределение (χ^2_{m-p-q}) . Если остатки не удовлетворяют этим условиям, модель уточняется, и процедура повторяется.

5. *Выбор и интерпретация окончательной модели.* Из нескольких кандидатных моделей выбирается та, которая обеспечивает наилучшее качество описания данных при минимальном числе параметров. Для этого используются информационные критерии Акаике (AIC) и Шварца (BIC).
6. *Прогнозирование.* После окончательной калибровки модели строятся точечные и интервальные прогнозы. Одношаговый прогноз для

AR(p)-модели определяется формулой

$$\hat{x} * t + 1 = \hat{\mu} + \sum *i = 1^p \hat{\varphi} * ix * t + 1 - i,$$

а многошаговые прогнозы получаются итеративным применением этой зависимости, заменяя будущие значения их прогнозами. Дисперсия ошибки прогноза увеличивается с горизонтом, что учитывается при построении доверительных интервалов.

Метод Бокса–Дженкинса базируется на анализе вторых моментов распределения, то есть на автокорреляционных и ковариационных характеристиках. Это делает его особенно эффективным для линейных стационарных процессов, где вся зависимость между наблюдениями описывается именно через ковариационную структуру. Основные статистические предпосылки применения метода — линейность модели, стационарность процесса, независимость и одинаковое распределение ошибок (белый шум), а также устойчивость и инвертируемость характеристических многочленов (корни должны лежать вне единичного круга).

С практической точки зрения метод требует достаточной длины временного ряда для устойчивой оценки параметров и чувствителен к структурным сдвигам, гетероскедастичности и шумам измерений. При нарушении этих условий возможны неточности в прогнозах, поэтому в современной практике часто применяются гибридные или модифицированные подходы (например, ARIMA–GARCH, регрессионные расширения или субполосные методы).

Таким образом, метод Бокса–Дженкинса представляет собой последовательную процедуру построения, проверки и использования линейной стохастической модели временного ряда, основанную на анализе автокорреляционных свойств данных. Он сочетает строгие статистические принципы с практической удобством и до сих пор остаётся базовым инструментом анализа и прогнозирования временных рядов в экономике, инженерии и естественных науках.

Для рассмотренных ранее примеров можно проанализировать

В первом примере для случая единственной ГЭС в выборке объектов нельзя оценить статистические характеристики, например дисперсию или среднее значение за конкретный день года, путём усреднения по множеству ГЭС. Однако если предположить, что поступление воды на плотину является стационарным и эргодичным процессом, то можно использовать многолетние

наблюдения для оценки: среднего объёма воды для определённого сезона, дисперсии колебаний, вероятности экстремальных притоков.

Таким образом, свойства стационарности и эргодичности позволяют корректно анализировать динамику даже уникального гидротехнического объекта, используя исключительно данные его собственных наблюдений во времени.

Во *втором примере* появляется принципиальная трудность: организм человека нестационарен. Со временем происходят возрастные изменения, колебания гормонального фона, метаболическая адаптация к лекарству, изменения образа жизни и состояния здоровья. Всё это нарушает условие стационарности временного ряда.

Даже если предположить наличие эргодичности, нарушение стационарности делает временные оценки неустойчивыми и потенциально искаженными. Для корректного анализа в таких случаях требуются специализированные методы обработки нестационарных временных рядов, включая фильтрацию, выделение трендов, нормализацию по сопутствующим показателям и использование адаптивных моделей.

В *третьем примере* усреднение по другим клиентам нельзя применять из-за различий в бизнес-моделях, однако можно использовать локально стационарные участки ряда или модели, адаптирующиеся к изменениям во времени. В целом, если объект уникален и ансамбль недоступен, статистические характеристики временного ряда приходится оценивать только по его собственным значениям, и корректность таких оценок напрямую зависит от степени стационарности и эргодичности процесса, либо от возможностей применяемых методов компенсировать нестационарность.

1.5 Оценка адекватности учета неопределенности временных рядов в подходах на основе теории нечетких множеств

Тогда как вероятность (стандартная теория Колмогорова) моделирует стохастическую неопределенность — случайность событий при повторимых экспериментах, нечёткая теория (Zadeh, Dubois–Prade и др.) моделирует нечеткость или эпистемическую неопределенность (высказывания о степени принадлежности, степени совместности состояния с наблюдением).

Теория нечетких множеств позволяет выразить неопределенность даже по единичной реализации временного ряда, используя функцию принадлежности

$$\mu : X \rightarrow [0,1].$$

Такая возможность полезна для уникальных объектов, где неопределенность может быть субъективной или основанной на экспертных оценках. Более того, методы нечетких множеств проще в применении, поскольку понятие функции принадлежности соответствует более простому аналогу вероятностной меры в теории вероятностей, что делает их удобными даже для случаев, когда неопределенность могла бы быть описана вероятностно. Это позволяет интегрировать экспертные знания напрямую, без строгих предположений о стационарности, и формализовать лингвистические переменные.

При прогнозировании нестационарных рядов нечеткие системы (например, на основе правил типа "если-то") не накладывают жестких ограничений на модель порождающего процесса, что хорошо подходит для нестационарных сценариев. Вместо этого они сопоставляют текущие данные с «образцами» в базе правил, фокусируясь на сходстве, а не на статистических предположениях о стационарности. Степень сходства отражается в результате композиции входного нечеткого множества с нечетким правилом:

$$A' \circ (A \rightarrow B).$$

Такие правила позволяют адекватно описывать слабо структурированные данные временных рядов через нечеткие множества, интерпретируя неопределенность, связанную не только со случайностью (как в вероятностных моделях), но и с недостаточной точностью датчиков, структурными сдвигами или экспертными допущениями. Это обеспечивает робастность: алгоритмы становятся нечувствительными к малым отклонениям от предположений, включая нестационарность процесса, благодаря минимаксным операциям Заде.

1.6 Постановка задачи

В результате проведения диссертационного исследования требуется разработать информационную технологию прогнозирования временных рядов,

предназначенную для применения в системах управления, где решения принимаются на основании анализа динамики параметров уникальных объектов или процессов.

Для этого нужно сперва разработать нечеткую модель, использующую нечеткую систему на основе правил с несинглтонной фазификацией. Затем сформировать метод использования разработанной нечеткой модели для прогнозирования временных рядов.

Составить алгоритмическое обеспечение для эффективной реализации этапов разработанного метода прогнозирования. Для этого требуется подготовить эффективный алгоритм нечеткого логического вывода с использованием несинглтонной фазификацией. Также в рамках разработки метода прогнозирования требуется адаптировать алгоритм построения базы правил на основе данных.

Затем следует в рамках реализации информационной технологии необходимо разработать программное обеспечение на основе предложенного метода прогнозирования с эффективной эксплуатацией возможностей аппаратной платформы.

Для разработанной информационной технологии прогнозирования следует провести оценку работоспособности. При проведении такой оценки имеет смысл продемонстрировать улучшение качества прогнозирования при применении данной информационной технологии по сравнению с другими подходами, менее адекватно учитывающими неопределенность значений временного ряда уникальных объектов. Также стоит экспериментально показать эффективность выбранной конфигурации нечеткой модели и разработанных алгоритмов.

1.7 Выводы по главе

1. При построении информационных систем поддержки принятия решений часто используется информационная технология прогнозирования на основе временных рядов. Временные ряды собираются с объектов или процессов и поступают на вход информационной системе. В некоторых случаях эти объекты или процессы являются уникальными, то есть не имеющими аналогов в изучаемой совокупности.

2. Для временных рядов, собранных с уникальных объектов или процессов, использование статистического подхода оценки неопределенности измерений не является вполне адекватным. Неадекватность вызвана тем, что для получения статистических оценок требуется выполнить усреднение по ансамблю достаточного количества объектов, либо ожидается эргодичность (или хотя бы стационарность) временного ряда.
3. Рассмотренные распространенные модели прогнозирования — линейный предсказатель и модель Бокса-Дженкинса — либо вообще не предусматривают учет неопределенности во временном ряде, либо используют для этого статистические оценки.
4. Нечеткие множества позволяют выразить неопределенность значений единичной реализации временного ряда без использования статистических оценок. Нечеткая система вывода на основе правил не накладывает ограничений на модель временного процесса и позволяет строить аппроксимирующую функцию даже для нестационарного временного ряда.

Глава 2. Разработка метода прогнозирования временных рядов с использованием нечетких систем на основе правил

2.1 Задача нечеткого логического вывода

Нечеткая система на основе правил 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, 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 2.1 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 функций нечеткой 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 (2.1), 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

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

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, нечеткое множество B'_k определяется композицией нечеткого множества \mathbf{A}' и отношения \mathbf{R}_k , т.е.

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

или, на уровне функций принадлежности

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

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 $O(|X_1| \cdot |X_2| \cdot \cdots \cdot |X_n| \cdot |Y|)$, т.е. экспоненциальная.

2.2 Применение предложенного метода вывода в нечеткой модели для задачи прогнозирования временных рядов

Пусть задан временной ряд $\{y_t\}_{t=1}^T = \{y_1, \dots, y_T\}$, где $y_t \in \mathbb{R}$ — измеренное значение наблюдаемой переменной в момент времени t , а T — длина доступной выборки. При моделировании временных последовательностей с использованием нейро-нечетких систем каждое значение $y_t \in \mathbb{Y} \subseteq \mathbb{R}$ фаззифицируется в нечеткое множество A'_t . Тогда для прогнозирования значения \hat{y}_{t+h} с горизонтом h на основании среза наблюдений y_{t+1}, \dots, y_t можно использовать нечеткую

систему с базой из N правил вида:

$$R_k : \text{Если } \bigwedge_{i=1}^p (y_{t-i+1} \text{ есть } A_{ki}), \text{ то } y_{t+1} \text{ есть } A_{kp+1}, k = \overline{1, N},$$

где p — размер окна запаздывания (порядок модели, количество входов нечеткой системы).

2.2.1 Несинглтонная фазификация

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

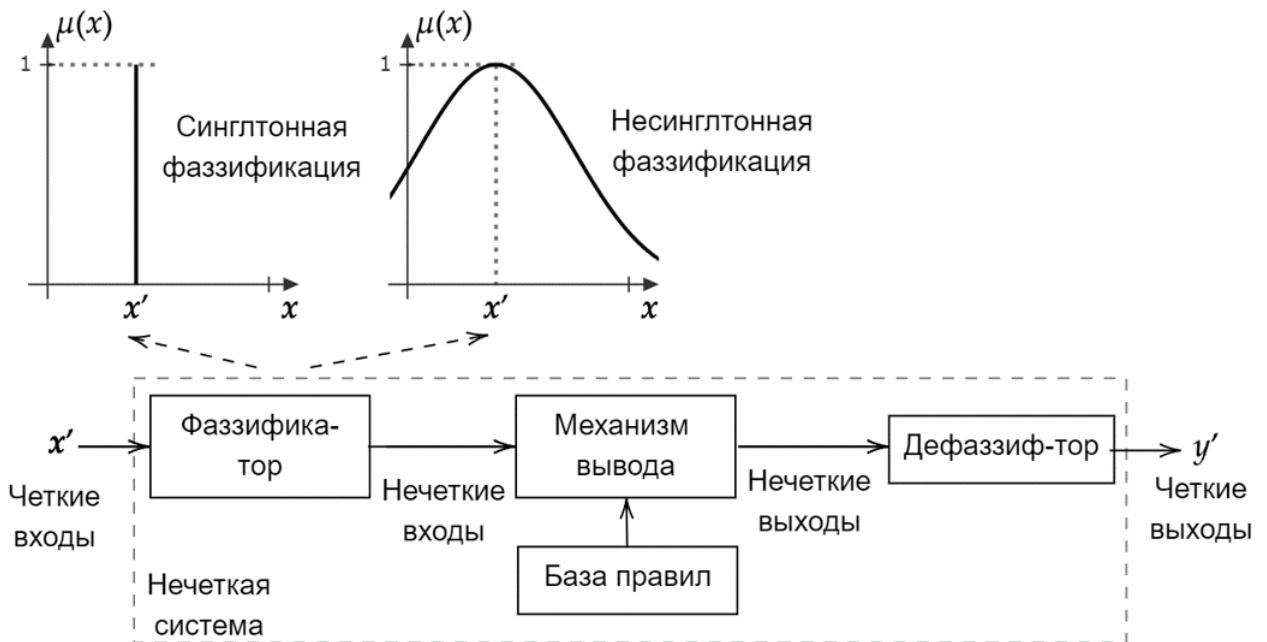


Рисунок 2.1 — Схема системы нечеткого вывода на основе правил с использованием синглтонной и несинглтонной фазификации

Нечеткая система представляется в виде композиции фазификатора, базы правил, модуля вывода и дефазификатора, как показано на рисунке 2.1. Фазификация — отображение входных данных из исходного пространства в пространство нечетких множеств. Наиболее распространенным является метод синглтонной фазификации. Основная причина его широкого использования является существенное упрощение реализации систем нечеткого вывода. При использовании синглтонной фазификации поданное на вход значение x'

интерпретируется как истинное значение измеренной величины, что эквивалентно использованию функции принадлежности входного нечеткого множества $\mu_{A'}(x) = [x = x']$. При задании значений входных нечетких множеств нечеткой системы с использованием синглтонного типа фазификации теряется информация о неопределенности измеренных значений x' и они рассматриваются как достоверные. Поэтому в работе используется *несинглтонный способ фазификации*.

Альтернативный подход с использованием несинглтонной фазификации 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.

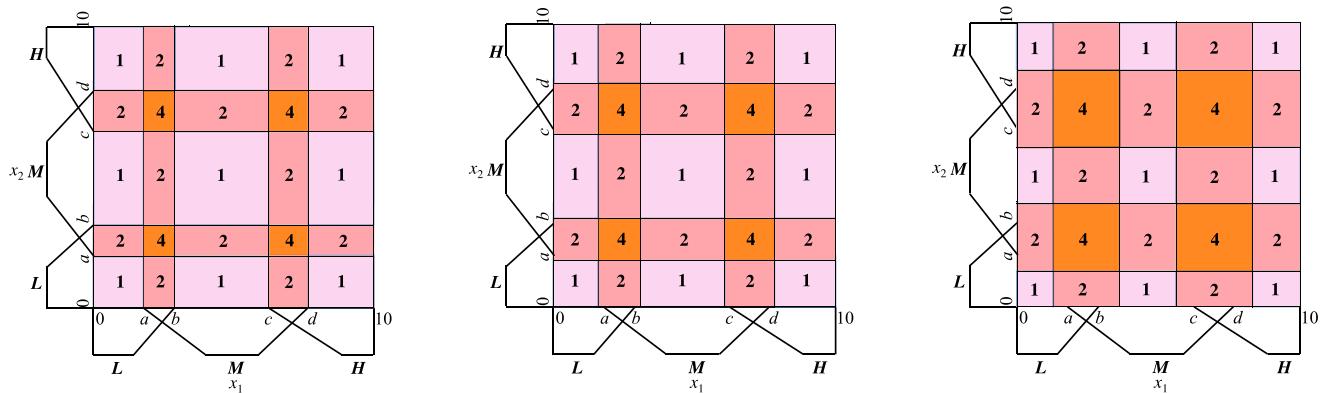
a) $\sigma_{A'} = 0\%$ б) $\sigma_{A'} = 4\%$ в) $\sigma_{A'} = 12\%$

Рисунок 2.2 — Сравнение областей активации правил при переходе от синглтонной фазификации к несинглтонной и при увеличении ширины области неопределенности $\sigma_{A'}$.

Влияние перехода от синглтонной фазификации к несинглтонной и величины окрестности погрешности на внутреннее поведение и итоговое качество нечеткой системы продемонстрировал Мендель в [1; 2].

Для анализа влияния от использования того или иного способа фазификации на корректность получаемого результата в статье [3] 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 2.3 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 про наблюдать, упомянутый ранее, эффект корректировки входного значения антецедентом третьего правила.

В случае с нечеткой системой на основе вывода 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 подходе 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 2.4 видно, что при увеличении ширины в

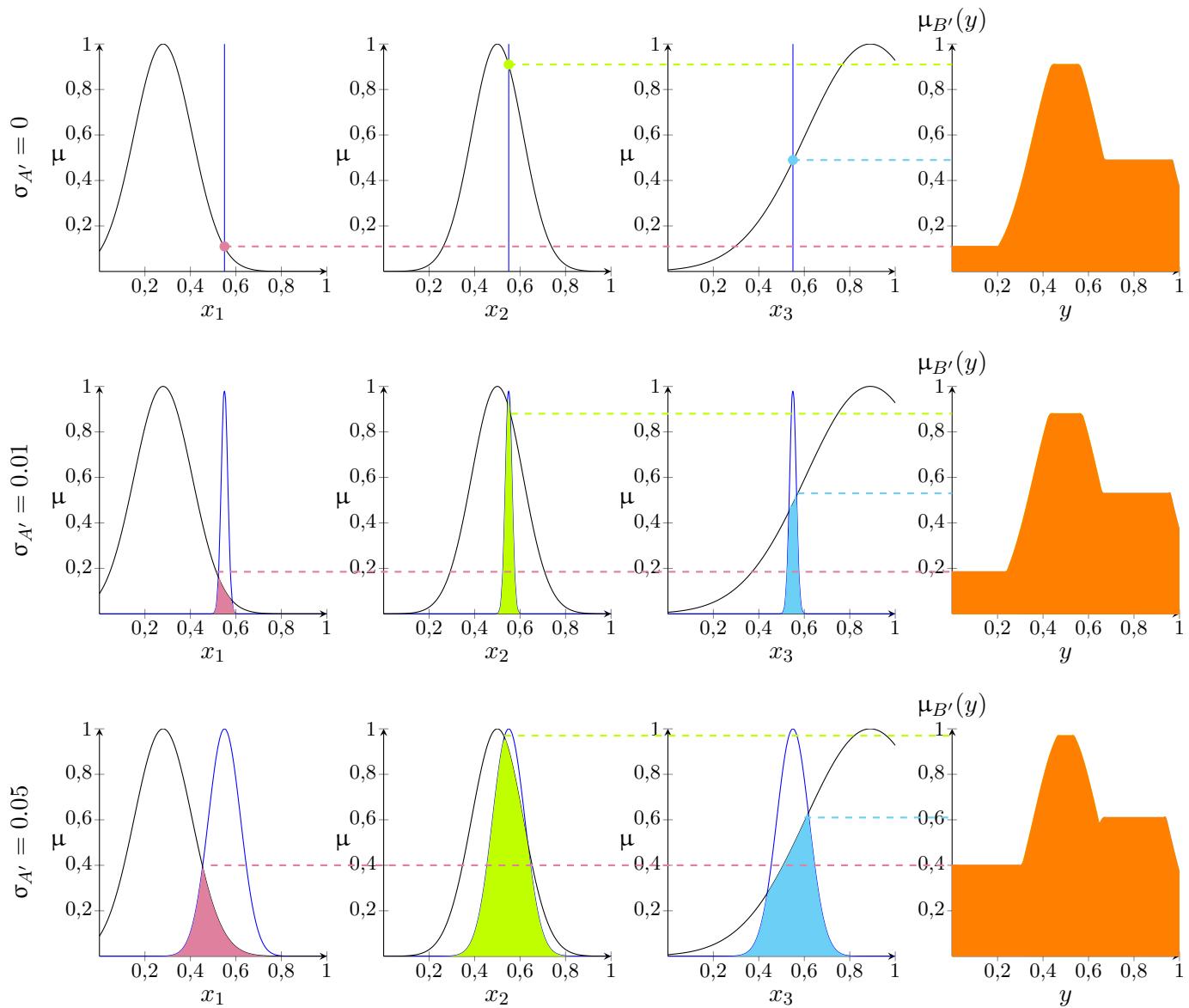


Рисунок 2.3 – Сравнение формы функций принадлежности выходных нечетких множеств для подхода Мамдани

пересечении проекций импликации на пространство выходной переменной формируется область пересечения более сложной формы.

2.2.2 Адаптивная процедура несинглтонной фазификации временных последовательностей

В задаче прогнозирования временных рядов измеренные значения могут содержать в себе неопределенность, зачастую возникающую из-за шумового искажения измерений. Такая неопределенность может быть выражена

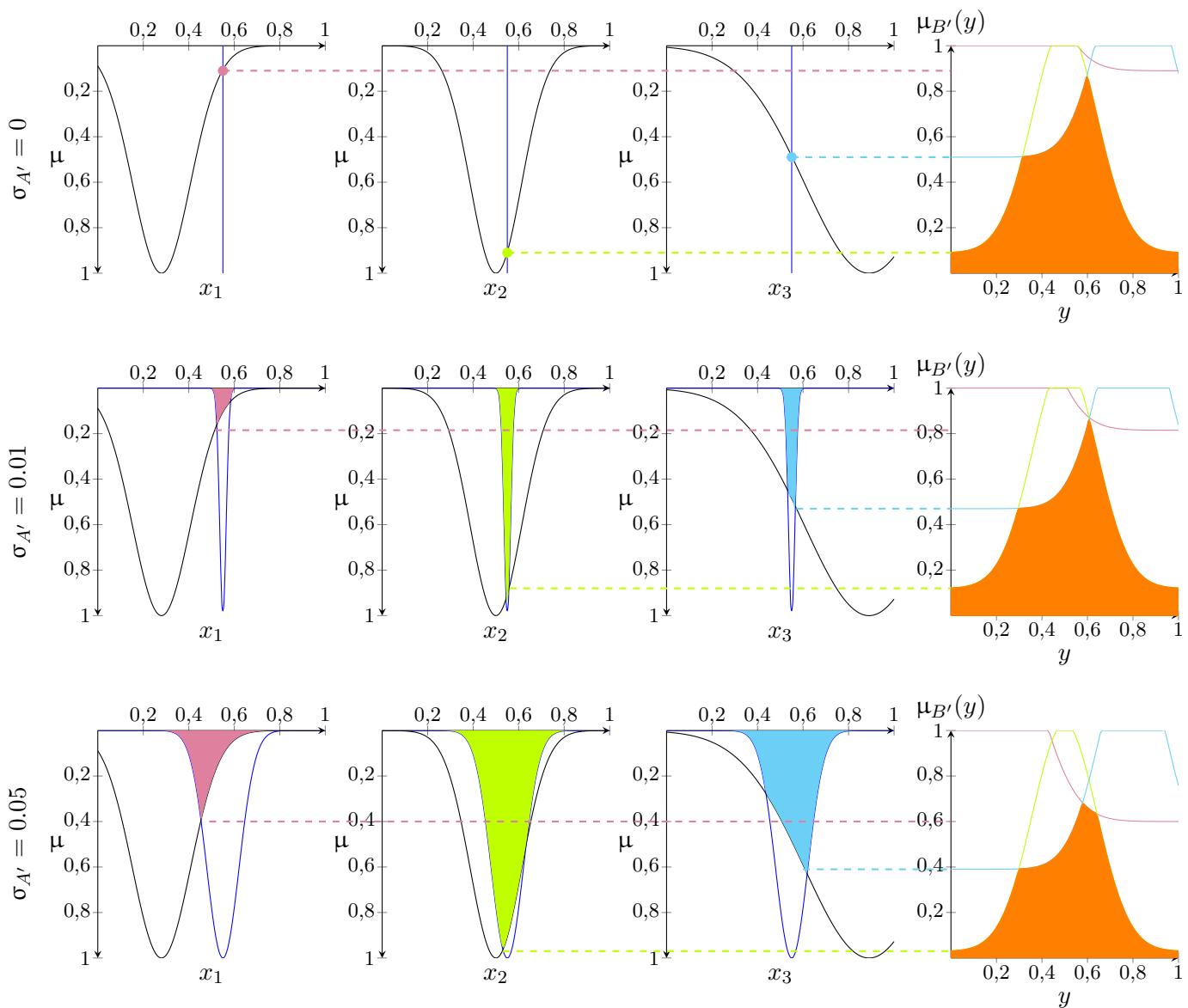


Рисунок 2.4 – Сравнение формы функций принадлежности выходных нечетких множеств для логического подхода. Выходные нечеткие множества получены в результате применения импликации Лукасевича $I(a, b) = 1 - a + b$, поэтому для легкости восприятия антецеденты правил и входные нечеткие множества показаны в виде выражений $1 - \mu_{A_k}(x)$ и $1 - \mu_{A'}(x)$.

при сопоставлении каждому измеренному значению несинглтонной функции принадлежности нечеткого множества. **Концептуально, несинглтонный фаззификатор подразумевает, что входное значение x_t является наиболее возможным значением, среди всех значений в её окрестности σ^{est} однако, поскольку входные данные неопределенны, соседние точки также, возможны, но в меньшей степени. По мере удаления от входного значения x возможность того, что оно будет правильным уменьшается. Таким образом, несинглтонные нечеткие**

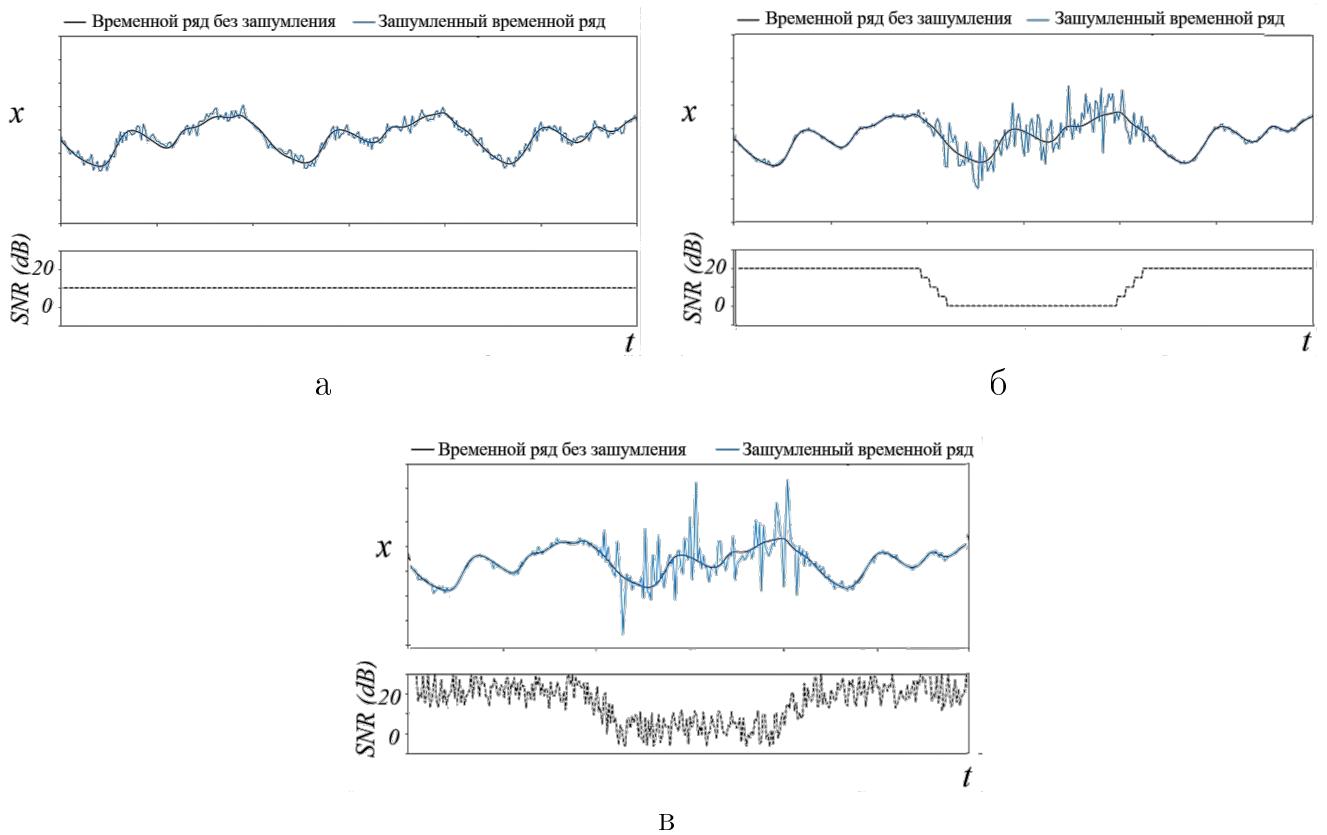


Рисунок 2.5 — Примеры временных рядов с различными типами шума: (а) стабильный шум, (б) смешанный стабильный шум, (в) переменный шум. $SNR(dB) = 10 \log_{10} \frac{\sigma_s^2}{\sigma_n^2}$ — отношение среднеквадратичного значения сигнала σ_s^2 к среднеквадратичному значению шума σ_n^2 (ОСШ, *signal-to-noise ratio*) в децибелах. Большее значение $SNR(dB)$ соответствует меньшему уровню шума.

множества могут эффективно улавливать входную неопределенность, не требуя изменений в других частях нечеткой системы, таких как антецеденты или консеквенты правил. Например, если несинглтонная входная ф. п. нечеткого множества, связанная с данным четким значением x_0 , выражается гауссовой функцией:

$$\mu(x) = \exp(x; x_0, b),$$

где b — ширина или стандартное отклонение гауссовой ф. п., которая может быть задана для отражения уровня неопределенности. На практике, когда предполагается, что входные данные системы подвержены низким уровням неопределенности, ширина или значения b соответствующих гауссовых ф. п. интуитивно мала, в то время как большие значения используются для моделирования более высоких уровней неопределенности.

Специфика оценки шумовой неопределенности временной последовательности измерений возникает поскольку, отражающей какую-то характеристику некоторого процесса, которому характерны различные виды переменчивости силы шумового воздействия. Примеры различных видов изменчивости уровня шума изображены на рис. 2.5. В [4] предложена обобщенная процедура *адаптивной оценки неопределенности* для сопоставления значений исходной последовательности с несинглтонными ф. п. входных нечетких множества. Данная процедура состоит из четырех шагов, записанных в блоках блок-схемы на рисунке 2.6:

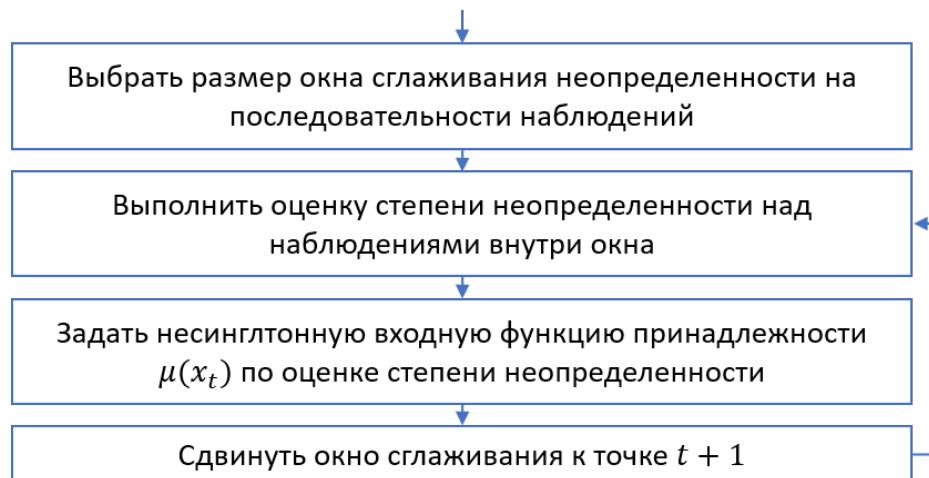


Рисунок 2.6 — Схема обобщенной процедуры адаптивной несинглтонной фазификации временной последовательности.

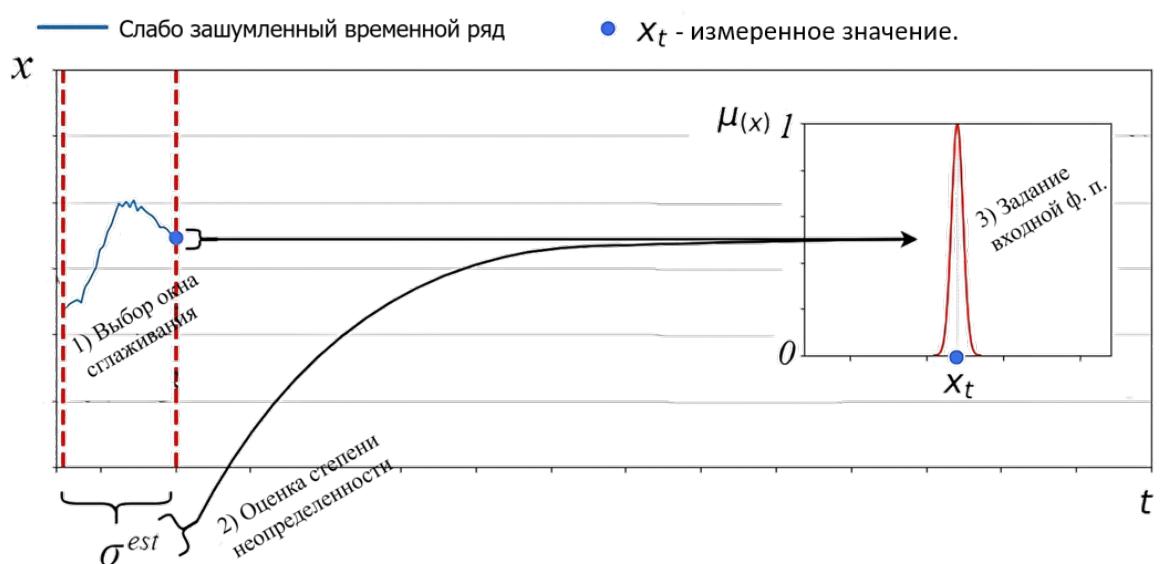


Рисунок 2.7 — Иллюстрация процедуры несинглтонной фазификации временного ряда с низким уровнем шума.

1. *Выбрать размер окна сглаживания:* Определяется размер окрестности точки во временном ряде, по которому выполняется оценка степени неопределенности в точке. Размер окна может быть фиксированным или динамическим (например, при использовании экспоненциально взвешенного скользящего среднего). Например, при использовании датчиков (например, в робототехнике) размер окна может быть выбран в зависимости от частоты съема датчиков или на основе фиксированного временного интервала. Другой подход, например, в контексте временных рядов, может заключаться в использовании алгоритма подбора для определения оптимального размера окна.
2. *Оценить степень неопределенности внутри окна:* Теперь для оценки степени неопределенности в точке используется сформированный набор наблюдений временной последовательности. В зависимости от условий эксперимента, могут использоваться различные техники оценки неопределенности: статистическая (дисперсия), максимальное правдоподобие, глубокие нейросетевые автоэнкодеры, метод главных компонент.
3. *Задать функцию принадлежности входного нечеткого множества:* Используя оценку степени неопределенности из предыдущего шага, следует задать параметры функции принадлежности входного нечеткого множества, то есть, выполнить непосредственно фазификацию.
4. *Сдвинуть окно сглаживания:* Затем следует выполнить оценку степени неопределенности и фазификацию для точки, следующей за текущей.

На рисунке 2.7 изображено как каждое несинглтонное входное нечеткое множество задается на основе оценки уровня шума, определенного по описанной только что процедуре с применением скользящего окна сглаживания. На этом рисунке показан участок наблюдений с низким уровнем шума приводящий к узкой ширине (среднеквадратичному отклонению) ф. п. входного нечеткого множества. На рисунке 2.8 уровень шума заметно больше и, как следствие, ф. п. входного нечеткого множества имеет большую ширину.

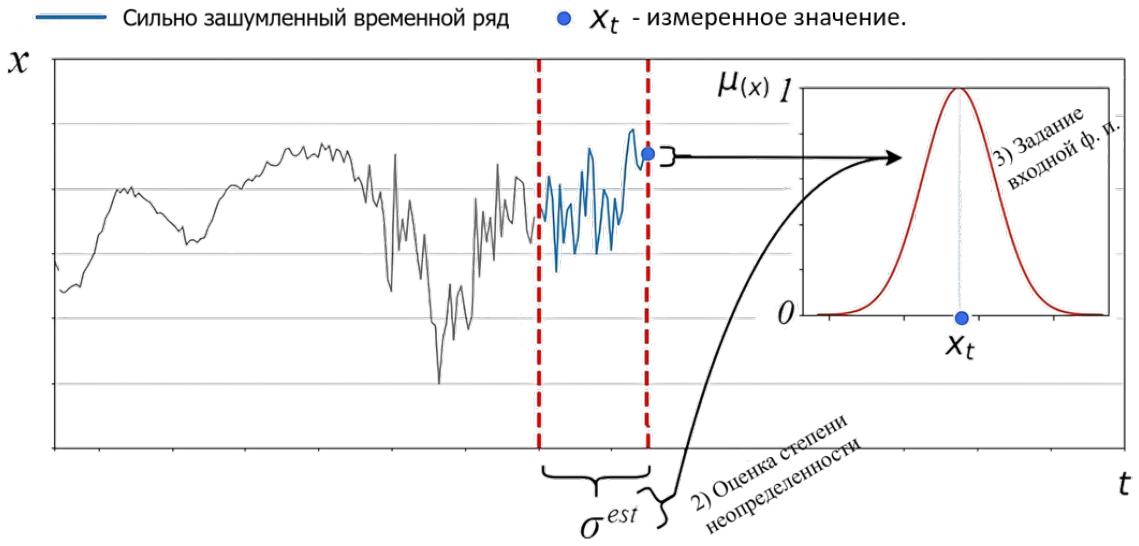


Рисунок 2.8 — Иллюстрация процедуры несинглтонной фазификации временного ряда с высоким уровнем шума.

2.2.3 Выбор схемы дефазификации.

Рассмотрим некоторые распространенные методы дефазификации, описанные в [5]:

1. Дефазификация по методу среднего центра вычисляется в центрах \bar{y}_k правил, где функция принадлежности k -го консеквента $\mu_{B'_k}(y_k)$ принимает максимальное значение:

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

2. Дефазификация по методу центра тяжести определяется отношением:

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

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 [5]. Эта схема выражается формулой ниже:

$$\hat{y}_{CoG} = \frac{\sum_{k=1}^N \bar{y}_k \mu_{B'}(\bar{y}_k)}{\sum_{k=1}^N \mu_{B'}(\bar{y}_k)}, \quad (2.5)$$

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

3. Дефазификация по методу среднего максимума

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

где $core(B') = \{y | y \in Y \text{ and } \mu_{B'}(y) = \sup_{y' \in Y} \mu_{B'}(y')\}$. В случае унимодального вида функции принадлежности $\mu_{B'}(y)$ данный способ дефазификации можно упростить до метода максимума функции принадлежности:

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

В задаче регрессии дискретная формула дефазификации по центру тяжести или другие более простые схемы (например, метод среднего центра) не показывают достаточной точности, а непрерывная формулировка первой имеет большую вычислительную сложность. Поэтому в работе для регрессии используется дефазификация по среднему максимуму.

Сетевая структура нечеткой системы для данного способа дефазификации изображена на рис. 2.9.

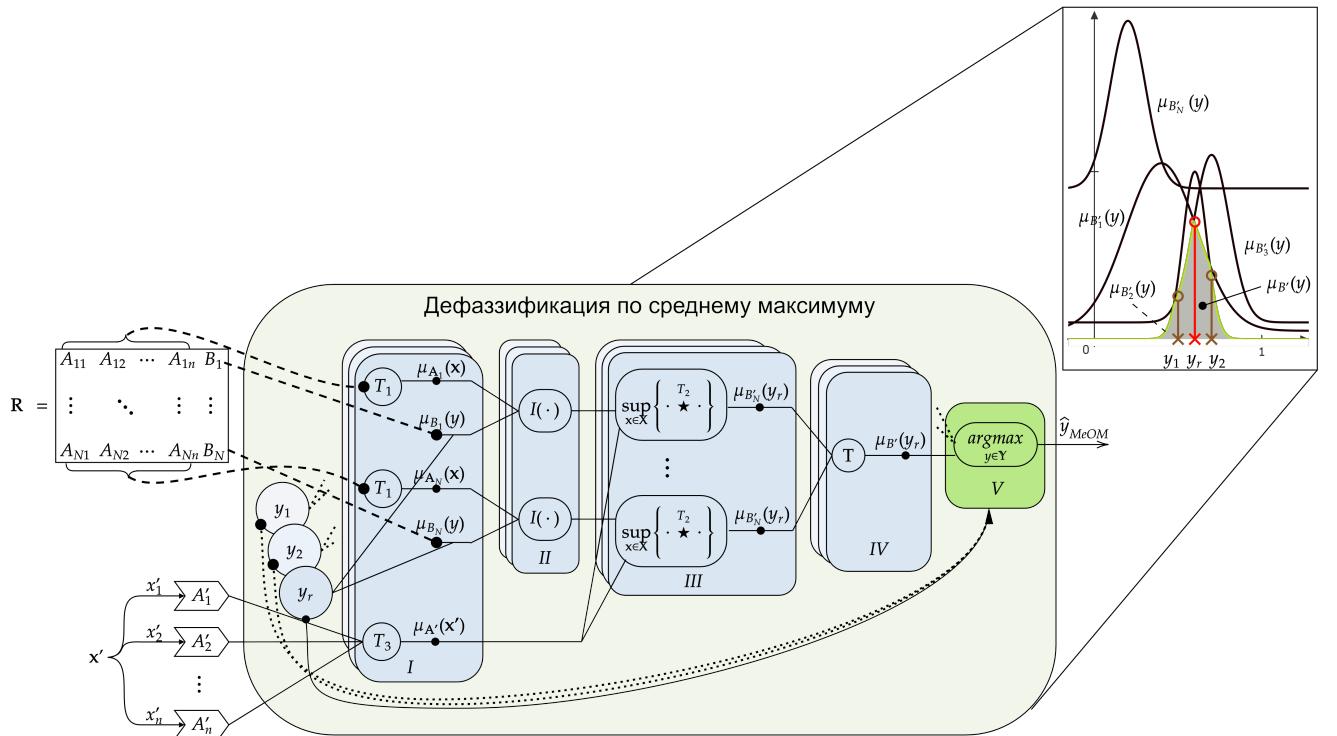


Рисунок 2.9 — Нейросетевая структура нечеткой модели с несинглтонной фазификацией и дефазификацией по среднему максимуму.

2.3 Альтернативный метод нечеткого вывода с полиномиальной вычислительной сложностью

Вычислительная сложность выражения композиционного правила (2.4) для вывода логического типа при использовании non-singleton фазификации составляет $O(|X_1| \cdot |X_2| \cdot \dots \cdot |X_n| \cdot |Y|)$ т.е. экспоненциальная. Это является актуальным препятствием использования нечеткого вывода логического типа совместно с фазификацией non-singleton.

2.3.1 Нечеткое значение истиности

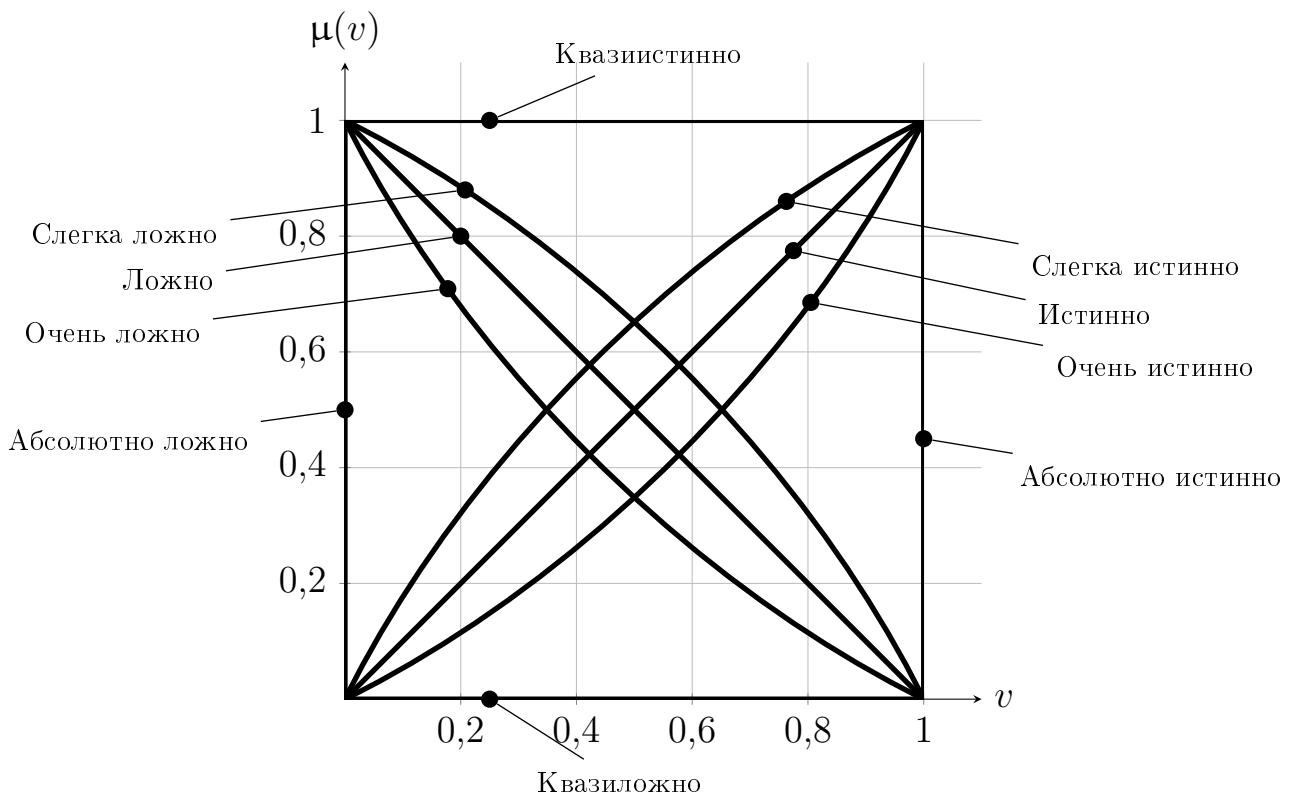


Рисунок 2.10 – Значения лингвистической переменной «истинность»

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», «слег-

ка 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.

На рисунке 2.10 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 которого оценивается истинность высказывания заданного нечетким множеством A .

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

$$\mu_{CP(A, A')}(v) = \sup_{\substack{\mu_A(x)=v \\ x \in X}} \{\mu_{A'}(x)\}. \quad (2.6)$$

Процедура вычисления истинности одного нечеткого множества относительно другого, согласно формуле (2.6), отражена на рисунке 2.11.

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

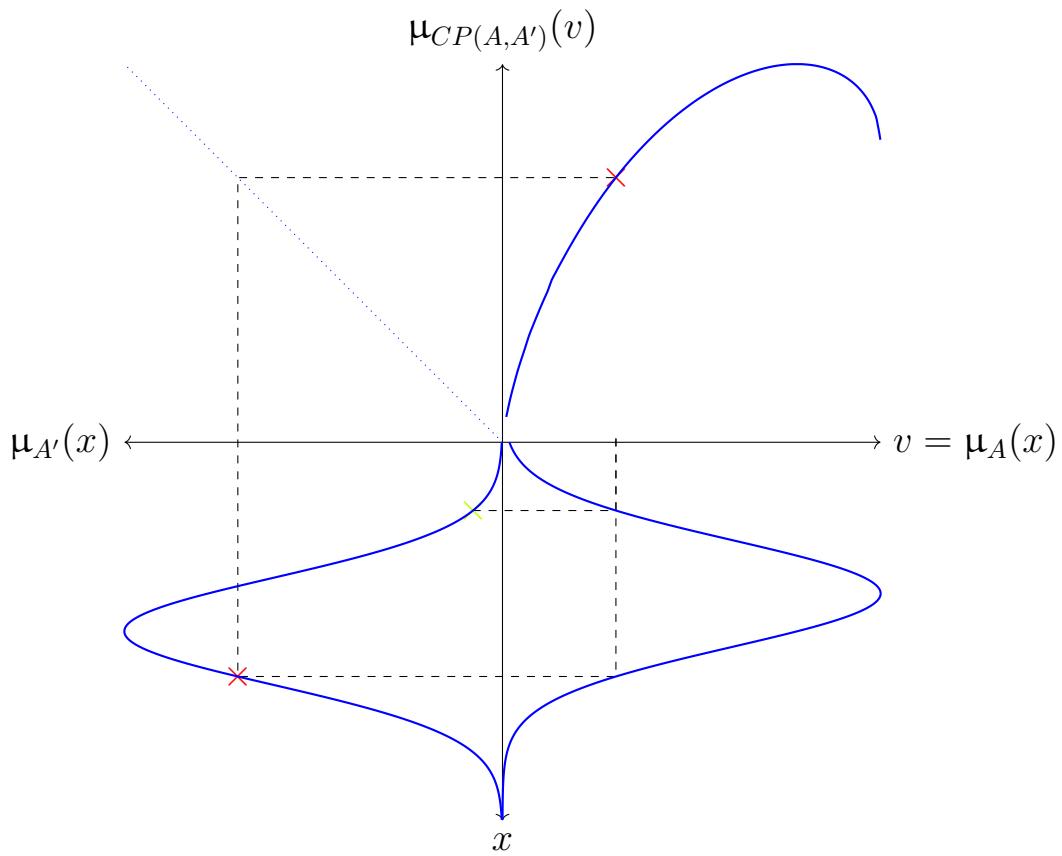


Рисунок 2.11 — Пример вычисления нечеткого значения истинности

- *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.

На рис. 2.12 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:

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

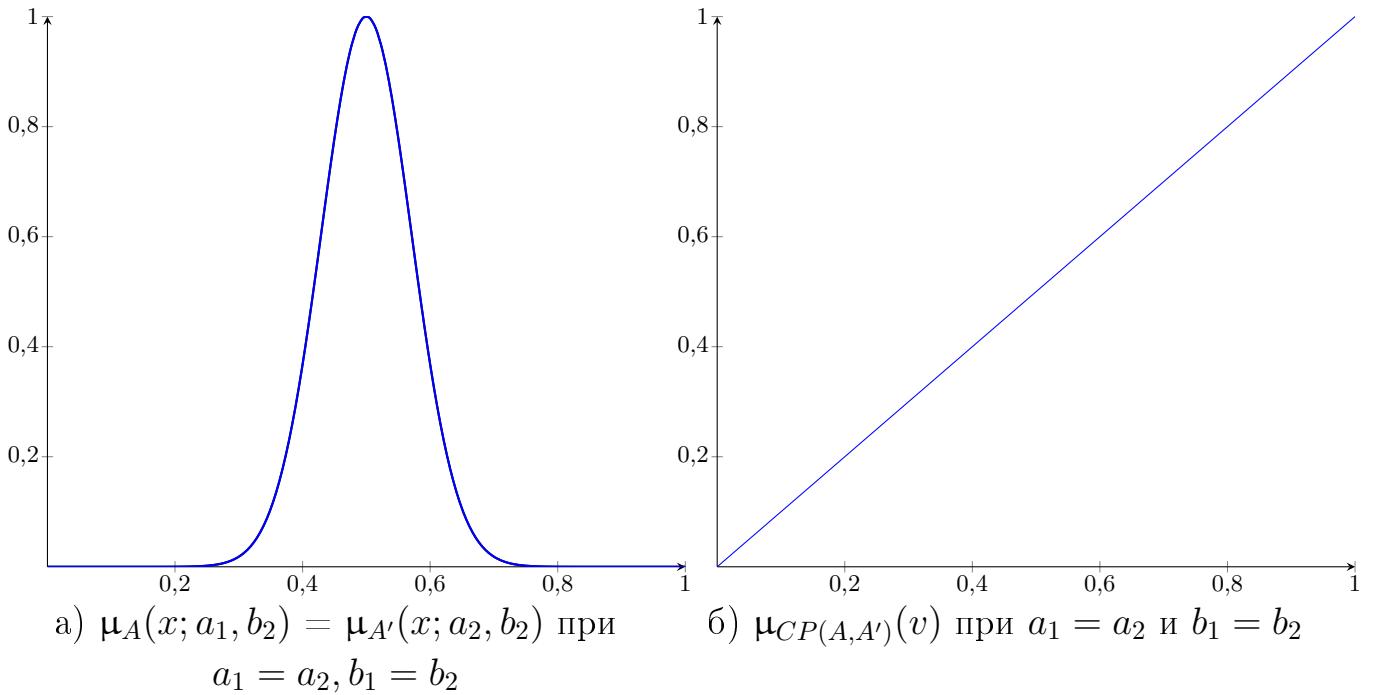


Рисунок 2.12 — Иллюстрация случая истинного отношения высказываний

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

- *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. 2.14 представлены графики функций принадлежности высказывания A' , включенного в A и функции принадлежности нечеткого

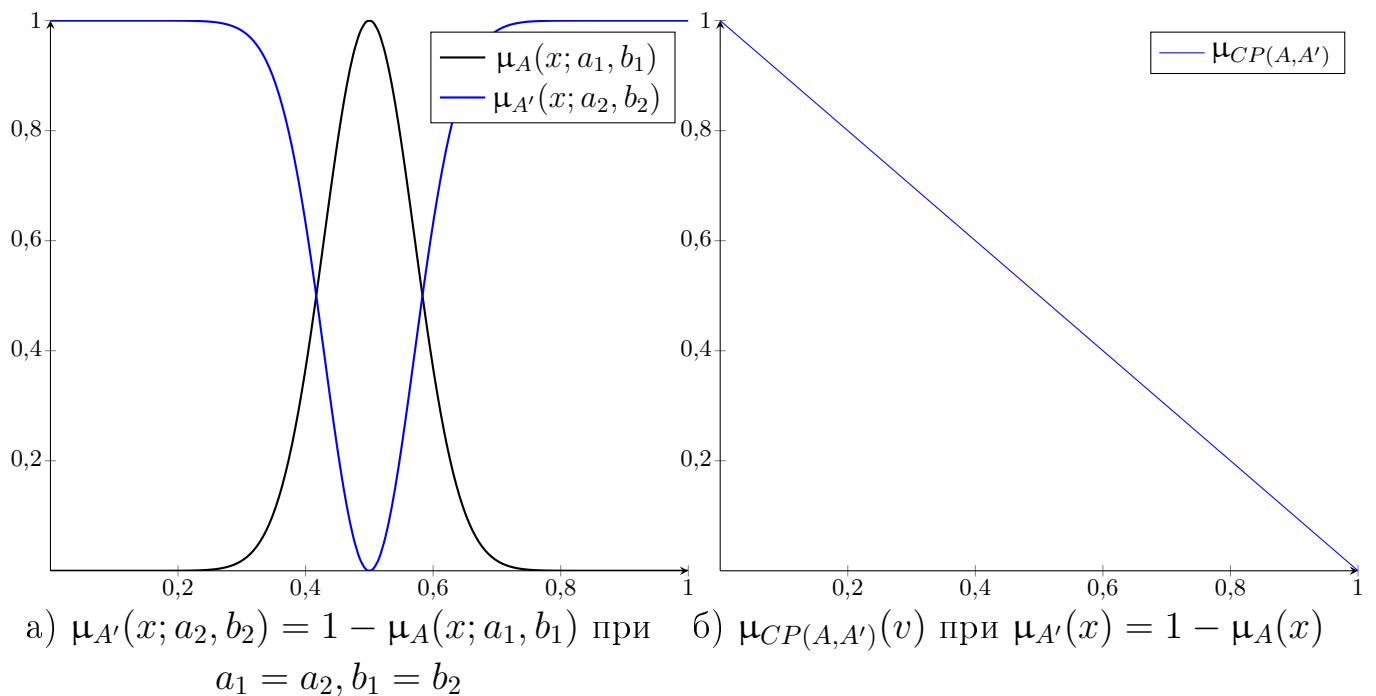


Рисунок 2.13 — Иллюстрация случая ложного отношения высказываний

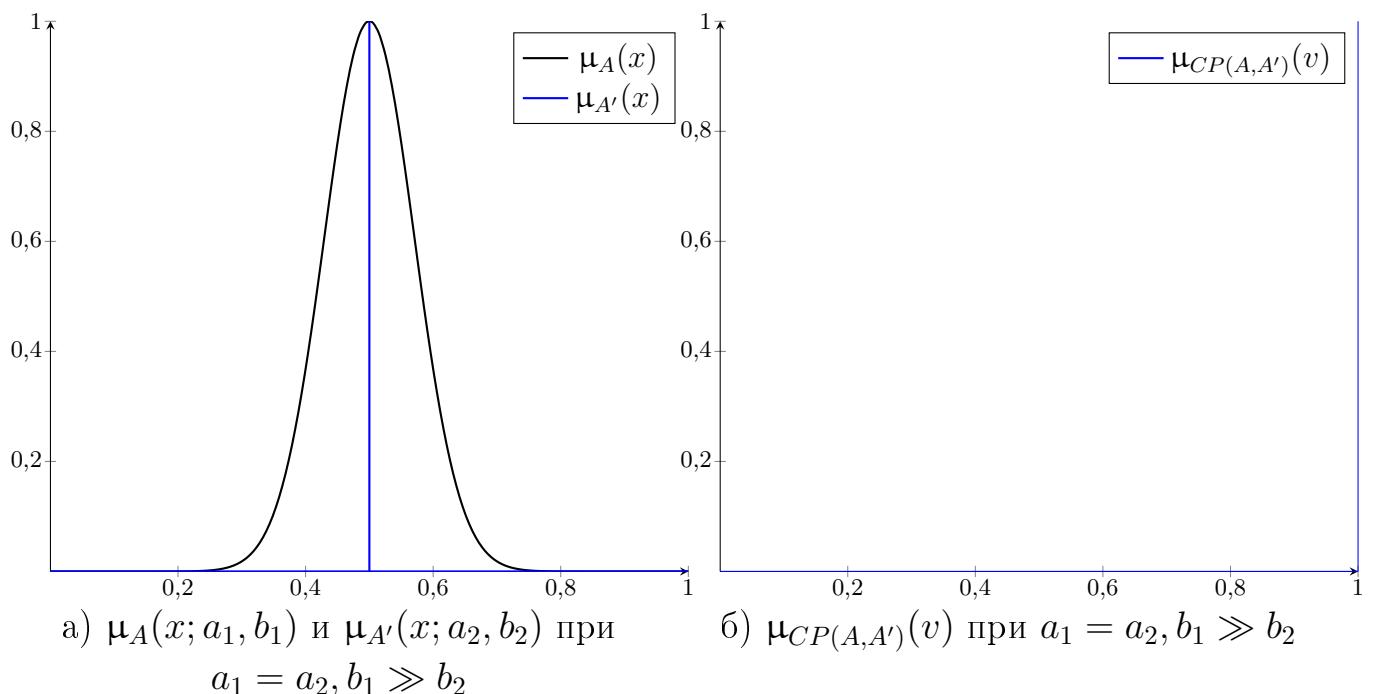


Рисунок 2.14 — Иллюстрация случая абсолютно истинного отношения высказываний

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.

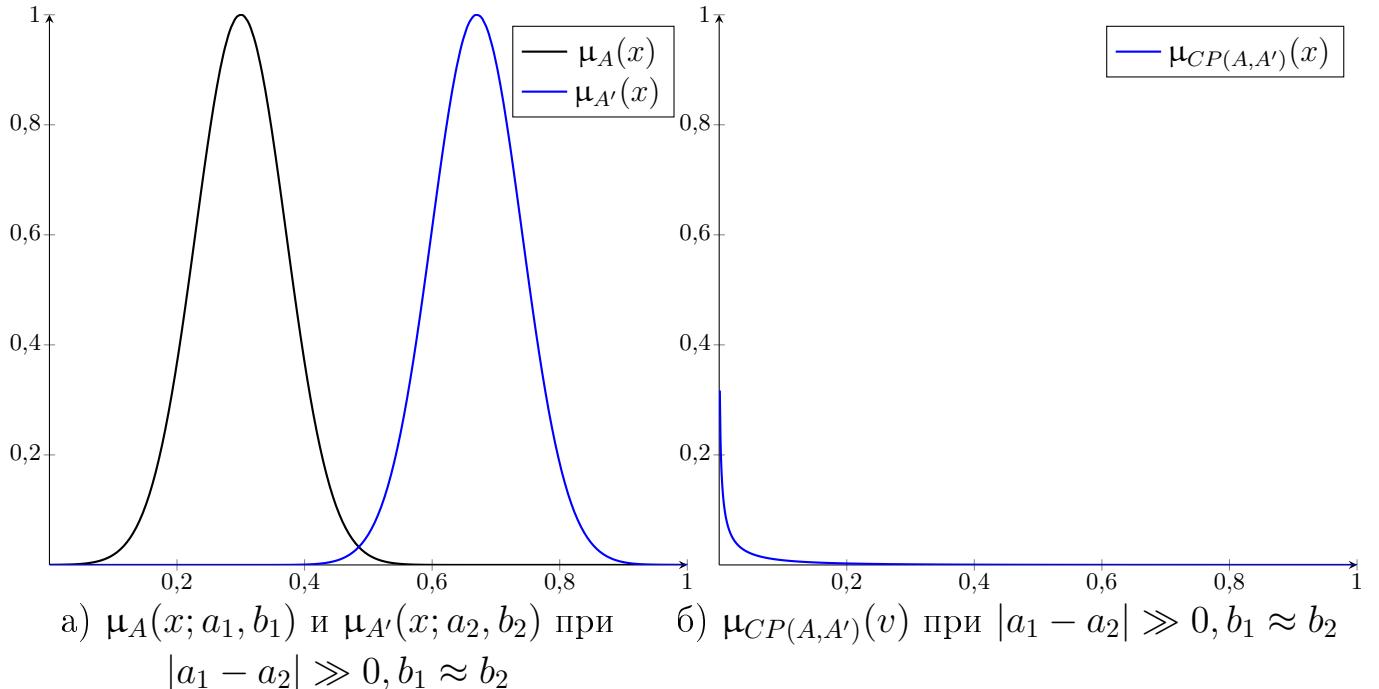


Рисунок 2.15 — Иллюстрация случая абсолютно ложного отношения высказываний

На рис. 2.15 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 .

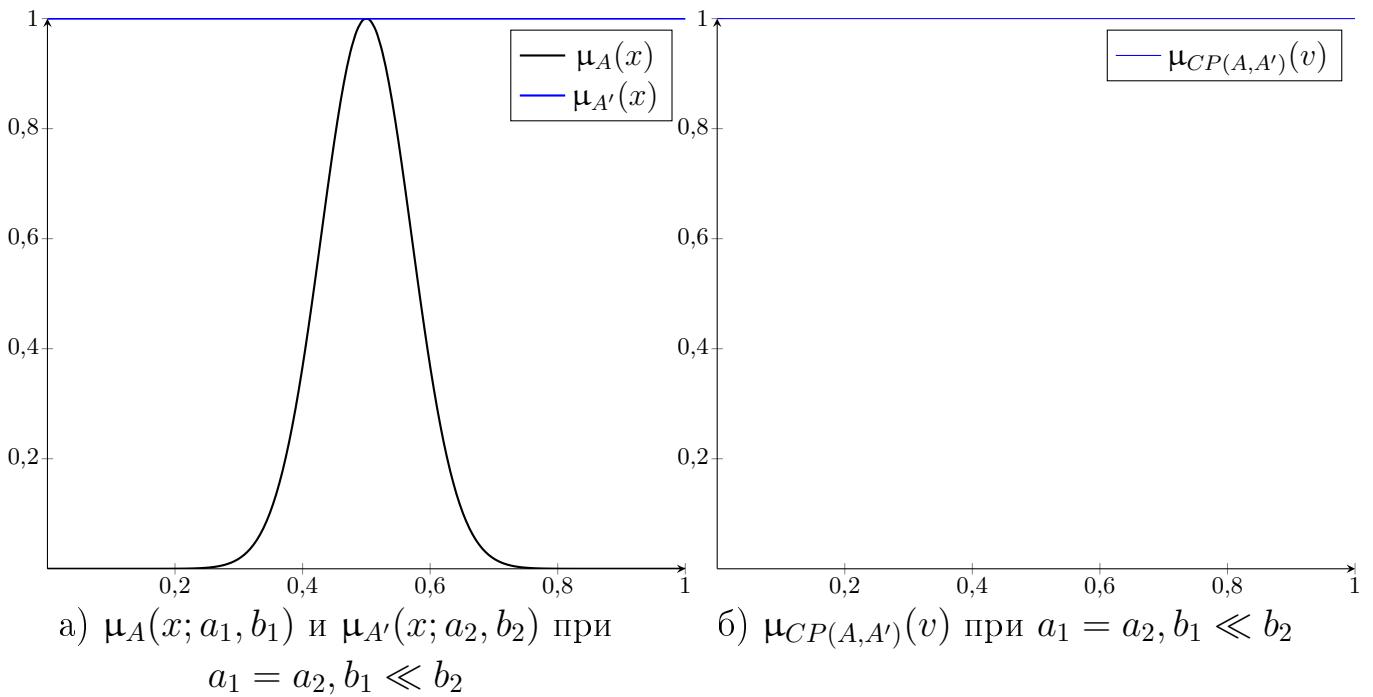


Рисунок 2.16 — Иллюстрация случая квазистинного отношения высказываний

На рис. 2.16 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:

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

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.

2.3.2 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 (2.6), 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

$$\begin{aligned} \mu_{CP(A,A')}(v) &= \max \left\{ e^{-\frac{((a-b\sqrt{-2 \ln v})-c)^2}{2d^2}}, e^{-\frac{((a+b\sqrt{-2 \ln v})-c)^2}{2d^2}} \right\} \\ &= \max \left\{ e^{-\frac{((a-c)-b\sqrt{-2 \ln v})^2}{2d^2}}, e^{-\frac{((a-c)+b\sqrt{-2 \ln v})^2}{2d^2}} \right\} \end{aligned} \quad (2.7)$$

2.3.3 Метод вывода на основе НЗИ

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 совместимости $CP(A_k, A')$ A_k по отношению к A' , причем A' рассматривается как достоверное:

$$\tau_{A_k|A'}(v) = \mu_{CP(A_k, A')}(v) = \sup_{\substack{\mu_{A_k}(x)=v \\ x \in X}} \{\mu_{A'}(x)\}. \quad (2.8)$$

Таким образом НЗИ отражает совместимость факта с посылкой в нечеткой форме.

Перейдем от переменной x к переменной v в выражении композиционного правила вывода (2.4), обозначив

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

Тогда для нечеткой системы с одним входом истинностное преобразование позволяет выполнить переход к новому виду выражения (2.4):

$$\mu_{B'_k}(y|x') = \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'}(v) \stackrel{T_2}{\star} I(v, \mu_{B_k}(y)) \right\}. \quad (2.9)$$

Это соответствует новой структуре правил в базе правил, отличную от канонических структур Заде и Такаги-Сугено:

$$\text{Если } \text{нзи есть ИСТИННО, то } y \text{ есть } B'_k. \quad (2.10)$$

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

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

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

$$\tau_{\mathbf{A}_k|\mathbf{A}'}(v) = \sup_{\substack{T_1 \\ i=1,n}} \left\{ \prod_{i=1}^n \mu_{A_{ki}}(x_i) \right\} \stackrel{T_3}{\star} \mu_{A'_k}(v). \quad (2.11)$$

Вместо выражения (2.11), НЗИ для n входов может быть вычислено как свертка НЗИ по каждомуциальному входу:

$$\tau_{\mathbf{A}_k|\mathbf{A}'}(v) = \tilde{T}_{\overline{i=1,n}} \tau_{A_{ki}|A'_k}, v \in [0,1], \quad (2.12)$$

где \tilde{T} - расширенная по принципу обобщения n -местная t -норма [6], которая определяется как

$$\tilde{T}_{\overline{i=1,n}} \tau_{A_{ki}|A'_k}(v) = \sup_{\substack{T_1 \\ i=1,n \\ v_i=v \\ (v_1, \dots, v_n) \in [0,1]^n}} \left\{ \prod_{i=1}^n \tau_{A_{ki}|A'_k}(v_i) \right\}. \quad (2.13)$$

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

$$\tilde{T}_{\overline{i=1,2}} \tau_{A_{ki}|A'_k}(v) = \sup_{\substack{v_1 T_1 v_2 = v \\ v_1, v_2 \in [0,1]}} \{ \tau_{A_{k1}|A'_1}(v_1) T_3 \tau_{A_{k2}|A'_2}(v_2) \}, v \in [0,1]. \quad (2.14)$$

Рекурсивная схема вычисления свертки НЗИ по формуле (2.13) иллюстрируется выражением:

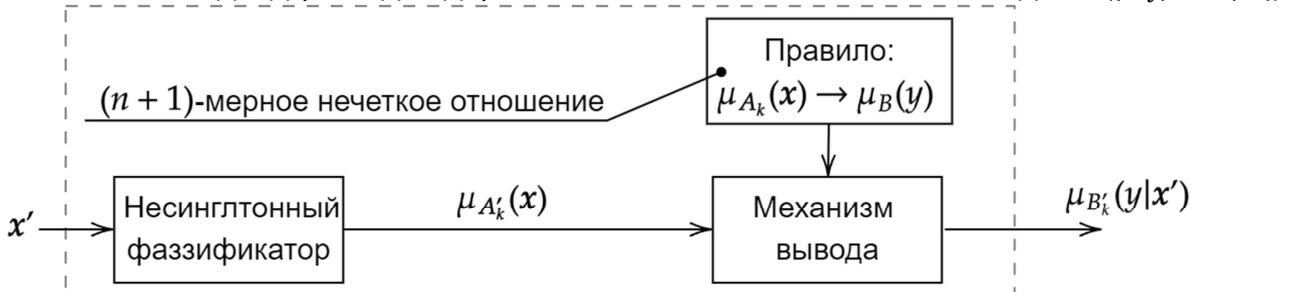
$$\tau_{A_k|A'}(v) = \tilde{T}_1 \tau_{A_{k1}|A'_1}(v_1) \quad (2.15)$$

$$= \left(\dots \left(\left(\mu_{CP(A_{k1}, A'_1)}(v_1) \tilde{T}_1 \mu_{CP(A_{k2}, A'_2)}(v_2) \right) \tilde{T}_1 \dots \right) \tilde{T}_1 \mu_{CP(A_{kn}, A'_n)} \right). \quad (2.16)$$

Тогда для системы с n входами выражения нечеткого вывода на основе НЗИ (2.9) примет вид:

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

Классический подход (метод Заде):



Альтернативный подход (предлагаемый метод):

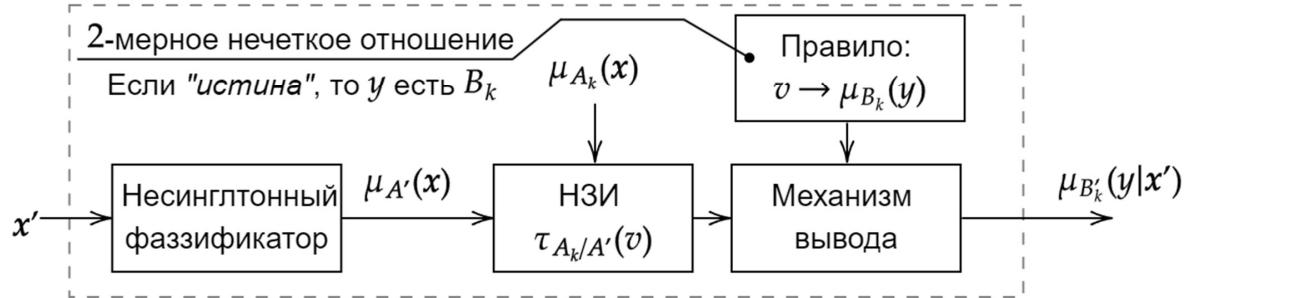


Рисунок 2.17 — Сравнение классической схемы нечеткого вывода и схемы нечеткого вывода на основе НЗИ

В формуле (2.17) 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.

Тогда, поскольку применение нового вида правила не зависит от количества входов в нечеткой системе, порядок функции временной сложности вычисления B'_k на основе выражения (2.17) составляет $O(n|V|^2 + |V| \cdot |Y|)$, где $V = [0; 1]$. Сравнение схем нечетких выводов в соответствии с соотношениями (2.4) и (2.17) представлены на рис. 2.17.

2.3.4 Вывод логического типа на основе нечеткого значения истинности

Для использованной в работе дефазификации по среднему максимуму вывод на основе нечеткого значения истинности выражается формулами:

$$\tau_{A_k|A'}(v) = \tilde{T}_{i=1,n} \left\{ \mu_{CP(A_{ki}, A'_{t-p+i})}(v_i) \right\}, k = \overline{1, N}, \quad (2.18)$$

$$\hat{y}_{t+h} = \arg \max_{y \in \mathbb{Y}} \left\{ \tilde{T}_{k=1}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'}(v) \stackrel{T_2}{\star} I(v, \mu_{A_{k,p+1}}(y)) \right\} \right\} \right\}. \quad (2.19)$$

Данным формулам соответствует сетевая структура нейро-нечеткой системы, изображенная на рисунке 2.18.

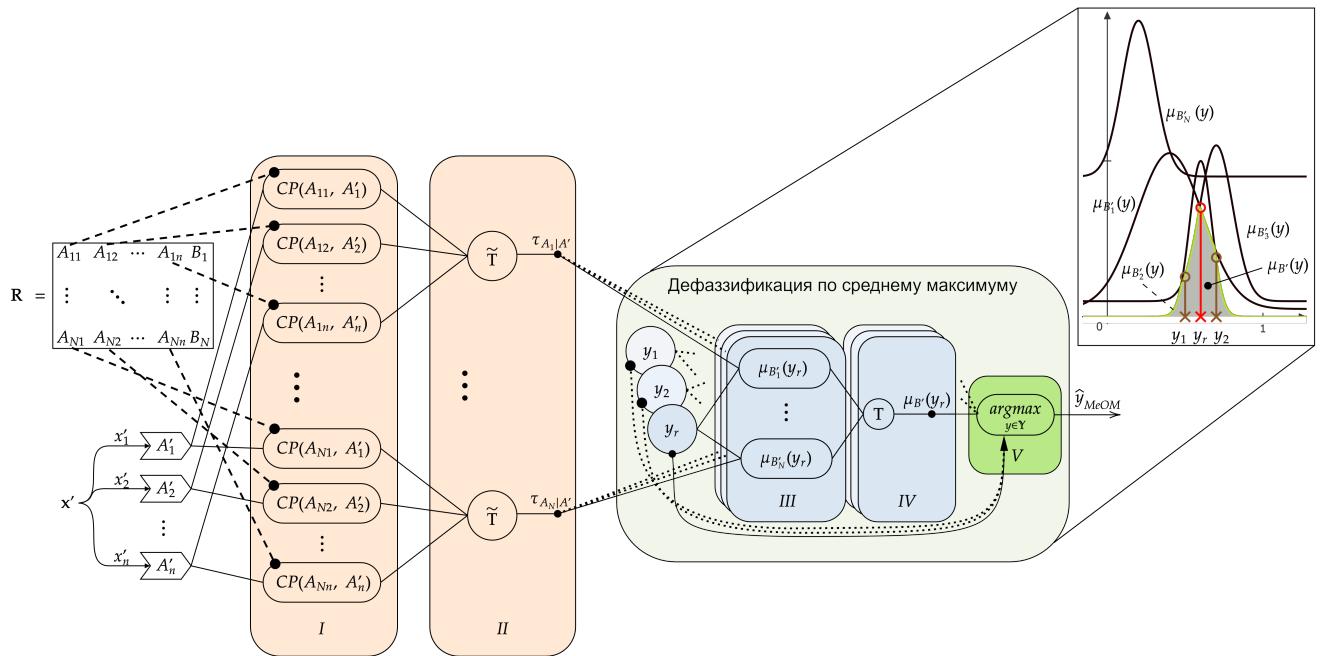


Рисунок 2.18 — Схема нейро-нечеткой системы с вычислением НЗИ и дефазификацией по среднему максимуму, а также пример работы процедуры дефазификации.

Как показано в [7], одна из возможностей упрощения процедуры вывода возникает, когда функции принадлежностей термов выходной лингвистической переменной 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 2.19.

Рассмотрим вычисление $\tau_{k,r}$ для различных категорий импликаций:

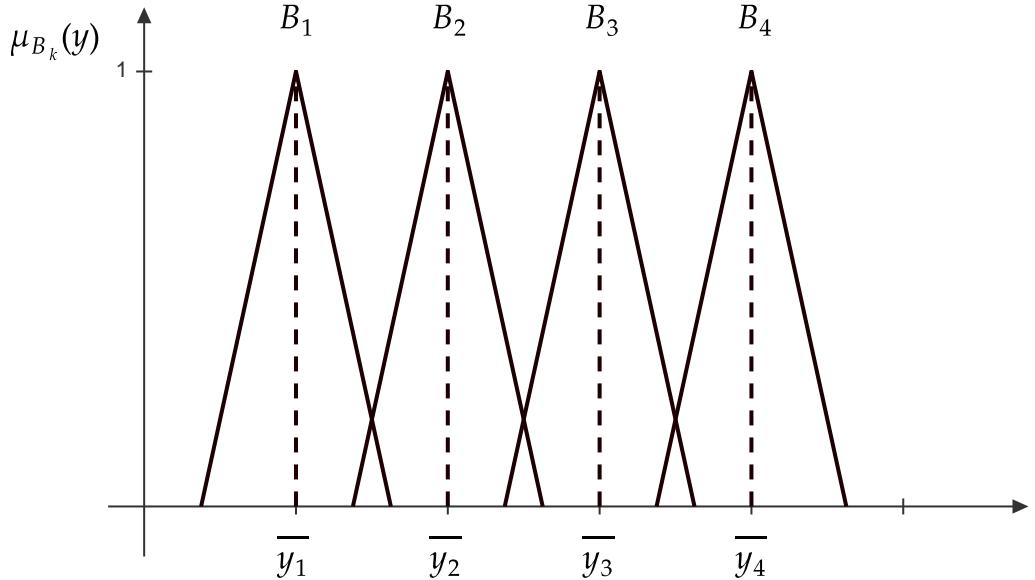


Рисунок 2.19 — Пример нечетких множеств, удовлетворяющих условию

$$\mu_{B_k}(y_r) = 0 \text{ для } y \neq r$$

— для S -импликации

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

— для R -импликации

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

— для Q -импликации

$$\hat{y}_{CoG} = \frac{\sum_{k=1}^N \bar{y}_k T_2 \left\{ \begin{aligned} & \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'}(v) \stackrel{T_2}{\star} \max(1-v, v) \right\} \times \\ & \times \sum_{\substack{r=1 \\ r \neq k}}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'}(v) \stackrel{T_2}{\star} (1-v) \right\} \right\} \end{aligned} \right\}}{\sum_{k=1}^N T_2 \left\{ \begin{aligned} & \sup_{v \in [0,1]} \left\{ \tau_{A_k|A'}(v) \stackrel{T_2}{\star} \max(1-v, v) \right\} \times \\ & \times \sum_{\substack{r=1 \\ r \neq k}}^N \left\{ \sup_{v \in [0,1]} \left\{ \tau_{A_r|A'}(v) \stackrel{T_2}{\star} (1-v) \right\} \right\} \end{aligned} \right\}}.$$

2.3.5 Свойства вывода типа Мамдани на основе нечеткого значения истинности

В [8] ослаблено ограничение на использование одной и той же T -нормы в формуле композиционного правила вывода в работах Менделя для случая вывода типа Мамдани, а также показано, что вывод по отдельному правилу в случае $T_2 = T_4 = T$ может быть записан через *меру возможности*:

$$\mu_{B'_k}(y) = \sup_{v \in [0;1]} \left\{ \tau_{A_k|A'}(v) \stackrel{T_2}{\star} (v \stackrel{T_4}{\star} \mu_{B_k}(y)) \right\} = \prod_{A_k|A'} \stackrel{T}{\star} \mu_{B_k}(y),$$

где $\prod_{A_k|A'} = \sup_{v \in [0;1]} \left\{ \tau_{A_k|A'}(v) \stackrel{T}{\star} v \right\}$.

Если при этом используется дефазификация *по центру сумм* (*CoS*) и T -норма Ларсена, то, как доказано в [8], результат дефазификации зависит от ширины гауссовой или треугольной функции принадлежности консеквента, тогда как дефазификация *по среднему центру* учитывает только параметр центра. Например, при использовании в качестве выходной ф. п. гауссовой функции $\mu_{B_k}(y) = \exp(-((y - \bar{y}_k)/\sigma_k)^2)$ формула дефазификации имеет вид:

$$\hat{y}_{CoS} = \frac{\int_{\mathbb{Y}} y \sum_{k=1}^N \prod_{A_k|A'} \stackrel{T_2}{\star} \mu_{B_k}(y)}{\int_{\mathbb{Y}} \sum_{k=1}^N \prod_{A_k|A'} \stackrel{T_2}{\star} \mu_{B_k}(y)} = \frac{\sum_{k=1}^N \prod_{A_k|A'} \bar{y}_k \sigma_k}{\sum_{k=1}^N \prod_{A_k|A'} \sigma_k}, \quad (2.20)$$

поскольку $\int_{-\infty}^{\infty} \mu_{B_k}(y) dy = \sigma_k \sqrt{\pi}$ и $\int_{-\infty}^{\infty} y \mu_{B_k}(y) dy = \bar{y}_k \sigma_k \sqrt{\pi}$.

2.4 Выводы по главе

1. Приведенное описание нечеткого вывода логического типа для нечеткой системы на основе правил в обобщенном виде может быть использована с несинглтонной фазификацией. Нечеткий вывод при несинглтонной фазификации позволяет учесть информацию о неопределенности входных данных, но имеет экспоненциальную вычислительную сложность.
2. Нечеткая модель прогнозирования временных рядов, полученная из нечеткой системы на основе правил, легла в основу разработанного

- метода прогнозирования, позволяющего адекватно учесть неопределенность значений временных рядов уникальных объектов или процессов.
3. При использовании несинглтонного способа фазификации показан эффект активации большего количества правил с ростом неопределенности. Для разработанный метод прогнозирования временных рядов с учетом неопределенности предложено использовать процедуру адаптивной фазификации и дефазификацию по среднему максимуму.
 4. Предложенный альтернативный метод нечеткого логического вывода на основе НЗИ обеспечивает полиномиальную вычислительную сложность. В этом методе вывода в процедуре вывода выделяется предварительный этап вычисления НЗИ, которые вычисляются по каждому входу независимо с последующей сверткой в единое истинностное пространство. Метод определяет новую структуру правил вида: «Если есть *ИСТИННО*, то y есть B_k ».

Глава 3. Разработка алгоритмической реализации метода прогнозирования временных рядов на основе нечетких моделей

3.1 Параллельный алгоритм свертки НЗИ

При программной реализации вычисления НЗИ и свертки НЗИ $\tau_{A_{ki}|A'_i}$ вычисление производится в точках расчетной сетки. Значение НЗИ по i -му входу в точке расчетной сетки v_j в данной работе обозначается $ftv_i[v_j]$ (ftv – *fuzzy truth value*). Расчетная сетка размера D_{ftv} задается на пространстве $\mathbb{V} = [0; 1]$ мощности $|\mathbb{V}|$.

Для нахождения свертки НЗИ по одному правилу можно составить параллельный алгоритм на основе формулы (2.15). Вычислительная сложность при параллельной реализации такого алгоритма составит $O(D_{ftv}^2 \cdot \log n)$. Значения $ftv_i[v_j]$ необходимо вычислить до запуска алгоритма свертки, что потребует сложности по памяти $O(D_{ftv} \cdot n)$. Кроме того параллельная свертка по формуле (2.15) потребует вычисления порядка $O(\log n)$ промежуточных попарных сверток в каждому входу. Узким местом в таком способе организации вычислений будет выступать ограниченность количества арифметико-логических модулей, между которыми распределяется работа в вычислительном устройстве.

В [7] разработан параллельный алгоритм для вычисления свертки НЗИ сразу по всем входам параллельно. Алгоритм опирается на допущение, что $T_1 = \min$, и имеет вычислительную сложность $O(D_{ftv} \cdot \log n)$.

Как проиллюстрировано на рисунке 3.1, данный алгоритм итеративно вычисляет значения $\tau_{A_k|A'}(v_j)$, продвигаясь от точки $v_j = 1$ к $v_j = 0$. На каждой j -й итерации вычисляются значения $ftv_i[v_j]$, которые агрегируются в одном вспомогательном массиве \max_ftv , что требует сложности по памяти $O(n)$ и реализует технику динамического программирования, избавляя 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

Algorithm 1 Алгоритм свертки НЗИ при $T_1 = \min$

Require: $ftv_i, i = \overline{1, n}$ — это $\tau_{A_{ki}|A'_i}$ дискретизированная в точках v_j

$\max_ftv[i] = 0;$

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

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

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

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

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

$r[i] \leftarrow ftv_i[v_j];$

else

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

end if

$ftv_reduced[v_j] \leftarrow T_3 \{r[i]\}_i;$

end for

return $ftv_reduced$

$T_1 = \min$ и согласно выражению из формулы (2.13)

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

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 .

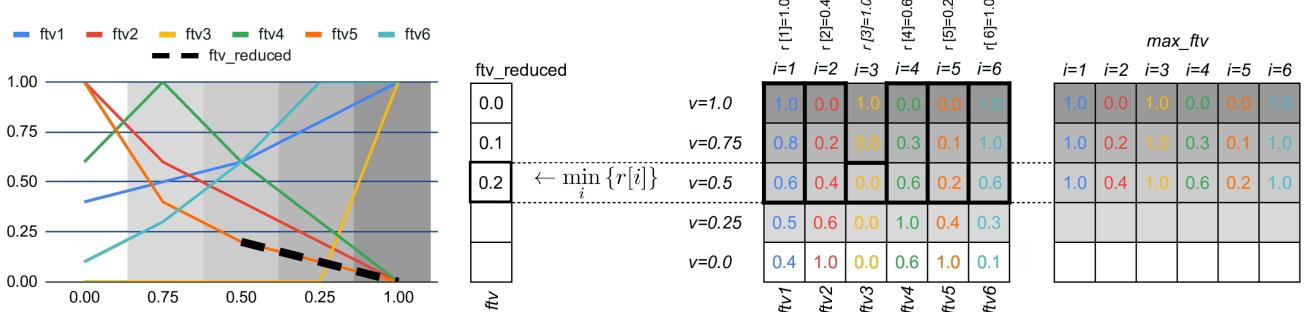


Рисунок 3.1 — Пример работы параллельного алгоритма свертки НЗИ при расчетной сетке состоящей из 5 точек.

3.2 Адаптация алгоритма PSO для построения базы правил.

Для построения базы правил у популярных в литературе типов нечетких систем Мамдани и Такаги-Сугено широко применяется алгоритм формирования правил на из данных [9]. В контексте задачи прогнозирования временных рядов широкое применение в публикациях получили различные методы, являющиеся развитием или продолжением данного алгоритма [10].

В этих подходах нечеткие множества, задающие значения в правилах, интерпретируются как термы лингвистических переменных. В контексте временных рядов это означает, что область значений изменяющейся во времени величины разбивается на соседствующие отсчеты (интервалы), каждому из которых соответствует терм лингвистической переменной.

1. Границы пространство значений временной последовательности $U = [D_{min} - \delta_1, D_{max} + \delta_2]$ задаются на основе минимального D_{min} и максимального D_{max} присутствующих в данных значений с некоторыми отступами δ_1 и δ_2 соответственно.
2. Пространство U разбивается на t равных отсчетов u_1, u_2, \dots, u_t .
3. Набор соответствующих отсчетам нечетких множеств формирует множество термов лингвистических переменных $y_t, t = \overline{1, T}$.
4. Затем каждому измеренному значению ставится в соответствие терм лингвистической переменной с наибольшей степенью истинности. Из последовательностей таких термов формируется набор правил — набор нечетких отношений.

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.

Чаще используются подходы построения базы правил, где значения термов в каждом правиле являются индивидуальным, то есть подбираются параметры нечетких множеств в правилах с уходом от лингвистической интерпретации.

Тогда каждое такое нечеткое множество в каждом правиле будет определяться индивидуальными значениями параметров функции принадлежности, а для задания всей базы правил требуется задание матрицы параметров θ_R

$$\theta_R = \begin{bmatrix} \theta_{\mu_{R_1}} \\ \vdots \\ \theta_{\mu_{R_N}} \end{bmatrix} = \begin{bmatrix} \theta_{\mu_{A_{11}}} & \theta_{\mu_{A_{12}}} & \dots & \theta_{\mu_{A_{1n}}} & \theta_{\mu_{B_1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \theta_{\mu_{A_{N1}}} & \theta_{\mu_{A_{N2}}} & \dots & \theta_{\mu_{A_{Nn}}} & \theta_{\mu_{B_N}} \end{bmatrix}.$$

При использовании гауссовых функций принадлежности с параметрами a и b для нечетких множеств в правилах матрица параметров может быть записана следующим образом:

$$\theta_R = \begin{bmatrix} \langle a_{\mu_{A_{11}}}, b_{\mu_{A_{11}}} \rangle & \langle a_{\mu_{A_{12}}}, b_{\mu_{A_{12}}} \rangle & \dots & \langle a_{\mu_{A_{1n}}}, b_{\mu_{A_{1n}}} \rangle & \langle a_{\mu_{B_1}}, b_{\mu_{B_1}} \rangle \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \langle a_{\mu_{A_{N1}}}, b_{\mu_{A_{N1}}} \rangle & \langle a_{\mu_{A_{N2}}}, b_{\mu_{A_{N2}}} \rangle & \dots & \langle a_{\mu_{A_{Nn}}}, b_{\mu_{A_{Nn}}} \rangle & \langle a_{\mu_{B_N}}, b_{\mu_{B_N}} \rangle \end{bmatrix}.$$

Для подбора параметров функций принадлежности в наборе правил используется оптимизация этих параметров на основе набора данных, а критерием оптимизации является минимизация метрики среднеквадратичной ошибки (RMSE - Root Mean Square Error). В качестве такого алгоритма выбран *метод роя частиц* (*Particle Swarm Optimization* - *PSO*), который приведен на рисунке Алгоритм 2. Частицы в PSO используют как собственный опыт, так и опыт соседей, сглаживая свою траекторию за счет использования промежуточного вектора скорости для каждой точки, дающего эффект инерции. 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 S , вес инерции ω и коэффициенты влияния текущего локального φ_l и глобального φ_g оптимума.

Для построения базы правил R на основе набора данных сперва данные были представлены в виде входных нечетких множеств посредством фазификации. Затем выполнялась оптимизация матрицы параметров ф. п. нечетких множеств базы правил θ_R :

$$\theta_R = \begin{bmatrix} \theta_{\mu_{R_1}} \\ \vdots \\ \theta_{\mu_{R_N}} \end{bmatrix} = \begin{bmatrix} \theta_{\mu_{A_{11}}} & \theta_{\mu_{A_{12}}} & \dots & \theta_{\mu_{A_{1n}}} & \theta_{\mu_{B_1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \theta_{\mu_{A_{N1}}} & \theta_{\mu_{A_{N2}}} & \dots & \theta_{\mu_{A_{Nn}}} & \theta_{\mu_{B_N}} \end{bmatrix}.$$

В качестве функций принадлежностей в работе используются гауссовые функции, которые задаются набором параметров:

$$\theta_{\mu_{A_{ki}}} = \langle a_{\mu_{A_{ki}}}, b_{\mu_{A_{ki}}} \rangle, \quad \theta_{\mu_{B_k}} = \langle a_{\mu_{B_k}}, b_{\mu_{B_k}} \rangle,$$

где a и b — математическое ожидание и среднеквадратичное отклонение гауссовой функции принадлежности.

Подбор параметров для фиксированного количества правил и заданной метрики осуществлялся с использованием адаптированного алгоритма оптимизации метод роя частиц (Particle Swarm Optimization, PSO), который приведен в алгоритме 2.

Минимально необходимый объем данных в наборе для обучения обычно оценивается формулой:

$$C \times N \times n \times \langle \text{количество параметров ф. п.} \rangle,$$

где C — коэффициент, зависящий от однородности распределения измеренных значений временного ряда и сложности закономерностей в исходных временных рядах.

3.3 Применение алгоритма оптимизации на основе метода роя частиц для вычисления дефазификации по среднему максимуму

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 градиентного 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.:

$$y^{(k+1)} = y^{(k)} - \alpha \frac{\mu'_{B'}(y^{(k)})}{\mu''_{B'}(y^{(k)})}, \quad (3.1)$$

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 (3.1) 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 (3.1) 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:

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

$$y^{(k+1)} = y^{(k)} + \alpha v^{(k+1)}, \quad (3.2)$$

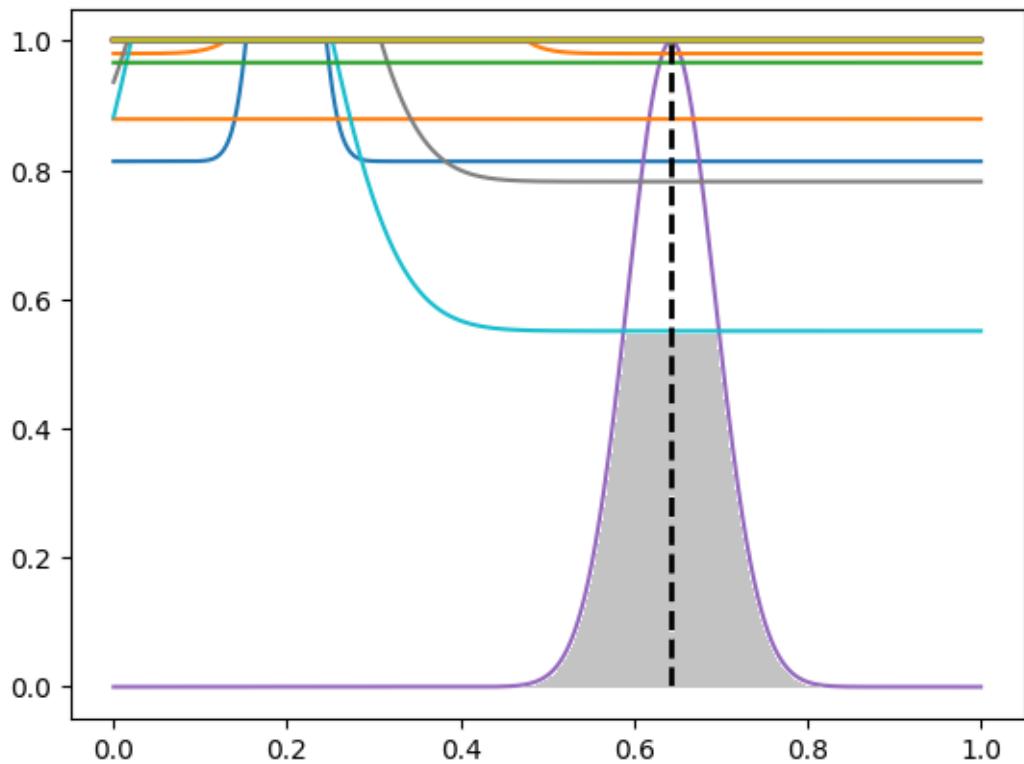


Рисунок 3.2 — Ситуация необходимости нахождения среднего максимума при дефазификации агрегированной функции принадлежности $\mu_{B'}(y)$.

где ω — параметр скользящего среднего значения момента.

Необходимость усреднения максимумов при использовании метода дефазификации Mean of Maximum возникает в ситуации изображенной на рисунке 3.2, 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 (3.2) для данного метода формулы

шага оптимизации можно составить следующим образом:

$$\begin{aligned}
 v_i^{(k+1)} &= \omega v_i^{(k)} + \\
 &+ \alpha_l \left(r_l \left(y_i^{best} - y_i^{(k)} \right) + (1 - r_l) \left(-\frac{\mu'_{B'}(y_i^{(k)})}{\mu''_{B'}(y_i^{(k)})} \right) \right) + \\
 &+ \alpha_g \left(r_g \mu_{B'}(y^{best}) (y^{best} - y) + (1 - r_g) (1 - \mu_{B'}(y^{best})) (\hat{y}_{CA} - y^{(k)}) \right), \\
 y^{(k+1)} &= y^{(k)} + v^{(k+1)},
 \end{aligned} \tag{3.3}$$

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.

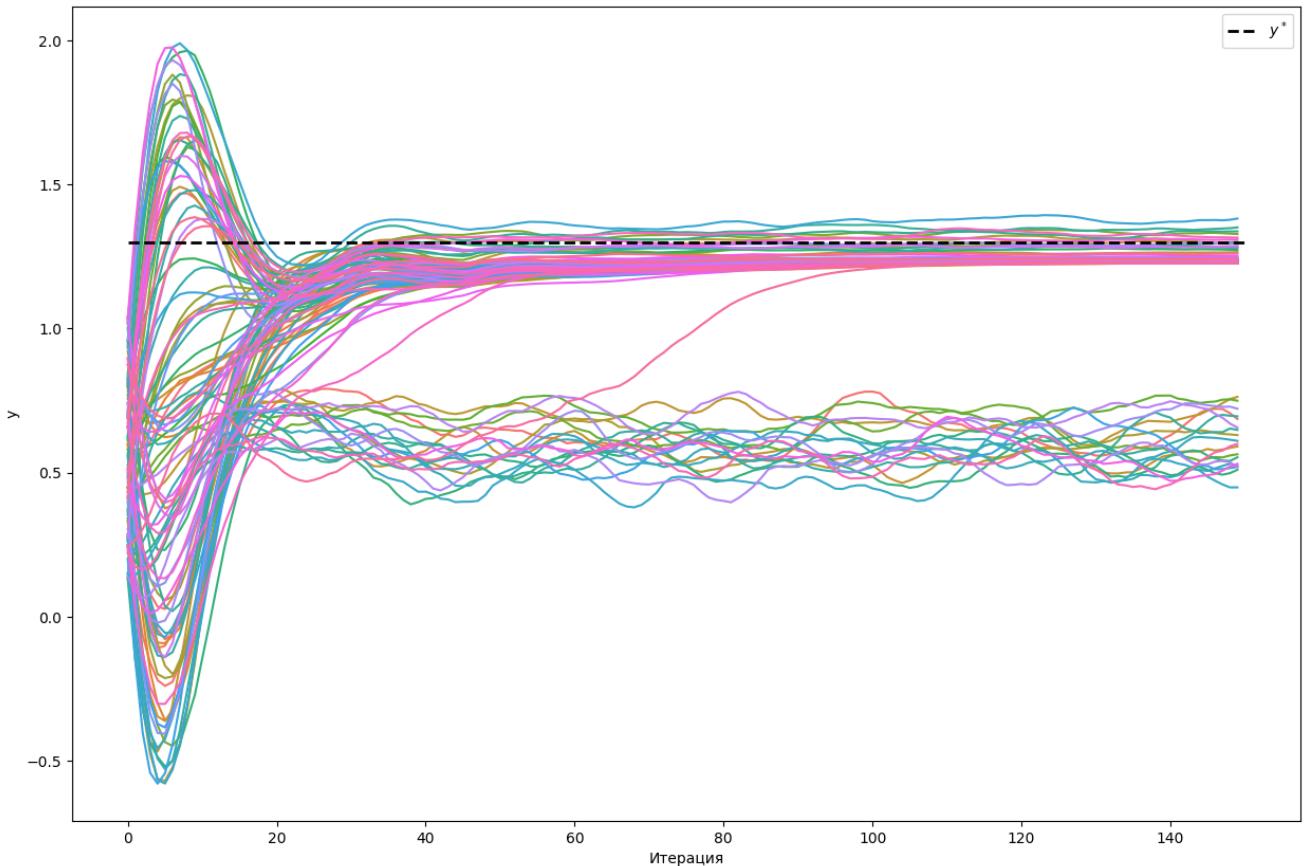


Рисунок 3.3 — Динамика движения точек популяции при работе алгоритма глобальной оптимизации Gradient-aware PSO.

Компонента после коэффициента α_l в формуле (3.3) обеспечить разнообразие среди точек популяции, достигших максимума $\mu_{B'}(y_i^{(k)})$. График работы

такого алгоритма PSO приведен на рисунке 3.3. 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) использовать численное интегрирование, например, методом Симпсона.

3.4 Процедура метода прогнозирования временных рядов на основе разработанной нечеткой модели

Разработанный метод прогнозирования временных рядов на основе нечеткой модели включает две основные процедуры: построение базы правил (обучение модели) и прогнозирование будущих значений временного ряда. Обе процедуры опираются на единую схему предварительной обработки входных данных и нечеткого вывода, но отличаются целями и используемыми алгоритмами оптимизации.

Процедура построения базы правил начинается с подготовки обучающего набора данных путем нарезания временного ряда на перекрывающиеся окна размером p , где p — параметр запаздывания, определяющий количество входов нечеткой модели. Каждое окно представляет собой вектор из p последовательных значений, которые служат входами в нечеткую модель, а целевым значением является следующее значение временного ряда. На этапе адаптивной фазификации каждое окно преобразуется в набор входных нечетких множеств, которые отражают степень принадлежности значений окна к различным нечетким множествам. Адаптивность фазификации заключается в автоматической подстройке параметров функций принадлежности в соответствии с распределением данных. Затем выполняется оптимизация параметров базы правил с использованием алгоритма метода роя частиц (PSO), который минимизирует выбранную целевую метрику (MAE, MAPE, sMAPE, MSE или RMSE) на обучающем наборе данных. Процесс оптимизации включает вычисление нечеткого вывода для каждого фазифицированного окна, агрегирование результатов по правилам базы и дефазификацию с использованием адаптированного алго-

ритма Gradient-aware PSO для методов дефазификации, требующих поиска экстремумов.

Algorithm 3 Построение базы правил нечеткой модели для прогнозирования временных рядов

Require:

X_{train} — обучающий набор временных рядов

p — размер окна запаздывания (количество входов)

N — количество правил в базе правил

$target_metric$ — целевая метрика оптимизации (MAE, MAPE, sMAPE, MSE, RMSE)

$impl_type$ — тип импликации (Лукасевича, Рейхенбаха, Ягера, Клини-Динса)

$rules_pso_sz$ — количество частиц в PSO для оптимизации базы правил

$rules_pso_cnt$ — количество итераций алгоритма PSO для оптимизации базы правил

$defuz_pso_sz$ — количество частиц в Gradient-aware PSO для дефазификации

$defuz_pso_cnt$ — количество итераций Gradient-aware PSO для дефазификации

- 1: $\{X'_{train}\}_{l=1}^{M'_{train}} \leftarrow \text{SlidingWindow}(X_{train}, p)$ ▷ Нарезание на окна запаздывания
 - 2: $\{\mathbf{A}'_{train}\}_{l=1}^{M'_{train}} \leftarrow \text{AdaptiveFuzzification}(\{X'_{train}\}_{l=1}^{M'_{train}})$ ▷ Адаптивная фаззификация входов
 - 3: $\{y_{train}\}_{l=1}^{M'_{train}} \leftarrow \text{ExtractTargets}(X_{train}, p)$ ▷ Извлечение целевых значений
 - 4: $\mathbf{R} \leftarrow \text{OptimizeRuleBase}(\{\mathbf{A}'_{train}\}_{l=1}^{M'_{train}}, \{y_{train}\}_{l=1}^{M'_{train}}, N)$, ▷ Оптимизация параметров базы правил
 - 5: **Возврат \mathbf{R}** ▷ Построенная база правил
-

Процедура прогнозирования временных рядов с использованием построенной базы правил структурно подобна процедуре обучения на этапах подготовки данных, но вместо оптимизации параметров выполняется пакетное вычисление прогнозных значений. На входе процедура получает временной ряд для прогноза, построенную базу правил и те же параметры фаззификации, которые

использовались при обучении. Нарезание временного ряда на окна осуществляется с тем же размером запаздывания p , что обеспечивает согласованность между этапом обучения и прогнозирования. Адаптивная фазификация применяется к входным окнам, преобразуя их в нечеткие множества, которые интерпретируются моделью. Затем для каждого фазифицированного окна выполняется нечеткий вывод с использованием построенной базы правил, что приводит к вычислению агрегированной выходной функции принадлежности. Дефазификация этой функции с помощью адаптированного алгоритма Gradient-aware PSO преобразует нечеткий результат в четкое числовое значение прогноза.

Algorithm 4 Прогнозирование временного ряда на основе построенной базы правил

Require:

X — временной ряд для прогноза

\mathbf{R} — построенная база правил

p — размер окна запаздывания (количество входов)

$impl_type$ — тип импликации (Лукасевича, Рейхенбаха, Ягера, Клини-Динса)

$defuz_pso_sz$ — количество частиц в Gradient-aware PSO для дефазификации

$defuz_pso_cnt$ — количество итераций Gradient-aware PSO для дефазификации

1: $\{X'\}_{l=1}^{M'} \leftarrow \text{SlidingWindow}(X, p)$ \triangleright Нарезание на окна запаздывания

2: $\{\mathbf{A}'\}_{l=1}^{M'} \leftarrow \text{AdaptiveFuzzification}(\{X'\}_{l=1}^{M'})$ \triangleright Адаптивная фазификация входов

3: $\{\hat{y}\}_{l=1}^{M'} \leftarrow \text{FuzzyInferenceBatch}(\{\mathbf{A}'\}_{l=1}^{M'}, \mathbf{R}, impl_type, defuz_pso_sz, defuz_pso_cnt)$ \triangleright Пакетный нечеткий вывод

4: **Возврат** $\{\hat{y}\}_{l=1}^{M'}$ \triangleright Прогнозные значения

Ключевой связью между двумя процедурами является адаптивная фазификация, параметры которой должны быть согласованы между этапами обучения и прогнозирования. На этапе обучения параметры фазификации адаптируются на основе статистических свойств обучающего набора данных (математическое ожидание, дисперсия, минимальное и максимальное значения), и эти же параметры впоследствии применяются при фазификации

входных окон для прогнозирования без каких-либо изменений. Это обеспечивает корректную интерпретацию входных данных нечеткой моделью и согласованность между фазами обучения и применения.

Алгоритм OptimizeRuleBase в процедуре построения базы правил осуществляет итеративный поиск оптимальных параметров функций принадлежности нечетких множеств в правилах, минимизируя выбранную метрику на обучающем наборе. Количество правил N фиксируется заранее и определяет структуру базы правил. Для каждого набора параметров, генерируемых алгоритмом PSO, вычисляется нечеткий вывод с использованием выбранного типа импликации для всех обучающих примеров, что требует дефазификации промежуточных результатов с использованием Gradient-aware PSO. Оптимизируемые параметры включают центры и ширины гауссовых функций принадлежности для каждого входа и выхода каждого правила, образуя матрицу θ_R , размерность которой зависит от количества правил N , количества входов p и выбранной параметризации функций принадлежности.

Алгоритм FuzzyInferenceBatch в процедуре прогнозирования применяет уже оптимизированную базу правил R ко всем входным окнам прогнозируемого временного ряда, производя массивно-параллельные вычисления нечеткого вывода с использованием технологии CUDA. Параллельный алгоритм свертки НЗИ, описанный в предыдущем разделе, обеспечивает эффективное агрегирование результатов правил с вычислительной сложностью $O(D_{ftv} \cdot \log n)$, позволяя обрабатывать большие объемы данных в реальном времени. Выбранный тип импликации и метод дефазификации остаются неизменными между обучением и прогнозированием, обеспечивая консистентность вывода.

Проведение параллельных вычислений нечеткого вывода для множества окон временного ряда одновременно на GPU позволяет достичь значительного ускорения прогнозирования по сравнению с последовательной реализацией. Каждое окно обрабатывается независимо, что делает возможным эффективное распределение вычислений между нитями и блоками CUDA. Таким образом, разработанная двухэтапная процедура (обучение и прогнозирование) представляет собой целостную систему, в которой этап обучения готовит модель на основе исторических данных, а этап прогнозирования применяет эту модель к новым данным, используя согласованные параметры и алгоритмы.

3.5 Выводы по главе

1. Предложенный параллельный алгоритм свертки НЗИ обеспечивает линейную зависимость сложности вычислений $O(D_{ftv} \cdot \log n)$ от размера расчетной сетки D_{ftv} и линейную сложность по памяти $O(n)$ от числа выходов n .
2. Адаптированный алгоритм метода роя частиц (PSO) позволяет организовать построение базы правил на основе набора обучающих данных за счет подбора оптимальных значений параметров ф. п. нечетких множеств.
3. Для вычисления дефазификации по среднему максимуму адаптирован алгоритм метод роя частиц, учитывающий информацию о градиенте в точках оптимизируемой функции (Gradient-aware PSO). Учет градиента позволяет ускорить сходимость процедуры оптимизации.
4. Верхнеуровневое описание процедур построения базы правил и предсказания прогноза состоит из этапов: составления набора окон запаздывания из временных рядов, применения адаптивной фазификации, оптимизации базы правил или пакетного вычисления прогнозного значения для построенной базы правил.

Глава 4. Разработка информационной технологии прогнозирования временных рядов, прототипа ее программной реализации на основе нечеткой модели и оценка работоспособности

4.1 Библиотека с параллельной реализацией нечеткого вывода на основе технологии CUDA

Реализация информационной технологии прогнозирования для предложенного метода на основе нечеткой модели, согласно разработанным алгоритмам, является композицией последовательности этапов вычислений различной сложности. Этап, содержащий наибольший объем вычислений, включает в себя процедуру нечеткого вывода. Его реализация потребует организации эффективного использования ресурсов аппаратной платформы для организации вычислений по разработанным алгоритмам.

Согласно схемам 2.18, изображенными во 2-м 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** [11; 12], 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 программной 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 НРС-приложениях, упрощая разработку масштабируемых кодов при сохранении производительности на постоянно развивающемся оборудовании.

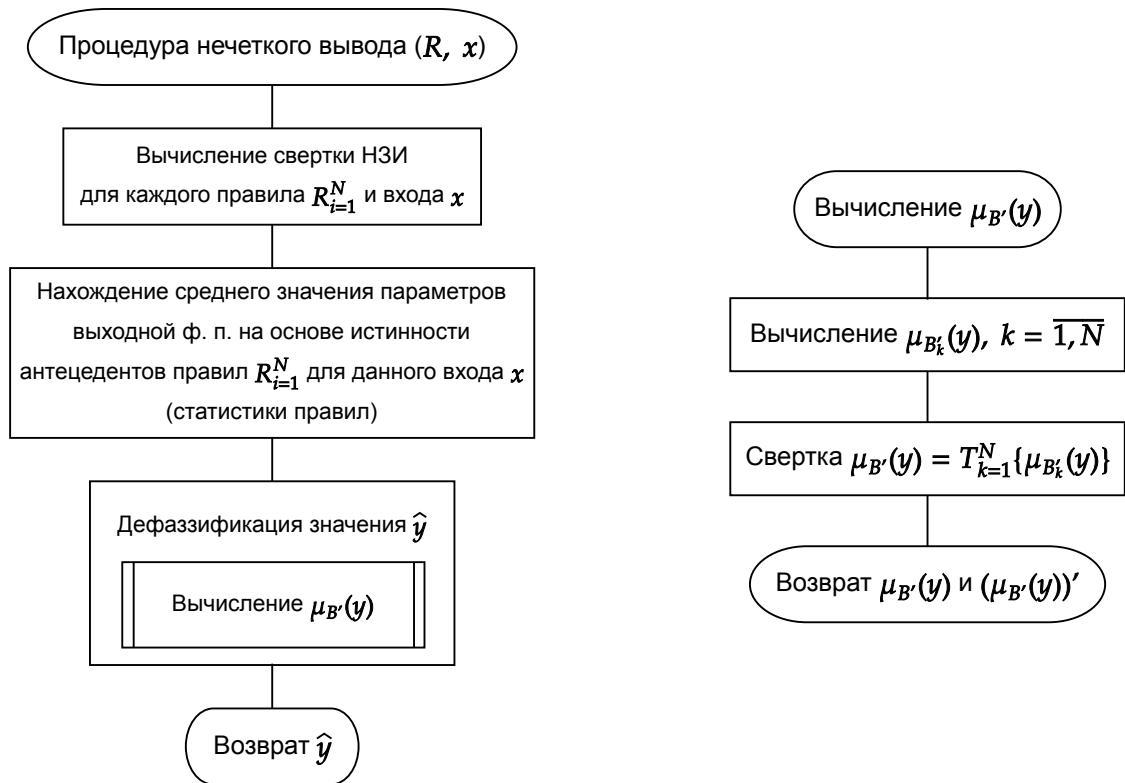


Рисунок 4.1 — Блок схема процедуры нечеткого логического вывода на основе нечеткого значения истинности.

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 (распределяемых 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 потоком мультипроцессоре. 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:

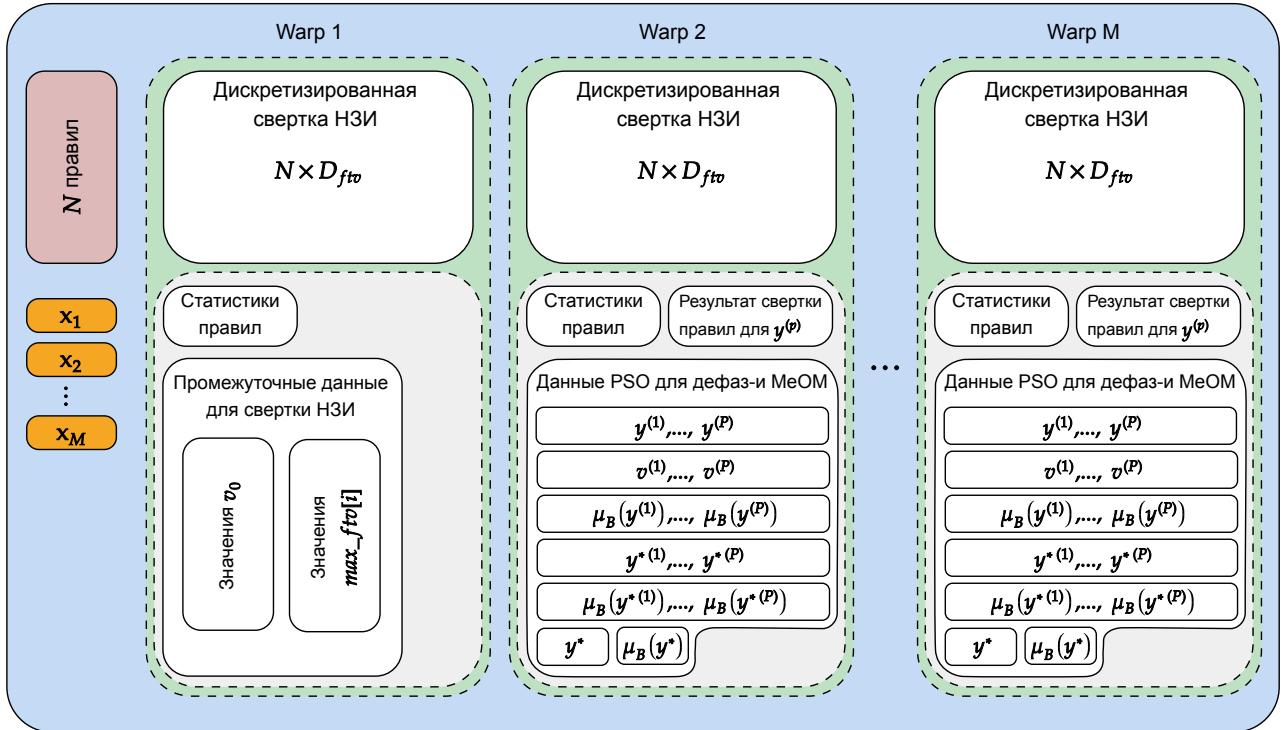


Рисунок 4.2 — Карта разделяемой памяти одного CUDA-блока нечеткого логического вывода на основе нечеткого значения истинности при использовании дефазификации МеОМ.

- параметры ф. п. N правил при n входах (2 параметра для гауссовой ф-и): (`sizeof float`) \times $(n + 1) \times N \times 2$;
- параметры ф. п. входного экземпляра данных при n входах (2 параметра для гауссовой ф-и): (`sizeof float`) \times $(n + 1) \times 2$;
- значения свертки НЗИ для N правил при размере расчетной сетки равному $D_{f\text{tv}}$: (`sizeof float`) $\times D_{f\text{tv}} \times N$ и промежуточные данные:
 - индексы координат расчетной сетки с максимальным значением НЗИ каждому из N правил и n входов: (`sizeof char`) $\times n \times N \times 2$;
 - максимальные значения НЗИ на текущей итерации свертки НЗИ по каждому из N правил и n входов: (`sizeof float`) $\times n \times N \times 2$;
- 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 4.2.

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 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.

4.1.1 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 (2.7) 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 (2.7) можно представить в виде композиции функций

$$\mu_{CP(A,A')}(v) = \exp(-z(v)^2), \quad (4.1)$$

и

$$z(v) = \alpha \pm \beta \varphi(v). \quad (4.2)$$

Для иллюстрации условий возникновения описанных вырождений функции (2.7) графики функций (4.1) и (4.2) при $\alpha = \beta = 1$ приведены на рисунках 4.3 и 4.4 соответственно. Из рисунка 4.4 видно, что вырождение выражения (4.1) в близкий к асимптоте вид возможно при очень больших или очень малых значениях α и β . Данное наблюдение проиллюстрировано на рисунке 4.5.

В ситуациях, изображенных на рисунках 4.5а, 4.5б, 4.5г и 4.5д, 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. 4.5а и 4.5д)

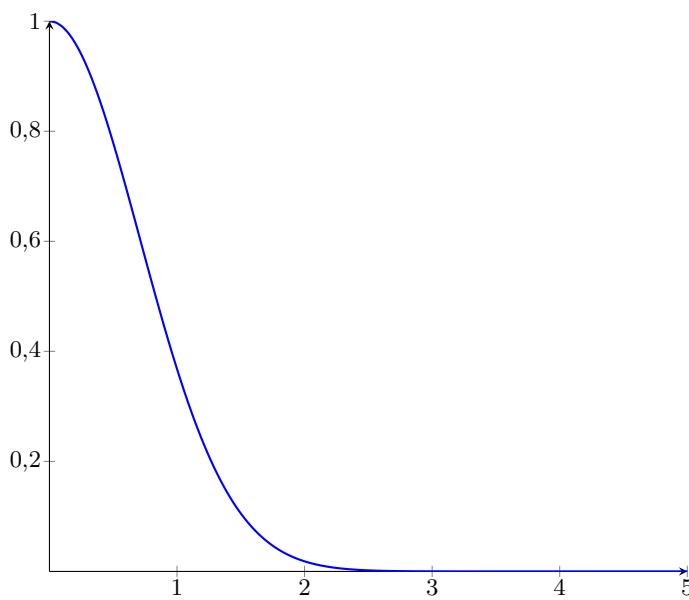


Рисунок 4.3 — График функции $\exp(-z^2)$

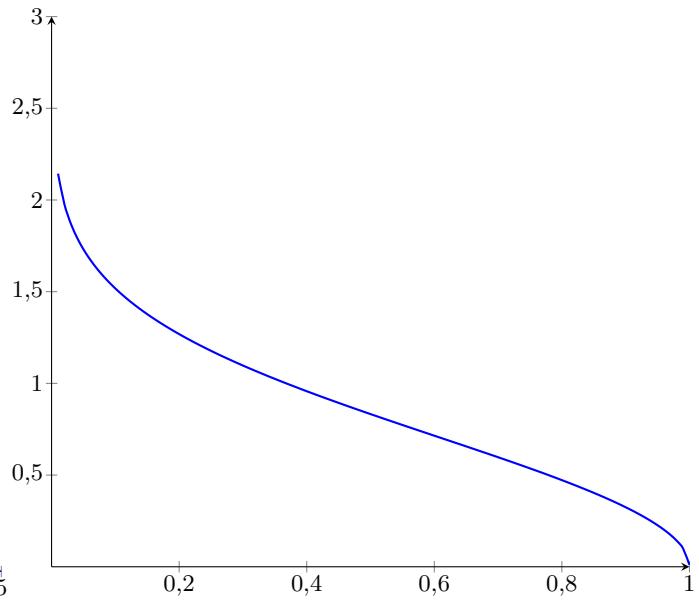


Рисунок 4.4 — График функции $\varphi(v) = \sqrt{-\ln v}$

или ф. п. НЗИ в точке v_0 имеет значение сильно отличающееся от значений в 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 4.3 видно, что максимальное значение выражения (4.1), равное 1, достигается при $z = 0$, а из рисунка 4.4 видно, что функция (4.2) 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 (2.7) 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 (2.7) принимает значение 1.

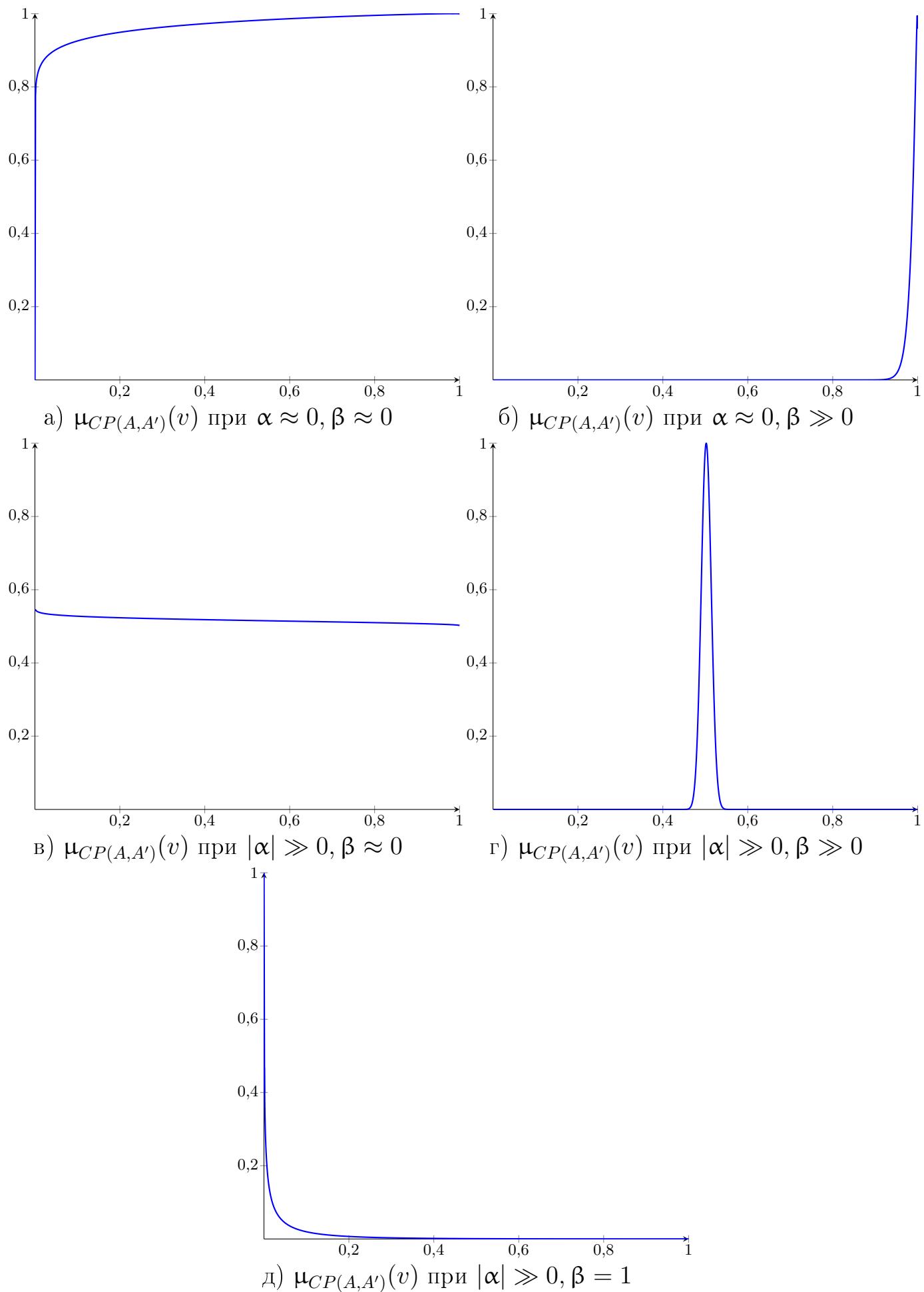


Рисунок 4.5 — Случаи вырождения ф. п. НЗИ в близкий к асимптоте вид.

$$\begin{aligned}
& \exp \left(- \left(\frac{(a_1 - a_2) \pm b_1 \sqrt{-\ln v_0}}{b_2} \right)^2 \right) = 1 \\
& \exp \left(- \left(\frac{(a_1 - a_2) \pm b_1 \sqrt{-\ln v_0}}{b_2} \right)^2 \right) = \exp(0) \\
& \frac{(a_1 - a_2) \pm b_1 \sqrt{-\ln v_0}}{b_2} = 0 \\
& \pm b_1 \sqrt{-\ln v_0} = a_1 - a_2 \\
& \sqrt{-\ln v_0} = \pm \frac{a_1 - a_2}{b_1} \\
& \ln v_0 = - \left(\frac{a_1 - a_2}{b_1} \right)^2 \\
& v_0 = \exp \left(- \left(\frac{a_1 - a_2}{b_1} \right)^2 \right) \quad (4.3)
\end{aligned}$$

Внедрение выражения (4.3) в алгоритм вычисления НЗИ позволит обеспечить соответствие определению дискретизированного представления ф. п. НЗИ. НЗИ с функцией принадлежности, имеющей горизонтальную асимптоту, как на рисунке 4.5в 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.

4.1.2 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.

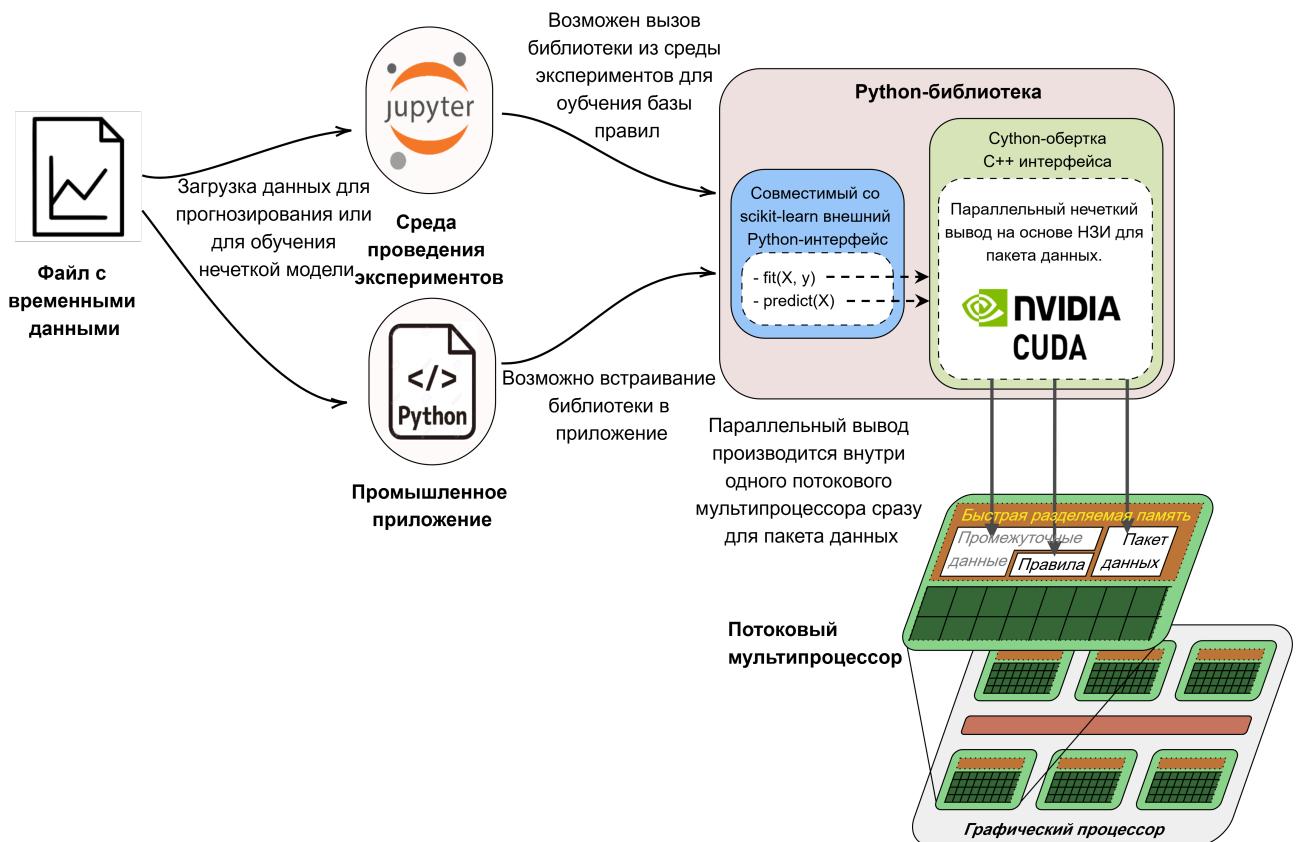


Рисунок 4.6 — Схема использования библиотеки для нечеткого моделирования и прогнозирования временных рядов.

4.2 Оценка работоспособности разработанного метода прогнозирования на основе нечеткого логического вывода относительно других типов нечетких моделей и с позиции вычислительной сложности

Для подтверждения утверждения о полиномиальной (линейной) временной сложности нечеткого вывода на основе нечеткого значения истинности и провести сравнение использования логического нечеткого вывода Л. Заде в задаче, где актуален учет нечеткости данных, с подходами Мамдани и Такаги-Сугено, следует провести несколько вычислительных экспериментов и выполнить оценку характеристик разработанного подхода в этой задаче.

Для простоты сравнения с результатами публикаций, использующих несинглтонную фазификацию, для проведения экспериментов будет использоваться синтетический набор данных Mackey-Glass (M-G). Набор данных с такими же значениями параметров уравнения, как в этих публикациях: $\tau = 30$, $\beta = 0.2$, $\gamma = 0.1$. Так же была сгенерирована последовательность из 2000 точек на участке от $t = -999$ до $t = 1000$. Первые 1000 точек так же выбрасываются как зона стабилизации процесса $x(t)$, а следующие $T = 1000$ точек на участке $t = \overline{1,1000}$ используются в эксперименте.

$$\frac{dx(t)}{dt} = \beta \frac{x(t - \tau)}{1 + x(t - \tau)^n} - \gamma x(t) \quad (4.4)$$

В данном эксперименте используется адаптивный метод оценки зашумленности в точке t временной последовательности. Для этого сперва вычисляется последовательность разностей d_t соседних значений временной последовательности (4.4).

$$d_1 = d_2, \quad (4.5)$$

$$d_t = \frac{1}{\sqrt{2}}(x_t - x_{t-1}), \quad (4.6)$$

где $t \in [1; T]$.

А затем для каждой точки вычисляются значения среднеквадратичных отклонений $\hat{\sigma}_t$ разностей соседних точек (4.8), но, в отличии от [4], вычисление производится не внутри скользящего окна фиксированного размера, а на основе формулы экспоненциально взвешенного скользящего среднего:

$$\begin{aligned}\hat{\sigma}_t^2 &= (1 - \alpha)\hat{\sigma}_{t-1}^2 + \alpha(d_t - \hat{d}_t) \\ &= \sum_{p=1}^t \alpha(1 - \alpha)^{t-p}(d_p - \hat{d}_p)^2,\end{aligned}\quad (4.7)$$

$$\hat{\sigma}_t = \sqrt{\hat{\sigma}_t^2}, \quad (4.8)$$

где \hat{d}_t — является экспоненциально взвешенным скользящим средним значений d_t :

$$\begin{aligned}\hat{d}_t &= (1 - \alpha)\hat{d}_{t-1} + \alpha d_t \\ &= \sum_{p=1}^t \alpha(1 - \alpha)^{t-p}d_p.\end{aligned}\quad (4.9)$$

Таким образом, в данном способе оценки уровня шума в измеренных значениях [13] сперва с помощью разностного оператора выполняется преобразование в стационарный временной ряд (исключение компоненты тренда) для более точной оценки амплитуды шума, а затем среднеквадратичное отклонение в экспоненциально взвешенной области отражает резкие изменения значений стационарного временного ряда относительно сглаженного центра в данной точке t .

Для каждого значения временной последовательности $x(t)$ в результате фаззификации было получено нечеткое множество A'_t с гауссовой функцией принадлежности с параметрами a и b :

$$\begin{aligned}\mu_{A'_t}(y_t) &= gauss(y_t; a, b), \\ \Theta_{\mu_{A'_t}} &= \langle a, b \rangle = \langle x(t), \varphi \hat{\sigma}_t \rangle,\end{aligned}$$

где параметр среднеквадратичного отклонения равен ранее вычисленному значению $\hat{\sigma}_t$ с коэффициентом $\varphi = 0.3$.

Для построения базы правил использовались точек в диапазоне от $t = 1$ до $t = 700$ (обучающий диапазон) из ранее отложенных $T = 1000$ точек, а для оценки качества итоговой системы были взяты последние 300 точек в диапазоне от $t = 701$ до $t = 1000$ (тестовый диапазон). Диапазоны последовательности этих точек были составлены обучающий и тестовый наборы временных отрезков длины $p+1$ с использованием скользящего окна, где размер окна запаздывания p выбирался среди значений 3,5,7,9,11, а горизонт прогнозирования установлен

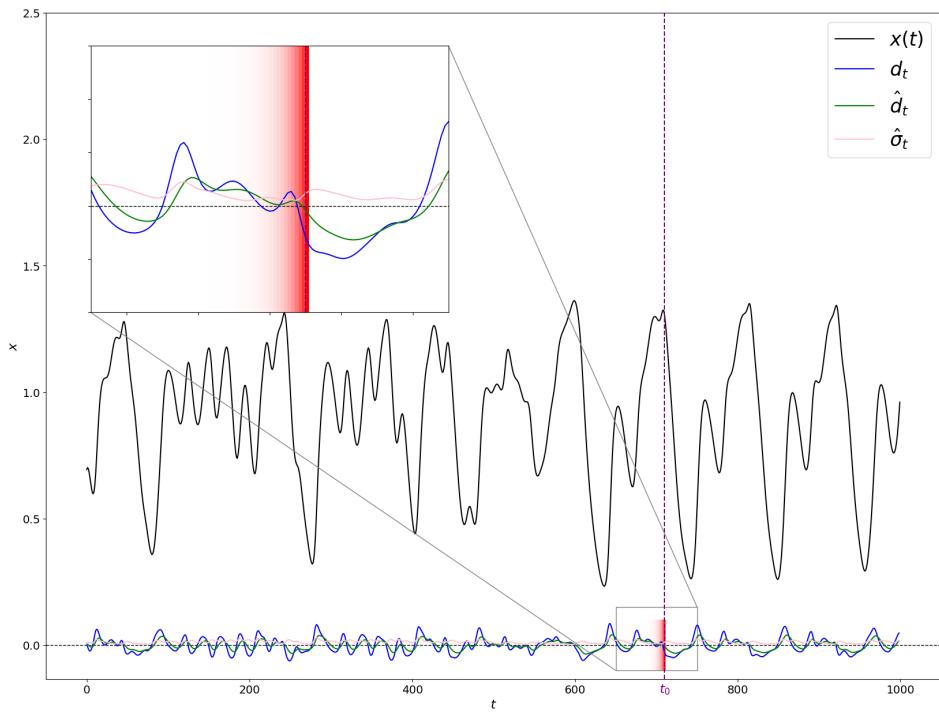


Рисунок 4.7 — График сгенерированной последовательности Mackey-Glass $x(t)$, $t \in [0; 999]$ с параметрами $\tau = 30$, $\beta = 0.2$, $\gamma = 0.1$, график разностей соседних точек d_t последовательности $x(t)$, график \hat{d}_t с наложением экспоненциально взвешенного скользящего среднеквадратичного отклонения разностей $\hat{\sigma}_t$. Наложенном изображении участка $t \in [650, 750]$ яркостью красного цвета показано значения весового коэффициента $\alpha(1-\alpha)^{t-p}$, $p = \overline{1, t_0}$ из формулы (4.7) при $\alpha = 0.2$, определяющего степень вклада предыдущих значений $(d_p - \hat{d}_p)^2$ в значение $\hat{\sigma}_t^2$ в точке t . Чем меньше значение α , тем слабее «доверие» к одному лишь значению d_t и тем больше предыдущих значений d_p делают вклад в сглаженное значение.

$h = 1$. В нечетком выводе использовалась импликация Лукасевича, Т-нормы — \min и метода дефазификации средний максимум (МеOM) с количеством точек алгоритма Gradient-aware PSO — 100 и количеством итераций — 100.

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).

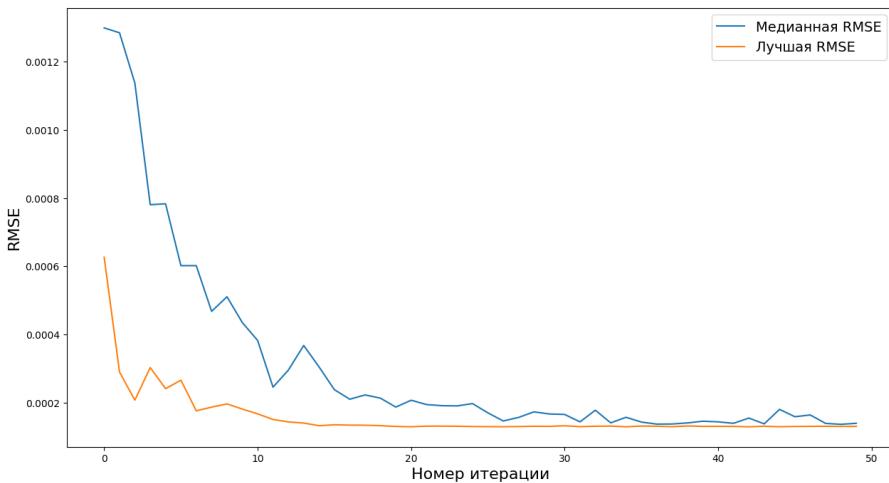


Рисунок 4.8 — Графики изменений среднего и лучшего по популяции значений оптимизируемой функции приспособленности при числе входов нечеткой системы —7, количестве правил — 20, размере набора точек алгоритма PSO — 50 и количестве итераций — 50.

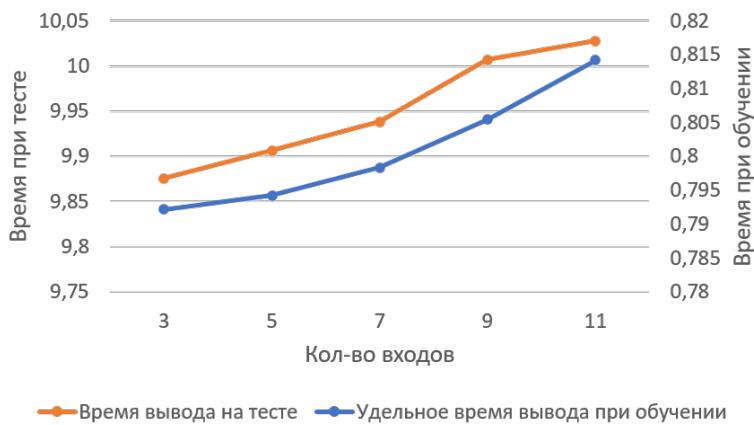


Рисунок 4.9 — График длительности выполнения параллельной реализации нечеткого вывода для обучающего и тестового набора данных при количестве правил $N = 10$.

Динамика среднего (медианного) и лучшего по набору точек значений оптимизируемой метрики в процессе PSO-оптимизации базы правил для указанных выше значений набора гиперпараметров изображена на рисунке 4.8.

После проведенного вычислительного эксперимента были получены показатели удельного (на одну точку из набора точек одной итерации алгоритма PSO) времени работы алгоритма нечеткого вывода на основе НЗИ на обучающем наборе данных и времени работы на тренировочном наборе данных для различного размера окна запаздывания, изображенные на рисунке 4.9. На этом рисунке наблюдается линейный рост времени выполнения алгоритма с увеличением количества входов нечеткой системы, что подтверждает утверж-

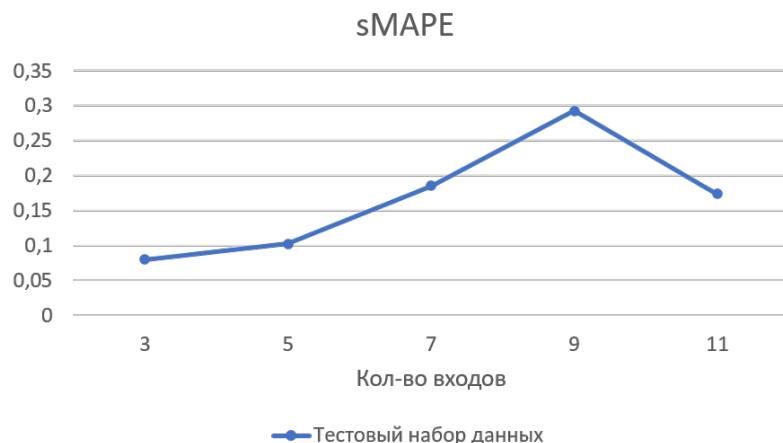


Рисунок 4.10 — График значений метрики sMAPE на обучающем и тестовом наборах данных при различных размерах окна запаздывания, количество правил — 30.

ждение о полиномиальной зависимости временной сложности метода нечеткого вывода на основе НЗИ от количества входов. Временные показатели алгоритма нечеткого вывода для тестового набора данных также включают значительную временную компоненту задержки, вызванную в основном процессом инициализации библиотек при первичном запуске нечеткого вывода для уже готового набора правил, тогда как на обучающем наборе данных получены значения времени работы непосредственно алгоритма нечеткого вывода с амортизированной компонентой задержки.

С целью сравнения с альтернативными нечеткими моделями на примере решения этой же задачи при наиболее близких условиях для полученной в результате экспериментов нечеткой системы затем выполнена оценка точности прогнозирования по метрике *Symmetric Mean Absolute Percentage Error (sMAPE)*, определяемой формулой:

$$sMAPE = \frac{100\%}{n} \sum_{t=1}^n \frac{|y_t - \hat{y}_t|}{\frac{|y_t| + |\hat{y}_t|}{2}},$$

где y_t и \hat{y}_t соответствуют истинному и предсказанному значению в момент времени t .

Значение этой метрики для разных размеров окна запаздывания приведены на рисунке 4.10, где лучший показатель качества прогнозирования временного ряда по метрике sMAPE достигается при размере окна запаздывания — 3 точки. Тогда соответствующая матрица параметров базы правил

имеет вид размера 30×4 :

$$\theta_{\mathbf{R}}^* = \begin{bmatrix} \langle a_{\mu_{A_11}}, b_{\mu_{A_11}} \rangle & \langle a_{\mu_{A_12}}, b_{\mu_{A_12}} \rangle & \langle a_{\mu_{A_13}}, b_{\mu_{A_13}} \rangle & \langle a_{\mu_{B_1}}, b_{\mu_{B_1}} \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle a_{\mu_{A_{301}}}, b_{\mu_{A_{301}}} \rangle & \langle a_{\mu_{A_{302}}}, b_{\mu_{A_{302}}} \rangle & \langle a_{\mu_{A_{303}}}, b_{\mu_{A_{303}}} \rangle & \langle a_{\mu_{B_{30}}}, b_{\mu_{B_{30}}} \rangle \end{bmatrix}.$$

Достигнутое в этом случае значение $sMAPE = 8\%$ при количестве правил 30, заметно превосходит точность моделирования незашумленной последовательности Mackey-Glass (M-G) с использованием синглтонной фазификации со значением $sMAPE \approx 40\%$ в [4]. Также полученное качество прогнозирования сопоставимо со значением этой метрики при аналогичной конфигурации эксперимента прогнозирования временного ряда M-G в той же публикации, где для достижения точности моделирования временной последовательности $sMAPE \approx 10\%$ используется нечеткая система типа Мамдани, содержащая 184 правила в базе правил для не зашумленного временного ряда M-G и 597, 402, 312 правил для различных конфигураций и амплитуды добавленного шума. Данные наблюдения показывают, что **метод регрессии временных рядов на основе логического нечеткого вывода при несинглтонной фазификации** значительно превосходит качество регрессии с использованием синглтонной фазификации, а также имеет сопоставимую с методом регрессии Мамдани при несинглтонной фазификации точность, но при меньшем количестве правил за счет возможности построения более сложной функции аппроксимации.

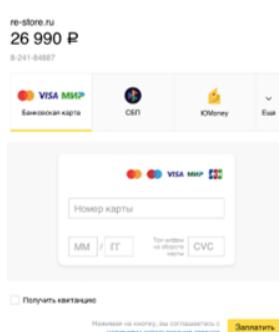
4.3 Оценка работоспособности разработанной информационной технологии прогнозирования на основе нечеткой модели относительно вероятностных подходов.

Второй эксперимент посвящен применению разработанного метода для прогнозирования помесячного объема транзакций безналичных платежей корпоративных клиентов банка. Транзакционная активность юридических лиц является уникальным процессом из-за нишевой бизнес-модели и уникальных экономических условий, которые среди прочего определяют транзакционный профиль клиента банка.

Безналичные платежи называются термином эквайринг. 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.



POS-терминал



Интернет-эквайринг



Платеж по QR-коду

Рисунок 4.11 — Виды эквайринга.

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.

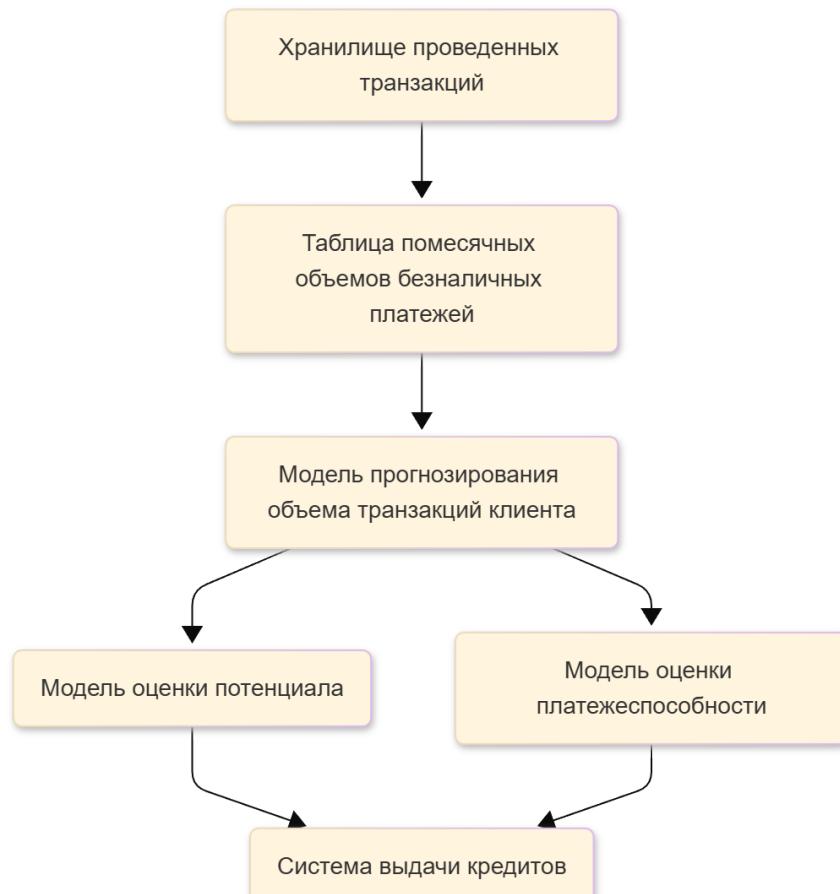


Рисунок 4.12 — Схема использования прогнозов объема транзакций по безналичным платежам в банковской системе.

На рисунке 4.12 показан пример схемы системы управления выдачи кредитов корпоративным клиентам банка, использующей прогноз объема транзакций безналичной оплаты юридических лиц для принятия решений.

Временной ряд клиента в этом наборе данных имеет вид помесячных значений объемов транзакций s_t , каждое из которых получено в результате агрегации (складывания) сумм отдельных транзакций x_{ti} за месяц:

$$s_t = \sum_{i=1}^{N_t} x_{ti},$$

где N_t — количество транзакций за месяц t .

В зависимости от рода деятельности клиента банка, сам временной ряд может иметь разную степень стохастичности, возникающей из-за скрытой многофакторной неопределенности при деятельности клиента. Степень стохастичности временного ряда в точке t можно оценить с помощью описанного в предыдущем эксперименте метода оценки среднеквадратичного отклонения в точке t с использованием формул (4.5), (4.9), (4.7) и (4.8).

В [14] подмечено: для стабильных временных рядов — более простые и плавные модели часто превосходят сложные, но с высокой степенью стохастичности справляются только сложные модели с большой емкостью знаний и учитывающие эту стохастичность. Как было показано в первой главе при увеличении ширины функции принадлежности нечеткого множества входного значения увеличивается количество активируемых окрестных правил нечеткой системы. То есть при малой ширине входной ф. п. выходное значение получается в узкой области небольшого количества правил, а при большой ширине входной ф. п. задействуется большое количество правил в широкой области, то есть по сути повышается емкость нечеткой системы. Тогда степень стохастичности месячного объема транзакций t может быть использована для задания ширины гауссовой функции принадлежности в процедуре фазификации:

$$\begin{aligned}\mu_{A'_t}(y_t) &= gauss(y_t; a, b), \\ \theta_{\mu_{A'_t}} &= \langle a, b \rangle = \langle s_t, \varphi \sigma_{s_t} \rangle,\end{aligned}$$

где коэффициент $\varphi = 0.3$.

Исходный набор данных составлен из помесячных объемов транзакций за период с 01-01-2021 по 31-12-2023 для 2500 клиентов. Из набора данных были удалены временные ряды для клиентов, состоящие только из нулевых значений, то есть клиенты без активности по продукту банка. В результате остались

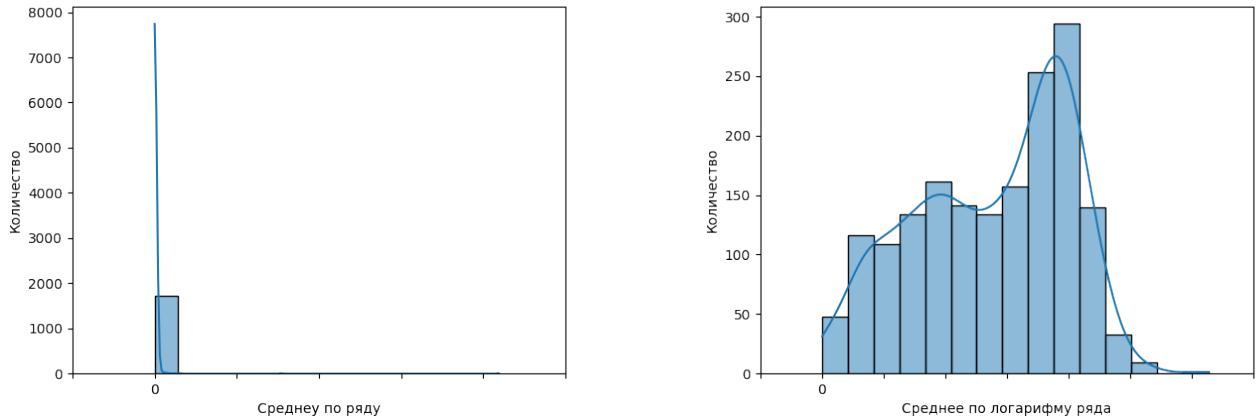


Рисунок 4.13 — Гистограммы математических ожиданий временных рядов.

временные ряды для 1731 активных клиентов. Случайная величина средних по клиенту $\mathbb{E}[s_t]$ имеет экспоненциальное распределение, как показано на рисунке 4.13. Для выравнивания этого распределения и для достижения более равномерного распределения пространства правил нечеткой системы было выполнено логарифмирование значений s_t временных рядов.

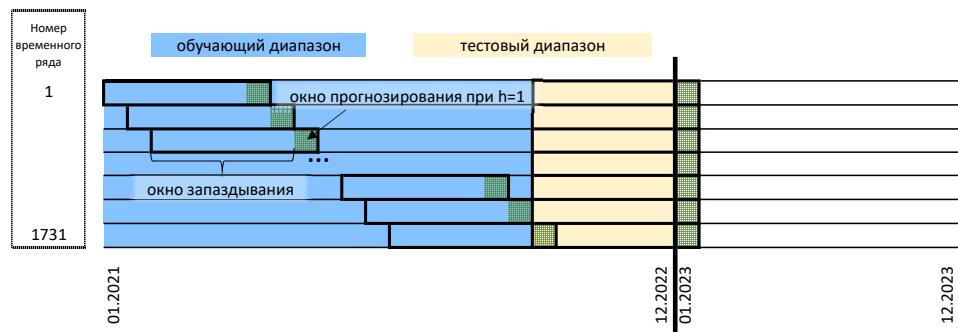


Рисунок 4.14 — Разбиение временных рядов на обучающий и тестовый диапазоны.

По требованиям задачи последние 12 месяцев были отложены в качестве прогнозируемых значений тестового диапазона, а период с 01-2021 по 12-2022 использовался в качестве обучающего диапазона для формирования набора обучающих окон с размером запаздывания $p = 3,6,9,12$ и горизонтом прогнозирования $h = 1, 2, 3$. Данный способ разбиения показан на рисунке 4.14. Для оценки качества прогнозирования использовалась целевая метрика RMAE, которая дает возможность сравнить насколько среднее значение ошибки высоко относительно среднего прогнозируемого значения:

$$RMAE = \frac{MAE}{\mathbb{E}[\mathbb{E}[s_t]]} = \frac{\sum_{t=1}^n |y_t - \hat{y}_t|}{\sum_{t=1}^n y_t},$$

Таблица 1 — Значения метрики МАЕ относительно $\mathbb{E}[\mathbb{E}[s_t]]$ для разработанной нечеткой и альтернативных моделей.

Модель	Relative Mean Absolute Error (RMAE)	
	Прогноз на 1 мес.	Средняя за 3 мес.
Наивная (среднее по ряду)	0.166	0.204
Модель Бокса-Дженкинса	0.187	0.230
Градиентный бустинг	0.152	0.187
Многослойный перцептрон	0.128	0.157
Рекуррентная нейросеть (DeepAR)	0.188	0.231
Разработанная нечеткая модель	0.109	0.142

где y_t и \hat{y}_t соответствуют истинному и предсказанному значению в момент времени t .

Обучение производилось при количестве правил нечеткой системы — 50, точек PSO-алгоритма — 50, количестве итераций — 50. Как и в предыдущем эксперименте использовалась импликация Лукасевича и дефазификация по среднему максимуму (МеОМ) с теми же параметрами алгоритма Gradient-aware PSO. В качестве функции приспособленности выступала метрика среднеквадратичной ошибки (RMSE).

Лучшее качество прогнозирования для разработанной нечеткой модели удалось достичь при размере окна запаздывания — 3 точки. Для сравнения с другими подходами, обычно применяемыми при решении такой задачи, в таблице 1 приведены значения метрики качества при прогнозирования на 1 и 3 месяца для моделей: наивная, Бокса-Дженкинса, градиентный бустинг, многослойный перцептрон, DeepAR.

Стоит описать некоторые детали применения и используемую конфигурацию других моделей:

- В наивной модели прогноз формируется как среднее значение в окне запаздывания.
- При использовании модели Бокса-Дженкинса для каждого клиента обучалась отдельная авторегрессионная модель.
- В качестве градиентного бустинга использовался CatBoostRegressor. На вход модели подавались признаки для различных размеров окон запаз-

дывания: 3, 6, 9 и 12. Для каждого прогнозного месяца была построена отдельная модель бустинга.

- Многослойный перцептрон имел размерность скрытых слоев — 1024 и количество полносвязных слоев — 4.
- В рекуррентной модели DeepAR использовалась размерность скрытого состояния LSTM-слоя — 128 и количество слоев LSTM — 4.

По итогу эксперимента было получено значение метрики $RMAE = 0.109/0.142$ при прогнозировании на один/три месяца соответственно. Это превосходит на 17%/11% качество прогнозирования с использованием многослойного перцептрана со значениями $RMAE = 0.128/0.157$.

Полносвязная нейросетевая модель достигает высокой точности прогнозирования за счет достаточной емкости. Это позволяет построить сложную аппроксимирующую функцию, содержащую в себе набор общих шаблонов помесячных объемов уникальных клиентов. То есть с точки зрения подхода к моделированию временных рядов клиенты не являются уникальными, а являются ансамблем однородных объектов со сложной функцией плотности вероятности. В используемой нечеткой модели на прогнозное значение влияет небольшое количество активируемых для входных значений правил, содержащих наиболее похожие шаблоны динамики объемов транзакций.

4.4 Выводы по главе

1. Высокая производительность реализации информационной технологии прогнозирования обеспечивается за счет использования технологии параллельного программирования CUDA с аппаратной поддержкой для реализации разработанных параллельных алгоритмов. При реализации параллельного алгоритма свертки НЗИ решена проблема «просеивания» значений НЗИ на сквозь точки расчетной сетки. Выполненная реализация информационной технологии оформлена в виде Python-библиотеки, взаимодействующей с низкоуровневой реализацией нечеткого вывода на языке CUDA/C++.
2. 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 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.
3. Проведенный вычислительный эксперимент для сравнения нечеткой модели прогнозирования на основе нечеткого вывода логического типа другими конфигурациями нечетких моделей показал значительный прирост точности прогнозирования по метрике sMAPE относительно модели использующей синглтонную фаззификацию и значительное сокращение количества правил в базе правил при сопоставимой точности прогноза при использовании несинглтонной фаззификации и нечеткого вывода типа Мамдани. Также экспериментально подтверждена полиномиальная зависимость вычислительной сложности используемого метода нечеткого вывода на основе НЗИ в зависимости от числа входов.
 4. Проведенный другой вычислительный эксперимент для сравнения разработанного метода прогнозирования с другими распространенными подходами на задаче прогноза помесячного объема транзакций безналичных платежей корпоративных клиентов банка. Результат эксперимента показал прирост точности по метрике RMAE на 17% при прогнозе на один месяц и на 11% при прогнозе на 3 месяца.

Заключение

Основные результаты работы заключаются в следующем.

1. Проведенный анализ показал, что существующие методы не позволяют адекватно отразить уникальность исследуемого объекта и учесть изменчивость характеристик данных входных временных рядов.
2. Разработан метод прогнозирования временных рядов в задачах управления на основе нечеткого вывода, который позволяет адекватно отразить уникальность и изменчивость характеристик входных данных.
3. Разработаны вычислительные процедуры предложенного метода прогнозирования временных рядов, обеспечивающие вычислительную эффективность за счет использования предложенного метода нечеткого вывода, алгоритма свертки НЗИ и адаптированного алгоритма PSO для параметрической оптимизации базы правил и дефазификации МеОМ.
4. Разработанный прототип программной поддержки информационной технологии прогнозирования на основе параллельной архитектуры вычислений продемонстрировал прирост точности при сравнении с другими методами прогнозирования.

Выводы

- Разработанная информационная технология позволяет усовершенствовать инструменты информационного обеспечения прогнозирования на основе временных рядов.
- Реализация информационной технологии может быть выполнена на стандартных компьютерах, оснащенных графическими ускорителями с поддержкой технологии CUDA.

Перспективы дальнейших исследований

Повышение эффективности алгоритма прогнозирования временных рядов с использованием нечеткой системы за счет более эффективного способа агрегации базы правил.

Рекомендации по использованию

Рекомендуется организациям разрабатывающим программы для информационного обеспечения прогнозирования.

Список литературы

1. *Mendel, J. M.* Non-Singleton Fuzzification Made Simpler [Текст] / J. M. Mendel // Information Sciences. — 2021. — Т. 559. — С. 286–308. — DOI: 10.1016/j.ins.2020.12.061. — URL: <https://www.sciencedirect.com/science/article/pii/S0020025520312275>.
2. *Mendel, J. M.* Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions [Текст] / J. M. Mendel. — 2nd. — Springer Publishing Company, Incorporated, 2017. — DOI: 10.1007/978-3-319-51358-2. — URL: <https://link.springer.com/book/10.1007/978-3-319-51358-2>.
3. *Mendel, J. M.* Comparing the Performance Potentials of Singleton and Non-Singleton Type-1 and Interval Type-2 Fuzzy Systems in Terms of Sculpting the State Space [Текст] / J. M. Mendel, R. Chimatapu, H. Hagras // IEEE Transactions on Fuzzy Systems. — 2020. — Апр. — Т. 28, № 4. — С. 783–794. — DOI: 10.1109/TFUZZ.2019.2916103.
4. *Pekaslan, D.* ADONiS—Adaptive Online Nonsingleton Fuzzy Logic Systems [Текст] / D. Pekaslan, C. Wagner, J. M. Garibaldi // IEEE Transactions on Fuzzy Systems. — 2020. — Окт. — Т. 28, № 10. — С. 2302–2312. — DOI: 10.1109/TFUZZ.2019.2933787.
5. *Рутковский, Л.* Методы и технологии искусственного интеллекта [Текст] / Л. Рутковский ; пер. И. Д. Рудинский. — Москва : Горячая линия-Телеком, 2010.
6. *Куценко Д. А. и Синюк, В. Г.* Методы вывода для систем со многими нечеткими входами [Текст] / В. Г. Куценко Д. А. и Синюк // Известия Российской академии наук. Теория и системы управления. — 2015. — № 3. — С. 48. — DOI: 10.7868/S0002338815030129.
7. *Karatach, S. A.* Method of Inference of Fuzzy Logical Type Systems with Nonsingleton Fuzzification [Текст] / S. A. Karatach, V. G. Sinuk // Pattern Recognition and Image Analysis. — 2024. — Сент. — Т. 34, № 3. — С. 645–651. — DOI: 10.1134/S1054661824700470. — URL: <https://doi.org/10.1134/S1054661824700470>.

8. *Sinuk, V. G.* The Inference Method for a Mamdani Type System with Nonsingleton Fuzzification [Текст] / V. G. Sinuk, S. A. Karatach // Pattern Recognition and Image Analysis. Advances in Mathematical Theory and Applications. — 2023. — Т. 33, № 3. — С. 506—510. — DOI: 10.1134/S1054661823030422.
9. *Wang, L.-X.* Generating Fuzzy Rules by Learning from Examples [Текст] / L.-X. Wang, J. M. Mendel // IEEE Transactions on Systems, Man, and Cybernetics. — 1992. — Ноябрь. — Т. 22, № 6. — С. 1414—1427. — DOI: 10.1109/21.199466.
10. *Chellai, F.* Forecasting using Fuzzy Time Series [Текст] / F. Chellai. — 2022. — URL: <https://ideas.repec.org/p/prapra/113848.html>.
11. *The Kokkos Team.* Kokkos: A C++ Performance Portability Programming EcoSystem [Текст] / The Kokkos Team. — 2023. — URL: <https://kokkos.org/kokkos-core-wiki/>. <https://github.com/kokkos/kokkos>.
12. *Edwards, H. C.* Kokkos: Enabling manycore performance portability through polymorphic memory access patterns [Текст] / H. C. Edwards, C. R. Trott, D. Sunderland // Journal of Parallel and Distributed Computing. — 2014. — Т. 74, № 12. — С. 3202—3216. — DOI: <https://doi.org/10.1016/j.jpdc.2014.07.003>. — URL: <http://www.sciencedirect.com/science/article/pii/S0743731514001257>.
13. *Rank, K.* Estimation of image noise variance [Текст] / K. Rank, M. Lendl, R. Unbehauen // IEE Proceedings - Vision, Image and Signal Processing. — 1999. — Vol. 146, no. 2. — P. 80—84. — DOI: 10.1049/ip-vis:19990238. — URL: <https://digital-library.theiet.org/doi/abs/10.1049/ip-vis%3A19990238>.
14. *Green, K. C.* Simple versus complex forecasting: The evidence [Текст] / K. C. Green, J. S. Armstrong // Journal of Business Research. — 2015. — Vol. 68, no. 8. — P. 1678—1685. — DOI: 10.1016/j.jbusres.2015.03.026. — URL: <https://www.sciencedirect.com/science/article/pii/S014829631500140X>.

Список рисунков

1.1 Схема системы управления на основе информационной технологии прогнозирования	12
2.1 Схема системы нечеткого вывода на основе правил с использованием синглтонной и несинглтонной фаззификации	27
2.2 Сравнение областей активации правил при переходе от синглтонной фаззификации к несинглтонной и при увеличении ширины области неопределенности $\sigma_{A'}$	28
2.3 Сравнение формы функций принадлежности выходных нечетких множеств для подхода Мамдани	30
2.4 Сравнение формы функций принадлежности выходных нечетких множеств для логического подхода. Выходные нечеткие множества получены в результате применения импликации Лукасевича $I(a, b) = 1 - a + b$, поэтому для легкости восприятия антецеденты правил и входные нечеткие множества показаны в виде выражений $1 - \mu_{A_k}(x)$ и $1 - \mu_{A'}(x)$	31
2.5 Примеры временных рядов с различными типами шума: (а) стабильный шум, (б) смешанный стабильный шум, (в) переменный шум. $SNR(dB) = 10 \log_{10} \frac{\sigma_s^2}{\sigma_n^2}$ – отношение среднеквадратичного значения сигнала σ_s^2 к среднеквадратичному значению шума σ_n^2 (ОСШ, <i>signal-to-noise ratio</i>) в децибелах. Большее значение $SNR(dB)$ соответствует меньшему уровню шума.	32
2.6 Схема обобщенной процедуры адаптивной несинглтонной фаззификации временной последовательности.	33
2.7 Иллюстрация процедуры несинглтонной фаззификации временного ряда с низким уровнем шума.	33
2.8 Иллюстрация процедуры несинглтонной фаззификации временного ряда с высоким уровнем шума.	35
2.9 Нейросетевая структура нечеткой модели с несинглтонной фаззификацией и дефаззификацией по среднему максимуму.	36
2.10 Значения лингвистической переменной «истинность»	37
2.11 Пример вычисления нечеткого значения истинности	39

2.12 Иллюстрация случая истинного отношения высказываний	40
2.13 Иллюстрация случая ложного отношения высказываний	41
2.14 Иллюстрация случая абсолютно истинного отношения высказываний	41
2.15 Иллюстрация случая абсолютно ложного отношения высказываний .	42
2.16 Иллюстрация случая квαιистинного отношения высказываний . . .	43
2.17 Сравнение классической схемы нечеткого вывода и схемы нечеткого вывода на основе НЗИ	46
2.18 Схема нейро-нечеткой системы с вычислением НЗИ и дефазификацией по среднему максимуму, а также пример работы процедуры дефазификации.	47
2.19 Пример нечетких множеств, удовлетворяющих условию $\mu_{B_k}(y_r) = 0$ для $y \neq r$	48
 3.1 Пример работы параллельного алгоритма свертки НЗИ при расчетной сетке состоящей из 5 точек.	52
3.2 Ситуация необходимости нахождения среднего максимума при дефазификации агрегированной функции принадлежности $\mu_{B'}(y)$. .	58
3.3 Динамика движения точек популяции при работе алгоритма глобальной оптимизации Gradient-aware PSO.	59
 4.1 Блок схема процедуры нечеткого логического вывода на основе нечеткого значения истинности.	66
4.2 Карта разделяемой памяти одного CUDA-блока нечеткого логического вывода на основе нечеткого значения истинности при использовании дефазификации МеОМ.	68
4.3 График функции $\exp(-z^2)$	71
4.4 График функции $\varphi(v) = \sqrt{-\ln v}$	71
4.5 Случай вырождения ф. п. НЗИ в близкий к асимптоте вид.	72
4.6 Схема использования библиотеки для нечеткого моделирования и прогнозирования временных рядов.	75

4.7 График сгенерированной последовательности Mackey-Glass $x(t), t \in [0; 999]$ с параметрами $\tau = 30, \beta = 0.2, \gamma = 0.1$, график разностей соседних точек d_t последовательности $x(t)$, график \hat{d}_t с наложением экспоненциально взвешенного сглаживания на последовательность разностей и график экспоненциально взвешенного скользящего среднеквадратичного отклонения разностей $\hat{\sigma}_t$. На вложенном изображении участка $t \in [650, 750]$ яркостью красного цвета показано значения весового коэффициента $\alpha(1 - \alpha)^{t-p}, p = \overline{1, t_0}$ из формулы (4.7) при $\alpha = 0.2$, определяющего степень вклада предыдущих значений $(d_p - \hat{d}_p)^2$ в значение $\hat{\sigma}_t^2$ в точке t . Чем меньше значение α , тем слабее «доверие» к одному лишь значению d_t и тем больше предыдущих значений d_p делают вклад в сглаженное значение.	78
4.8 Графики изменений среднего и лучшего по популяции значений оптимизируемой функции приспособленности при числе входов нечеткой системы — 7, количестве правил — 20, размере набора точек алгоритма PSO — 50 и количестве итераций — 50.	79
4.9 График длительности выполнения параллельной реализации нечеткого вывода для обучающего и тестового набора данных при количестве правил $N = 10$	79
4.10 График значений метрики sMAPE на обучающем и тестовом наборах данных при различных размерах окна запаздывания, количестве правил — 30.	80
4.11 Виды эквайринга.	82
4.12 Схема использования прогнозов объема транзакций по безналичным платежам в банковской системе.	83
4.13 Гистограммы математических ожиданий временных рядов.	85
4.14 Разбиение временных рядов на обучающий и тестовый диапазоны.	85

Список таблиц

1	Значения метрики МАЕ относительно $\mathbb{E}[\mathbb{E}[s_t]]$ для разработанной нечеткой и альтернативных моделей.	86
---	--	----

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",
        //     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 0692e7c10b300fb54f240f670cc2026
Владелец Зубов Юрий Сергеевич

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



Приложение В

Справка о практическом применении результатов диссертационной работы в организации

Утверждаю

руководитель направления

по исследованию данных

ПАО Сбербанк

Епишин Егор Васильевич

«04» 12 20_25 г.

СПРАВКА О ВОЗМОЖНОМ ПРАКТИЧЕСКОМ ИСПОЛЬЗОВАНИИ РЕЗУЛЬТАТОВ ИССЛЕДОВАНИЯ

Настоящая справка подтверждает возможное практическое использование результатов диссертационного исследования Карагач Сергея Александровича на тему «Разработка методов и алгоритмов обработки временных рядов в задачах их прогнозирования на основе нечетких систем с учетом нечеткости входов».

В рамках исследования была разработана информационная технология прогнозирования временных рядов, основанная на применении нечетких систем и методах учета нечеткости входных данных. Разработанные методы и алгоритмы позволяют повысить точность краткосрочного и среднесрочного прогнозирования временных рядов, особенно в условиях: когда каждый объект наблюдения является уникальными (т. е. отсутствуют объекты с аналогичными условиями наблюдения).

Данная информационная технология принята к использованию в ПАО Сбербанк для решения практических задач прогнозирования транзакционной активности отдельных групп уникальных клиентов. Внедрение технологии способствует повышению эффективности аналитических процессов, оптимизации распределения ресурсов и улучшению качества принимаемых управленческих решений.

Подписант:

Руководитель направления по исследованию данных в УРБС КИБ ПАО Сбербанк
Епишин Егор Васильевич