ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Чирихин Константин Сергеевич

ИСПОЛЬЗОВАНИЕ МЕТОДОВ ТЕОРИИ ИНФОРМАЦИИ И ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ РАЗРАБОТКИ И ИССЛЕДОВАНИЯ ВЫСОКОТОЧНЫХ МЕТОДОВ ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ

Специальность 05.13.18 —
«Математическое моделирование, численные методы и комплексы программ»

ДИССЕРТАЦИЯ
на соискание учёной степени
кандидата технических наук

Научный руководитель:
д-р техн. наук, проф.
Рябко Борис Яковлевич

Новосибирск — 2022

Оглавление

Стр.

Стр.

Введение

Актуальность исследования. Задача прогнозирования временных рядов обладает большой практической значимостью и привлекает внимание сотен исследователей во всём мире. Она возникает в самых разных областях человеческой деятельности. Например, методы прогнозирования временных рядов используются в экономике [1–3], здравоохранении [4], экологии [5], социальных науках [6] и др.

В настоящее время существует много подходов к решению данной задачи и уже получен ряд эффективных методов в рамках математической статистики, а в последние годы и с применением алгоритмов искусственного интеллекта (ИИ). Но несмотря на полученные методы, ряд важных проблем прогнозирования временных рядов оставался нерешённым, среди которых отметим задачу выявления скрытых закономерностей в данных для использования их при прогнозировании. Существование подобных закономерностей возможно, например, в рядах экономических показателей (таких как биржевые цены), поскольку участники соответствующих процессов могут оказывать влияние на значения этих показателей, используя для этого сложные алгоритмы. В качестве другого примера можно привести временных ряды, связанные с космическими объектами. Вероятно, простейшим примером последовательности с закономерностью, которую известные методы не способны обнаружить, может считаться $01001000100001\ldots$ Поэтому задача разработки методов прогнозирования, способных выявлять подобные закономерности и использовать их для повышения точности прогнозов, является актуальной.

Объект исследования — алгоритмы и методы прогнозирования временных рядов социальных, экономических, физических и других показателей.

Предмет исследования — разработка и применение алгоритмов и методов прогнозирования стационарных и нестационарных временных рядов.

Цель работы — разработка, программная реализация и экспериментальное исследование методов прогнозирования временных рядов, способных обнаруживать не только периоды и простейшие тренды, но и «сложные» скрытые нестационарные закономерности в данных и использовать их для повышения точности прогнозов.

Задачи исследования:

1. Построение методов прогнозирования временных рядов, использующих алгоритмы искусственного интеллекта для обнаружения скрытых закономерностей в данных с целью повышения точности прогнозов;
2. Разработка метода преобразования алгоритмов прогнозирования к методам сжатия данных, что позволит: 1) использовать алгоритмы ИИ, применяемые в методах сжатия; 2) объединять различные методы прогнозирования в один комплекс;
3. Решение задачи выбора лучшего метода прогнозирования из имеющихся без их «полного» перебора, с небольшими затратами времени.

Научная новизна:

1. Впервые построены методы прогнозирования, способные для повышения точности прогнозов находить в данных сложные закономерности за счёт использования многоголовочных автоматов и контекстно-свободных (КС) грамматик;
2. Впервые предложен и реализован теоретико-информационный метод интеграции нескольких методов прогнозирования, позволяющий «автоматически» выбирать наиболее подходящий метод для прогнозирования ряда;
3. Построены новые методы прогнозирования временных рядов, представляющие практический интерес (например, они были использованы для прогнозирования показателей Новосибирской области).

Теоретическая и практическая значимость работы. Предлагаемые в данной работе методы могут быть использованы для прогнозирования временных рядов из различных областей человеческой деятельности, но в первую очередь они разработаны для экономических рядов и рядов, связанных с космическими объектами. Например, результаты экспериментального исследования показывают, что методы обладают высокой точностью при прогнозировании временных рядов солнечных пятен и T-индекса — точность получаемых прогнозов на один шаг вперёд на исторических данных оказалась выше, чем точность прогнозов обсерваторий.

Результаты диссертации внедрены в учебный процесс на кафедре Компьютерных систем ФИТ НГУ при обучении аспирантов по направлению подготовки 09.06.01 Информатика и вычислительная техника, а также в учебный процесс на кафедре Прикладной математики и кибернетики СибГУТИ при обучении бакалав-

ров по направлению подготовки 09.03.01 Информатика и вычислительная техника (см. Приложение А).

Основная часть работы выполнена в рамках следующих проектов:

1. Проект РФФИ № 19-37-90009 «Методы прогнозирования временных рядов, базирующиеся на алгоритмах сжатия данных и искусственного интеллекта» (конкурс «Аспиранты»);
2. Проект РФФИ № 19-47-540001 «Разработка когнитивных методов прогнозирования и их применение для предсказания социально-экономических процессов в Новосибирской области». Отметим, что представленные в диссертации результаты прогнозирования показателей Новосибирской области были выполнены в рамках этого проекта, что является практическим внедрением разработанных методов (см. Приложение А);
3. Проект «Социально-экономическое развитие Азиатской России на основе синергии транспортной доступности, системных знаний о природно-ресурсном потенциале, расширяющегося пространства межрегиональных взаимодействий», проводимый при финансовой поддержке Российской Федерации в лице Министерства науки и высшего образования России, Соглашение № 075-15-2020-804 от 02 октября 2020 г. (грант № 13.1902.21.0016).

Методология и методы исследования. При разработке алгоритмов использовались методы из теории информации, дискретной математики, искусственного интеллекта, теории сложности вычислений. Для исследования точности полученных методов выполнялось построение прогнозов для уже известных значений.

Основные положения, выносимые на защиту:

1. Разработаны методы прогнозирования временных рядов, способные находить сложные закономерности новых классов в данных за счёт использования многоголовочных автоматов и КС-грамматик;
2. Разработаны эффективные (с точки зрения точности и трудоёмкости вычислений) методы объединения алгоритмов прогнозирования временных рядов в один, имеющий наилучшую точность (из объединяемых);
3. Использование сжатия данных при прогнозировании позволяет применять реализованные в алгоритмах сжатия методы ИИ, а также объединять различные методы прогнозирования в один;

4. Предлагаемые методы обладают высокой точностью, что подтверждается результатами вычислительных экспериментов, проводимых с реальными данными.

Соответствие паспорту специальности. В работе получены результаты, соответствующие трём пунктам паспорта специальности 05.13.18 — «Математическое моделирование, численные методы и комплексы программ» по техническим наукам:

1. Разработка новых математических методов моделирования объектов и явлений;
2. Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий;
3. Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента.

Апробация работы. Основные результаты работы докладывались на:

- XXI Всероссийская конференция молодых учёных по математическому моделированию и информационным технологиям (Россия, Новосибирск, 2020);
- The 34th annual European Simulation and Modelling Conference (France, Toulouse, 2020);
- Российская научно-техническая конференция «Обработка информации и математическое моделирование» (Россия, Новосибирск, 2020);
- Международная научно-практическая конференция «Распределенные информационно-вычислительные ресурсы: цифровые двойники и большие данные» (DICR 2019) (Россия, Новосибирск, 2019);
- International Workshop «Applied Methods of Statistical Analysis. Statistical Computation and Simulation — AMSA'2019» (Russia, Novosibirsk, 2019);
- The 39th International Symposium on Forecasting (Greece, Thessaloniki, 2019).

Публикации. Основные результаты по теме диссертации изложены в 9 печатных изданиях, 1 из которых издана в журнале, индексируемом в WoS и Scopus, 1 статья в журнале, индексируемом в Scopus и входящим в перечень ВАК, 1 статья в журнале из перечня ВАК и 6 публикаций, включённых в сборники трудов конференций. Получено свидетельство о государственной регистрации программы для ЭВМ (см. Приложение Б).

Объём и структура работы. Диссертация состоит из введения, 5 глав, заключения и 2 приложений. Полный объём диссертации составляет 107 страниц, включая 6 рисунков и 30 таблиц. Список литературы содержит 100 наименований.

В главе 1 приведена постановка задачи прогнозирования, кратко описана история развития существующих методов её решения, а также более подробно рассмотрен теоретико-информационный подход к прогнозированию временных рядов. Особое внимание уделено методам сжатия, в основе которых лежит идея поиска компактной контекстно-свободной грамматики, однозначно описывающей сжимаемые данные, что может рассматриваться как разновидность искусственного интеллекта. Также предложен эффективный с точки зрения точности получаемых прогнозов метод объединения нескольких алгоритмов сжатия в один метод прогнозирования.

В главе 2 предлагается алгоритм прогнозирования временных рядов на основе конечных автоматов с множественными головками и приводятся результаты вычислений, проведённых с его использованием. Данный алгоритм способен правильно прогнозировать полилинейные слова, этого не способны делать существующие методы прогнозирования временных рядов. Также он может быть рассмотрен как метод сжатия данных и использован совместно с другими методами сжатия при прогнозировании с целью получения более универсального метода.

В главе 3 описан адаптивный метод прогнозирования на основе универсальных по времени кодов. Этот метод предназначен для сокращения времени вычислений при объединении различных алгоритмов сжатия. Также эта глава содержит описание модификации, применимой к произвольному методу прогнозирования и позволяющей рассматривать этот метод в качестве метода сжатия данных. Эта модификация позволяет совместно использовать методы сжатия и любые другие методы прогнозирования, с возможностью применения адаптивного метода из данной главы.

В главе 4 содержатся результаты прогнозирования временных рядов реальных процессов с помощью предлагаемых методов, а также описание используемой методологии вычислений. Рассматривается прогнозирование данных из M3 Competition, временных рядов T-индекса и K-индекса, ряда солнечных пятен, а также некоторых показателей Новосибирской области.

В главе 5 приведено описание программной реализации разработанных алгоритмов и методов.

В заключении описаны основные выводы исследования.

В приложении А приведены акты о внедрении результатов диссертации.

Приложение Б содержит копию свидетельства о государственной регистрации программы для ЭВМ.

Глава 1. Прогнозирование временных рядов с помощью теоретико-информационных методов

1.1 Постановка задачи прогнозирования временных рядов

Неформально, под *временным рядом* понимают упорядоченную во времени последовательность числовых значений некоторого показателя, характеризующего изучаемый процесс или явление. Будем предполагать, что эти значения доступны в дискретные моменты времени $i_1, i_2, \ldots, i_t$, причём промежуток времени, через который производятся измерения показателя, постоянен (например, 1 месяц, 1 рабочий день, 5 минут и т.д.). При *прогнозировании* временного ряда требуется оценить значения показателя в будущие моменты времени $i_{t+1}, i_{t+2}, \ldots, i_{t+h}$, используя при этом его известные значения в моменты времени $i_1, i_2, \ldots, i_t$. Далее для краткости значение наблюдения в момент времени i_j будем обозначать как x_j, а прогноз для значения процесса в момент времени i_{t+k}, вычисленный с использованием $x_1, x_2, \ldots, x_t$, как $\hat{x}_{t+k|t}$. В качестве примеров временных рядов приведём последовательность измерений температуры воздуха в помещении с интервалом в 1 час и ряд значений официального курса доллара США к российскому рублю, устанавливаемых Банком России на следующий день ежедневно по рабочим дням.

В разных методах прогнозирования временных рядов используются различные подходы к получению прогнозных значений. Например, прогнозные значения могут вычисляться в виде некоторой линейной функции от k предыдущих значений ряда. В данной работе будем искать распределения вероятностей для будущих значений процесса и в качестве прогнозных значений использовать математические ожидания, вычисляемые по найденным распределениям. Опишем этот подход подробнее. Обозначим через $\mathcal{A}$ конечное или бесконечное множество *букв*, которое будем называть *алфавитом*. Для целей данной работы алфавит будет представлять собой либо конечное множество целых чисел, либо ограниченный числовой интервал. Далее, пусть некоторый вероятностный источник порождает последовательности $x_1, x_2, \ldots$ букв алфавита $\mathcal{A}$. Будем считать, что предельное распределение вероятностей $P(x_1 = a_{i_1}, x_2 = a_{i_2}, \ldots, x_n = a_{i_n})$, $a_{i_j} \in \mathcal{A}$ (или плотность вероятности $p(x_1, x_2, \ldots, x_n)$ в случае, если $\mathcal{A}$

является числовым интервалом) неизвестно, но даны t первых букв одной последовательности $x_1, x_2, \ldots, x_t$, порождённой этим источником, для которой и требуется построить прогноз. Для этого будем искать оценку $P^*(x_{t+1} = a_{i_1}, x_{t+2} = a_{i_2}, \ldots, x_{t+h} = a_{i_h} | x_1, x_2, \ldots, x_t)$, $a_{i_j} \in \mathcal{A}$, для неизвестного условного распределения вероятностей следующих h значений этой последовательности (или условной многомерной плотности вероятности в случае, если $\mathcal{A}$ представляет собой вещественный интервал). Зная такую оценку, в качестве прогнозных значений будем использовать математические ожидания, вычисленные по маргинальным условным распределениям (плотностям) вероятностей для каждого шага, на который строился прогноз. Так, например, для дискретного случая прогнозное значение $\hat{x}_{t+j|t}$ для x_{t+j}, $j \in \{1, 2, \ldots, h\}$, может быть записано как

$$\hat{x}_{t+j|t} = \sum_{a \in \mathcal{A}} a \sum_{(b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_h) \in \mathcal{A}^{h-1}} P^*(b_1, \ldots, b_{j-1}, a, b_{j+1}, \ldots, b_h | x_1, x_2, \ldots, x_t),$$

где $\mathcal{A}^{h-1}$ — множество всех последовательностей длины $h-1$ над алфавитом $\mathcal{A}$.

Помимо самих прогнозных значений для будущих наблюдений процесса, практический интерес представляют их доверительные интервалы, т.е. нижние и верхние границы числовых промежутков, которые с заданной вероятностью содержат будущие значения ряда. Ширина доверительного интервала может служить характеристикой точности вычисленного прогноза. Поэтому построение доверительных интервалов для прогнозируемых значений также будет рассматриваться в данной работе.

1.2 Обзор существующих методов прогнозирования

За многие годы исследований в области методов прогнозирования временных рядов было предложено достаточно большое количество разнообразных подходов к решению данной задачи. Кратко рассмотрим историю развития некоторых из них, широко использующихся в настоящее время на практике. Кроме того, рассмотрим историю развития теоретико-информационного подхода к прогнозированию, поскольку он представляет особый интерес для данной работы.

Начнём описание с методов экспоненциального сглаживания. Они сформировались в результате работ Брауна [7; 8], Хольта [9] и Уинтерса [10]. Вероятно,

простейшим методом данного класса является простое экспоненциальное сглаживание (Simple Exponential Smoothing). В этом методе уровень ряда l_t в момент времени t представляется как линейная комбинация всех имеющихся наблюдений в прошлые моменты времени $x_1, x_2, \ldots, x_t$, причём коэффициенты в линейной комбинации (веса) экспоненциально уменьшаются с увеличением возраста значений. Прогнозное значение $\hat{x}_{t+h|t}$ на h шагов вперёд, вычисленное по первым t значениям ряда, принимается равным l_t, $h \geqslant 1$. Ясно, что такой подход не позволяет учесть тренд и сезонность в данных при прогнозировании. В том числе по этой причине были разработаны более сложные методы, в которых для моделирования тренда и сезонности в уравнение включают дополнительные компоненты. Например, в классическом линейном методе Хольта значение тренда b_t оценивается для каждого момента времени t, и затем $\hat{x}_{t+h|t}$ вычисляется как $l_t + hb_t$, т.е. последнее оценённое значение тренда линейно экстраполируется в будущее. В методе Хольта-Уинтерса вводится сезонная компонента s_t, и прогноз на шаг h вычисляется как $l_t + hb_t + s_{t+h-m(k+1)}$, где m — длина одного сезона, $k = \lfloor (h-1)/m \rfloor$ [11]. В приведённых примерах методов тренд и сезонная составляющая записаны в аддитивной форме, но они могут быть и мультипликативными. В [12] отмечено, что методы с линейным трендом имеют тенденцию переоценивать значения ряда при долгосрочном прогнозировании и предложено добавлять в метод параметр φ, $0 \leqslant \varphi \leqslant 1$, обеспечивающий «затухание» тренда: $\hat{x}_{t+h|t} = l_t + \sum_{i=1}^{h} \varphi^i b_t$. Методы с затуханием тренда оказались достаточно эффективными и стали очень популярны [11]. Однако перечисленные методы представляют собой формулы для получения прогнозных значений без описания характеристик случайного процесса, порождающего наблюдаемую последовательность, что не позволяет строить интервалы для будущих значений, а также использовать информационные критерии для выбора наиболее подходящего метода. В работах [13; 14] была предложена удобная и относительно простая вероятностная модель SSOE (Single Source of Error, модель с единственным источником ошибок) и продемонстрировано, что она может лежать в основе некоторых методов экспоненциального сглаживания. В [15] показано, что для каждого из 12 методов экспоненциального сглаживания (считаем, что есть 4 возможных варианта для тренда: без него, линейный тренд, мультипликативный тренд или линейный тренд с затуханием, и 3 возможных варианта для сезонной компоненты: без неё, аддитивная и мультипликативная, в итоге получая 12 различных методов) существуют 2 модели

SSOE: одна с аддитивной ошибкой, а другая с мультипликативной. Обе модели порождают одинаковые точечные прогнозы, но прогнозные интервалы и значения функции правдоподобия будут различными. В результате суммарное число моделей, соответствующих методам экспоненциального сглаживания, возросло до 24. В [16] была исследована модель с мультипликативным трендом с затуханием, а в [17] предложена классификация моделей ETS (Error, Trend, Seasonal), куда помимо предыдущих 12 методов вошли методы с затухающим мультипликативным трендом, и в итоге число различных моделей возросло до 30, а число методов до 15. Далее, в работе [18] предложено обобщение моделей ETS на многомерный случай. В целом можно сказать, что методы данного класса являются относительно простыми, универсальными и хорошо показывают себя на практике, зачастую превосходя по точности более сложные модели [19]. Поэтому они являются одними из самых широко используемых методов прогнозирования в настоящее время [11].

Ещё один большой класс формируют различные методы, в основе которых лежат модели авторегрессии и скользящего среднего. Одними из важнейших ранних работ, посвящённых этим моделям, являются [20–24]. В модели авторегрессии порядка p (AR(p)) значение временного ряда x_t в момент времени t представляется как сумма линейной комбинации p предыдущих значений этого ряда, константы и случайной ошибки, представляющей собой белый шум. В модели скользящего среднего порядка q (MA(q)) текущее значение ряда выражается через сумму линейной комбинации q предыдущих ошибок прогнозов, константы и случайной ошибки, также являющейся белым шумом. Комбинированная модель авторегрессии-скользящего среднего ARMA(p, q) содержит AR-составляющую порядка p и MA-составляющую порядка q. Обобщением модели ARMA является модель ARIMA, включающая в себя третий параметр — d, который обозначает, сколько раз исходный ряд нужно продифференцировать для того, чтобы он стал стационарным. Модель ARIMA была популяризирована в широко известной книге Бокса и Дженкинса [25], где были обобщены имеющиеся на тот момент знания относительно методов данного класса и предложена методология их применения. Разработанные немного позднее информационные критерии, такие как, например, критерий Акаике (Akaike's Information Criterion, AIC) [26] и байесовский информационный критерий (Bayes information criterion, BIC) [27], позволили формализовать процедуру идентификации параметров модели и нашли широкое применение. В работе [28] был предложен альтернативный подход

к выбору модели, который был назван ARARMA. Существует обобщение ARIMA на многомерный случай [29], соответствующая модель носит название VARIMA. Перечислим ещё некоторые популярные модификации модели ARIMA: модель SARIMA (Seasonal ARIMA) [25] для прогнозирования данных с сезонностью, модель ARFIMA (Autoregressive Fractionally Integrated Moving Average) [30], позволяющая порядку дифференцирования d быть не целым числом, модель ARIMAX [31], где в уравнение включается внешняя переменная. Также стоит отметить, что модели ARIMA и ETS тесно связаны: для любого линейного метода экспоненциального сглаживания существует эквивалентная модель ARIMA, но в то же время для любого нелинейного метода экспоненциального сглаживания эквивалентной модели ARIMA не существует [11]. Несмотря на то, что методы из семейства ARIMA часто рассматриваются как более сложные по сравнению с экспоненциальным сглаживанием и теоретически хорошо обоснованы, точность прогнозов, получаемых с помощью экспоненциального сглаживания, часто не хуже, а иногда и лучше, чем у прогнозов, получаемых с помощью ARIMA [19].

В финансовых временных рядах часто присутствует явление, которое называется «кластеризацией волатильности»: периоды времени с низкой изменчивостью в данных сменяются периодами с высокой изменчивостью. Для моделирования подобных временных рядов часто используются модели ARCH (Autoregressive Conditional Heteroskedasticity, Авторегрессионная Условная Гетероскедастичность) [32] и GARCH (Generalized ARCH, обобщённая модель ARCH) [33]. Их отличительной чертой является то, что они явно моделируют дисперсию временного ряда и предназначены для случаев, когда средняя (безусловная) дисперсия ряда является постоянной, однако условная дисперсия меняется и зависит от прошлых значений. В ARCH и GARCH предполагается, что среднее значение ряда равняется 0, и они зачастую применяются к остаткам некоторой другой модели, такой как ARIMA. В модели ARCH(q) условная дисперсия остатка в момент времени t выражается как сумма линейной комбинации квадратов q предыдущих остатков и некоторой константы. В модели GARCH(q, p) в сумму дополнительно включается линейная комбинация p предыдущих условных дисперсий остатков. Данные модели стали достаточно популярны и существенно повлияли на методологии, используемые в области финансов (например, моделирование волатильности позволяет оценить степень риска по портфелю активов на фондовой бирже), а автор модели ARCH Р. Энгл (R. Engle) за её разработку получил Нобелевскую премию по экономике в 2003 году. С момента изобретения

для ARCH/GARCH было предложено множество различных модификаций, их достаточно подробный список приведён в [34]. Дополним, что методы данного класса могут применяться при прогнозировании многомерных данных, соответствующая модель носит название MGARCH (Multivariate GARCH) [35].

В последнее время особенной популярностью пользуются методы на основе нейронных сетей и машинного обучения. Среди преимуществ, которыми нейронные сети обладают по сравнению с классическими статистическими методами, можно отметить минимальные априорные предположения о характеристиках источника данных, способность моделировать более гибкую функциональную зависимость между прошлыми и будущими значениями (нейронные сети способны аппроксимировать любую непрерывную функцию с произвольной точностью [36]), а также нелинейность [37]. Однако общими проблемами нелинейных моделей являются «проклятия сложности и чрезмерной параметризации» [38], поэтому многие исследователи ставили под вопрос практическую применимость нейронных сетей для решения задачи прогнозирования [39]. Кроме того, как отмечено в [40], чаще всего модели на основе нейронных сетей являются расширениями AR-модели (они добавляют к ней нелинейную составляющую или вычисляют нелинейную функцию от AR-модели), в то время как MA-модель рассматривается редко. Тем не менее, в одном из последних соревнований по прогнозированию [41] самым точным оказался гибридный метод на основе нейронных сетей и экспоненциального сглаживания [42]. Организаторы указанного соревнования в качестве одного из результатов приводят заключение о том, что комбинированные методы, включающие в себя несколько статистических методов и/или методов машинного обучения, обеспечивают точность, превосходящую точность индивидуальных методов. Как отмечено в [39], если временной ряд относительно короткий или его статистические характеристики меняются с течением времени, нейронной сети может быть недостаточно данных для обучения. Однако наличие большого количества связанных между собой временных рядов, что особенно актуально в эпоху больших данных, позволяет модели на основе нейронной сети обучиться из нескольких источников и показывать лучшие результаты по сравнению с такими классическими статистическими методами, как экспоненциальное сглаживание и ARIMA.

В данной работе развивается подход к прогнозированию временных рядов, основанный на теоретико-информационных методах, к которым среди прочих относится сжатие данных. Поскольку в настоящее время методы сжатия зачастую

реализованы в составе программ, называющихся *архиваторами*, в дальнейшем иногда будем использовать этот термин, понимая под ним программную реализацию алгоритма сжатия. Впервые идея использования методов теории информации для прогнозирования временных рядов была изложена в [43]. Более точно, было показано, что асимптотически оптимальный метод прогноза для стационарных случайных процессов может быть построен на основе универсального кода. Подробное описание теоретико-информационного подхода к прогнозированию временных рядов может быть найдено в [44]. Экспериментальное исследование методов данного класса было проведено в работах [45; 46]. Авторы указанных работ приводят примеры реальных процессов, при прогнозировании которых методы на основе сжатия данных оказываются эффективными. Тем не менее, в упомянутых работах задача поиска скрытых закономерностей в данных не решалась, а также для прогнозирования был использован только один универсальный код, хотя в настоящее время существует несколько разных подходов к универсальному кодированию. Например, широко используются на практике алгоритмы Лемпела-Зива и их модификации [47–49], преобразование Барроуза-Уилера (Burrows-Wheeler Transform, BWT) [50; 51] совместно с кодом «стопка книг» [52] (который также известен как move-to-front или сокращённо MTF), предсказание по частичному совпадению (Prediction by Partial Matching, PPM) [53–55]. Особый интерес для прогнозирования представляют методы на основе формальных грамматик [56–58]. В кодах данного класса осуществляется поиск компактной грамматики (обычно контекстно-свободной), однозначно представляющей сжимаемый текст. Затем полученная грамматика дополнительно сжимается, причём на этом этапе могут быть использованы как хорошо известные алгоритмы энтропийного кодирования, такие как арифметический код [59], так и специально разработанные методы для сжатия грамматик. Хотя асимптотически все алгоритмы универсального кодирования достигают минимально возможного количества бит на букву, при работе с размерами данных, встречающихся на практике, их эффективность как правило отличается. Кроме того, в их реализациях присутствуют многочисленные оптимизации для повышения степени сжатия реальных данных. Это означает, что в общем случае нельзя заранее отдать предпочтение тому или иному методу.

Подводя итог, можно сказать, что в настоящее время область прогнозирования временных рядов активно развивается. В последние годы в ней наблюдается тенденция к активному использованию подходов искусственного интеллекта, а

также к объединению различных методов прогнозирования в один. Универсального, лучшего во всех случаях метода пока не существует, и усовершенствование методов прогнозирования остаётся актуальной задачей. Подход на основе использования архиваторов является перспективным, но эффективность применения различных универсальных кодов к решению задачи прогнозирования пока не изучена. Кроме того, существуют классы сложных закономерностей, которые известные методы не могут обнаруживать.

1.3 Теоретико-информационный подход к прогнозированию временных рядов

Математически, рассматриваемая задача прогнозирования и задача сжатия данных очень похожи. В данном параграфе приведено описание способа получения метода прогнозирования из произвольного метода сжатия данных без потерь, а в главе 3 предлагается способ преобразования методов прогнозирования к методам сжатия.

Сначала рассмотрим простейший вариант исследуемой задачи прогнозирования: для заданной последовательности x_1,x_2, $\ldots$, x_t, в которой x_i принадлежат некоторому конечному алфавиту $\mathcal{A}$, требуется построить прогноз на 1 шаг вперёд, т.е. оценить значение x_{t+1}. В соответствии с подходом, описанном в параграфе 1.1, представим прогноз в виде условных вероятностей $P(x_{t+1} = a|x_1,x_2,\ldots,x_t)$, $a \in \mathcal{A}$, при этом предполагая, что вероятностная мера P неизвестна. С помощью алгоритма сжатия φ можно построить меру P_φ, которая, при наложении некоторых условий на φ и P, является в определённом смысле непараметрической оценкой P.

Приведём некоторые определения, которыми будем пользоваться в дальнейшем. Множество всех последовательностей (слов) длины $n \in \mathbb{N}$, состоящих из букв алфавита $\mathcal{A}$, обозначим как $\mathcal{A}^n$. Пусть $\mathcal{A}^* = \bigcup\limits_{i=0}^{\infty} \mathcal{A}^i$. *Кодом* φ называется множество отображений $\varphi_n : \mathcal{A}^n \to \{0,1\}^*$, $n = 1, 2, \ldots$ таких, что для любых различных $x, y \in \mathcal{A}^n$ кодовые слова $\varphi(x)$ и $\varphi(y)$ различны. Код φ называется *однозначно декодируемым* (или *разделимым*), если для любой последовательности слов $x_1, x_2, \ldots, x_m$, $x_i \in \mathcal{A}^n$, $m \geqslant 1$, $n \geqslant 1$, последовательность $\varphi_n(x_1), \varphi_n(x_2), \ldots, \varphi_n(x_m)$ может быть однозначно декодирована как

$x_1, x_2, \ldots, x_m$. Далее будем рассматривать только однозначно декодируемые коды, поэтому под термином «код» фактически будем подразумевать однозначно декодируемый код. Длину кодового слова в битах, которое φ сопоставляет некоторому слову α, обозначим как $|\varphi(\alpha)|$. Код φ называется *универсальным*, если для любого стационарного и эргодического [60] источника P справедливы следующие равенства:

$$\lim_{t\to\infty} |\varphi(x_1, x_2, \ldots, x_t)|/t = H(P)$$

с вероятностью 1, и

$$\lim_{t\to\infty} \mathbb{E}(|\varphi(x_1, x_2, \ldots, x_t)|)/t = H(P),$$

где $H(P)$ — предельная энтропия P [61], $\mathbb{E}(f)$ — математическое ожидание f.

Как хорошо известно в теории информации, для любого разделимого кода φ выполняется неравенство Крафта-Макмиллана [61]:

$$\sum_{u\in\mathcal{A}^n} 2^{-|\varphi(u)|} \leqslant 1. \tag{1.1}$$

Используя 1.1, можно задать вероятностную меру P_φ на $\mathcal{A}^n$ [62]:

$$P_\varphi(u) = 2^{-|\varphi(u)|} / \sum_{v\in\mathcal{A}^n} 2^{-|\varphi(v)|},$$

где $u \in \mathcal{A}^n$.

Пусть φ — универсальный код. Тогда для любого стационарного и эргодического источника P выполняются следующие равенства [62]:

1. $\lim\limits_{n\to\infty} \dfrac{1}{n} \log(P_\varphi(x_1, x_2, \ldots, x_n)/P(x_1,x_2,\ldots,x_n)) = 0$ с вероятностью 1,
2. $\lim\limits_{n\to\infty} \dfrac{1}{n} \sum\limits_{u\in\mathcal{A}^n} P(u) \log(P(u)/P_\varphi(u)) = 0,$
3. $\lim\limits_{n\to\infty} \dfrac{1}{n} \sum\limits_{u\in\mathcal{A}^n} P(u)|P(u) - P_\varphi(u)| = 0.$

Таким образом, P_φ может использоваться для оценки искомых условных вероятностей $P(x_{t+1} = a|x_1,x_2,\ldots,x_t)$, $x_i, a \in \mathcal{A}$:

$$P_\varphi(x_{t+1} = a|x_1,x_2,\ldots,x_t) = P_\varphi(x_1,x_2,\ldots,x_t,a)/P_\varphi(x_1,x_2,\ldots,x_t),$$

что равносильно

$$P_{\varphi}(x_{t+1} = a|x_1,x_2,\ldots,x_t) = \frac{2^{-|\varphi(x_1,x_2,\ldots,x_t,a)|}}{\sum\limits_{b\in\mathcal{A}} 2^{-|\varphi(x_1,x_2,\ldots,x_t,b)|}}. \tag{1.2}$$

Рассмотрим пример. Предположим, что дана последовательность $X = 0001110001110001$ и для неё требуется построить прогноз на один шаг вперёд. Для этого воспользуемся широко известной программой для сжатия данных gzip версии 1.6. Будем считать, что в X могут присутствовать только символы из алфавита $\mathcal{A} = \{0, 1\}$. Создадим файл *file0.txt* и запишем в него текст 00011100011100010 в кодировке UTF-8, т.е. поместим в него X с добавленным символом '0' на конце. Аналогично, создадим второй файл *file1.txt* с содержимым 00011100011100011. Сожмём оба файла с помощью gzip с опцией -9 для включения максимального уровня сжатия. Размеры сжатых файлов приведены в таблице 1.1.

Таблица 1.1 — Размеры сжатых файлов из примера прогнозирования на один шаг вперёд

Название исходного файла	Размер сжатого файла, байт	Размер сжатого файла, бит
file0.txt	40	320
file1.txt	39	312

Оценку вероятности появления нуля в качестве следующего символа для X вычислим по 1.2, используя данные из таблицы 1.1:

$$P_{\varphi}(0|0001110001110001) = \frac{2^{-|\varphi(00011100011100010)|}}{2^{-|\varphi(00011100011100010)|} + 2^{-|\varphi(00011100011100011)|}} = \frac{2^{-320}}{2^{-320} + 2^{-312}} = \frac{2^{-8}}{2^{-8} + 2^{0}} = 0.003891051. \tag{1.3}$$

Действуя аналогично, вычислим вероятность появления единицы:

$$P_{\varphi}(1|0001110001110001) = \frac{2^{-|\varphi(00011100011100011)|}}{2^{-|\varphi(00011100011100010)|} + 2^{-|\varphi(00011100011100011)|}} = \frac{2^{-312}}{2^{-320} + 2^{-312}} = \frac{2^{0}}{2^{-8} + 2^{0}} = 0.996108949.$$

В качестве прогнозного значения $\hat{x}_{t+1|t}$ возьмём математическое ожидание, вычисленное по полученному распределению вероятностей:

$$\hat{x}_{t+1|t} = 0 \cdot 0.003891051 + 1 \cdot 0.996108949 = 0.996108949.$$

Округлив это значение до ближайшего элемента $\mathcal{A}$, получим $\hat{x}_{t+1|t} = 1$.

Можно обобщить 1.2 для построения прогнозов более чем на один шаг вперёд:

$$P_\varphi(x_{t+1} = a_{i_1}, \ldots, x_{t+h} = a_{i_h} | x_1, \ldots, x_t) = \frac{2^{-|\varphi(x_1,\ldots,x_t,a_{i_1},\ldots,a_{i_h})|}}{\sum\limits_{(b_{j_1},\ldots,b_{j_h})\in\mathcal{A}^h} 2^{-|\varphi(x_1,\ldots,x_t,b_{j_1},\ldots,b_{j_h})|}}. \quad (1.4)$$

Если требуется дать точечные прогнозы на $h \geqslant 1$ шагов вперёд, в их качестве можно использовать математические ожидания, вычисленные по соответствующим маргинальным распределениям вероятностей. Следовательно, для шагов $i \in \{1, 2, \ldots, h\}$ прогнозные значения $\hat{x}_{t+i|t}$ будут вычисляться как

$$\hat{x}_{t+i|t} = \sum_{a\in\mathcal{A}} a \sum_{(b_{j_1},\ldots,b_{j_{i-1}},b_{j_{i+1}},\ldots,b_{j_h})\in\mathcal{A}^{h-1}} P_\varphi(b_{j_1}, \ldots, b_{j_{i-1}}, a, b_{j_{i+1}}, \ldots, b_{j_h} | x_1, \ldots, x_t). \quad (1.5)$$

Рассмотрим пример прогнозирования на два шага вперёд. Пусть, как и в предыдущем примере, дана последовательность $X = 0001110001110001$ и для построения прогноза используется архиватор gzip версии 1.6. На этот раз необходимо создать четыре файла *file00.txt*, *file01.txt*, *file10.txt* и *file11.txt*. В каждый из них запишем последовательность X, но в первом допишем в конец X комбинацию 00, во втором — 01, в третьем — 10 и в четвёртом — 11. Так, например, файл *file10.txt* будет содержать текст $0001110001110001 10$. Сжав каждый файл программой gzip с ключом -9, получим размеры сжатых файлов, приведённые в таблице 1.2.

Проведём вычисления по формуле 1.4 с использованием значений из таблицы 1.2. Полученное двумерное распределение вероятностей приведено в таблице 1.3. Так, например, вероятность появления комбинации 10 в качестве следующих двух символов X оказалось равной 0.0039:

$$P(10|0001110001110001) = 2^{-328}/(2 \cdot 2^{-336} + 2^{-328} + 2^{-320}) \approx 0.0039.$$

Таблица 1.2 — Размеры сжатых файлов из примера прогнозирования на два шага вперёд

Название исходного файла	Размер сжатого файла, байт	Размер сжатого файла, бит
file00.txt	42	336
file01.txt	42	336
file10.txt	41	328
file11.txt	40	320

Таблица 1.3 — Совместное распределение вероятностей для возможных значений x_{t+1} и x_{t+2}, где t — номер последнего известного значения временного ряда, равный 16

x_{t+1}	x_{t+2}	
	0	1
0	$\approx 1.5199 \times 10^{-5}$	$\approx 1.5199 \times 10^{-5}$
1	≈ 0.0039	≈ 0.9961

Используя формулу 1.5, получим прогноз для x_{t+1} и x_{t+2} в виде числовых значений, а не распределения вероятностей:

$$\begin{aligned}\hat{x}_{t+1|t} =& 0 \cdot [P_\varphi(00|0001110001110001) + P_\varphi(01|0001110001110001)] + \\ & 1 \cdot [P_\varphi(10|0001110001110001) + P_\varphi(11|0001110001110001)] = \\ & 0 \cdot (2 \cdot 1.5199 \times 10^{-5}) + 1 \cdot (0.0039 + 0.9961) \approx 1,\end{aligned}$$

$$\begin{aligned}\hat{x}_{t+2|t} =& 0 \cdot [P_\varphi(00|0001110001110001) + P_\varphi(10|0001110001110001)] + \\ & 1 \cdot [P_\varphi(01|0001110001110001) + P_\varphi(11|0001110001110001)] = \\ & 0 \cdot (1.5199 \times 10^{-5} + 0.0039) + 1 \cdot (1.5199 \times 10^{-5} + 0.9961) \approx 0.9961.\end{aligned}$$

Заметим, что вычисления по формулам 1.2 и 1.4 можно упростить. Для этого нужно найти наибольшее слагаемое в знаменателе (т.е. слагаемое с наименьшим модулем степени), вынести его как множитель в числителе и знаменателе и сократить. Например, в 1.3 таким слагаемым было 2^{-312} — после сокращений на эту величину в числителе осталось 2^{-8}, а в знаменателе $2^{-8} + 1$. Данный подход позволяет уменьшить погрешность при вычислениях по перечисленным формулам на компьютере.

Далее рассмотрим прогнозирование вещественных временных рядов. Для построения прогнозов с помощью методов сжатия данных такие ряды необходимо предварительно преобразовывать к рядам с конечными алфавитами с помощью процедуры *квантования*. В простейшем варианте эта процедура заключается в разбиении отрезка, содержащего все значения рассматриваемого ряда, на некоторое конечное число равных пронумерованных интервалов и последующую замену каждого элемента ряда номером интервала, в который этот элемент попадает. Пусть дан вещественный временной ряд $Y = y_1, y_2, \ldots, y_t$, $y_i \in \mathbb{R}$. Обозначим его наименьший и наибольший элементы как m и M соответственно: $m = \min\limits_{i \in \{1,\ldots,t\}} \{y_i\}$, $M = \max\limits_{i \in \{1,\ldots,t\}} \{y_i\}$. Предположим, что разбиение производится на n интервалов (выбор значения n рассмотрим далее, пока будем считать что оно задано). Заменим значение y_i на k если $y_i \in [m + k\delta; m + (k+1)\delta)$, $k \in \{0, 1, \ldots, n-2\}$, $\delta = (M-m)/n$, или на $n-1$, если $y_i \in [m+(n-1)\delta; m+n\delta]$. В результате получим ряд $X = x_1, x_2, \ldots, x_t$, $x_i \in \{0, 1, \ldots, n-1\}$, для прогнозирования которого может быть применён описанный ранее подход. Получив прогноз для X на требуемое количество шагов вперёд, заменим прогнозные номера интервалов, например, на их середины: если $\hat{x}_{t+j|t} = l$, $j \in \{1, 2, \ldots, h\}$, $l \in \{0, 1, \ldots, n-1\}$, то $\hat{y}_{t+j|t} = m + (l + 1/2)\delta$.

В качестве примера рассмотрим построение прогноза для ряда $Y = 3.4, 0.1, 3.9, 4.8, 1.5, 1.8, 2.0, 4.9, 5.1, 2.1$ на два шага вперёд. Будем считать, что $n = 4$. Наименьшее значение в рассматриваемом ряде $m = 0.1$, наибольшее $M = 5.1$, $\delta = (5.1 - 0.1)/4 = 1.25$. Соответствие между интервалами и их номерами при квантовании приведено в таблице 1.4.

Таблица 1.4 — Соответствие между интервалами и их номерами при квантовании для примера прогнозирования вещественного временного ряда

Интервал	Номер
$[0.1; 1.35)$	0
$[1.35; 2.6)$	1
$[2.6; 3.85)$	2
$[3.85; 5.1]$	3

Заменив все значения Y на номера соответствующих интервалов, получим следующий целочисленный временной ряд: $X = 2, 0, 3, 3, 1, 1, 1, 3, 3, 1$. Поочерёдно допишем в конец X все возможные комбинации двух символов из алфавита

$\mathcal{A} = \{0, 1, 2, 3\}$, сжимая получающиеся последовательности, как и ранее, с помощью gzip. Полученные по результатам сжатия длины кодовых слов приведены в таблице 1.5. Так, например, значение, записанное на пересечении строки с $x_{t+1} = 0$ и столбца с $x_{t+2} = 1$, было получено путём сжатия последовательности 203311133101.

Таблица 1.5 — Размеры в битах сжатых представлений последовательности X из примера прогнозирования вещественного временного ряда с разными возможными комбинациями x_{t+1} и x_{t+2} на конце, где t — номер последнего известного значения ряда, равный 10

x_{t+1}	x_{t+2}			
	0	1	2	3
0	336	336	336	336
1	328	320	328	328
2	336	336	336	336
3	336	336	336	336

Далее, используя данные из таблицы 1.5 и формулу 1.4, найдем совместное распределение вероятностей для следующих двух значений X. Результаты вычислений приведены в таблице 1.6.

Таблица 1.6 — Совместное распределение вероятностей для возможных значений x_{t+1} и x_{t+2} из примера прогнозирования вещественного временного ряда, где t — номер последнего известного значения ряда, равный 10

x_{t+1}	x_{t+2}			
	0	1	2	3
0	1.50793×10^{-5}	1.50793×10^{-5}	1.50793×10^{-5}	1.50793×10^{-5}
1	0.00386	0.98824	0.00386	0.00386
2	1.50793×10^{-5}	1.50793×10^{-5}	1.50793×10^{-5}	1.50793×10^{-5}
3	1.50793×10^{-5}	1.50793×10^{-5}	1.50793×10^{-5}	1.50793×10^{-5}

С помощью таблицы 1.6 вычислим прогнозные значения $\hat{y}_{t+1|t}$ и $\hat{y}_{t+2|t}$. Для этого заменим номера интервалов их серединами и вычислим математические ожидания:

$$\begin{aligned}\hat{y}_{t+1|t} =& \frac{0.1+1.35}{2}[P_\varphi(00|203311133100) + P_\varphi(01|203311133100)+\\ & P_\varphi(02|203311133100) + P_\varphi(03|203311133100)]+\\ & \frac{1.35+2.6}{2}[P_\varphi(10|203311133100) + P_\varphi(11|203311133100)+\\ & P_\varphi(12|203311133100) + P_\varphi(13|203311133100)]+\\ & \frac{2.6+3.85}{2}[P_\varphi(20|203311133100) + P_\varphi(21|203311133100)+\\ & P_\varphi(22|203311133100) + P_\varphi(23|203311133100)]+\\ & \frac{3.85+5.1}{2}[P_\varphi(30|203311133100) + P_\varphi(31|203311133100)+\\ & P_\varphi(32|203311133100) + P_\varphi(33|203311133100)] =\\ & 0.725 \cdot 6.03172 \times 10^{-5} + 1.975 \cdot 0.99982 + 3.225 \cdot 6.03172 \times 10^{-5}+\\ & 4.475 \cdot 6.03172 \times 10^{-5} \approx 1.97515.\end{aligned}$$

Аналогично, $\hat{y}_{t+2|t} \approx 0.725 \cdot 0.00391 + 1.975 \cdot 0.98826 + 3.225 \cdot 0.00391 + 4.475 \cdot 0.00391 \approx 1.98476$.

Рассмотрим вопрос о выборе числа интервалов при квантовании n. Заметим, что количество последовательностей, которые необходимо сжать при прогнозировании, равно $|\mathcal{A}|^h$. Поскольку $|\mathcal{A}| = n$, это значение чрезвычайно сильно влияет на трудоёмкость алгоритма. Очевидно, оно влияет и на его точность. Если n слишком мало, важные для прогнозирования закономерности в данных могут быть утрачены при квантовании, замена номера интервала на его середину при вычислении точечного прогноза будет грубой. Если n слишком велико, представляющие интерес закономерности также могут быть потеряны из-за шумов, присутствующих в данных, и к тому же трудоёмкость вычислений будет высокой. Для «автоматического» выбора количества интервалов, обеспечивающего хорошую точность, предложим следующий подход. Выберем в качестве n некоторую положительную целую степень двойки, тогда $k = \log_2 n$ — положительное целое число. При построении прогноза будем использовать все временные ряды, получающиеся при квантовании с разбиениями области возможных значений ряда на 2^i, $i = 1, 2, \ldots, k$, интервалов. Пусть временной ряд $X = x_1, x_2, \ldots, x_t$, $x_j \in \mathcal{A}_i = \{0, 1, \ldots, 2^i - 1\}$, получен из ряда $Y = y_1, y_2, \ldots, y_t$, $y_j \in \mathbb{R}$, с помощью процедуры квантования с использованием 2^i интервалов. В таком случае будем обозначать X как $X^{[i]} = x_1^{[i]}, x_2^{[i]}, \ldots, x_t^{[i]}$, $x_j^{[i]} \in \mathcal{A}_i$. Будем вычислять $P_\varphi(x_1^{[k]}, x_2^{[k]}, \ldots, x_t^{[k]})$ как

$$P_{\varphi}(x_1^{[k]}, x_2^{[k]}, \ldots, x_t^{[k]}) = \frac{\sum\limits_{i=1}^{k} \omega_i 2^{-|\varphi(x_1^{[i]}, x_2^{[i]}, \ldots, x_t^{[i]})| + t(k-i)}}{\sum\limits_{i=1}^{k} \sum\limits_{(z_1, z_2, \ldots, z_t) \in \mathcal{A}_i^t} \omega_i 2^{-|\varphi(z_1, z_2, \ldots, z_t)| + t(k-i)}}, \qquad (1.6)$$

где ω_i — неотрицательные весовые коэффициенты, причём $\sum\limits_{i=1}^{k} \omega_i = 1$. Варьируя значения этих коэффициентов, можно отдавать предпочтение разбиениям с большим или меньшим количеством интервалов.

Поясним смысл прибавки $t(k-i)$ к длинам кодовых слов в 1.6. Для примера будем рассматривать разбиения на 2, 4 и 8 интервалов при квантовании, т.е. $k = 3$. Это означает, что всего будет три ряда $X^{[1]}$, $X^{[2]}$ и $X^{[3]}$ в введённых ранее обозначениях. Пусть каждый из них состоит из t элементов. Обратим внимание, что в 1.6 вычисляется вероятность для $X^{[k]}$, т.е. для $X^{[3]}$. Предположим, что нужно сжать $X^{[3]}$. Это можно сделать непосредственно, но также можно сжать $X^{[2]}$, получив $\varphi(X^{[2]})$, и вместе с ним «передать» вспомогательную последовательность из t бит, которую обозначим как $E = e_1, e_2, \ldots, e_t$ и которая необходима для получения $X^{[3]}$ из $X^{[2]}$. Поскольку каждый интервал из разбиения на 4 интервала содержит два интервала из разбиения на 8 интервалов, e_i сделаем равным 1, если x_i попадает в интервал с большим номером в разбиении на 8 интервалов, и 0, если с меньшим. Повторив эту процедуру, можем сжать $X^{[1]}$ вместо $X^{[3]}$, но на каждый элемент ряда потребуется ещё один дополнительный бит (1 бит чтобы восстановить $X^{[2]}$ по $X^{[1]}$, и ещё один бит чтобы восстановить $X^{[3]}$ по $X^{[2]}$). Естественно, при прогнозировании последовательности E не передаются и не формируются, а просто учитывается их размер путём прибавки $t(k-i)$ к полученным длинам кодовых слов.

Рассмотрим пример. Пусть интервал, в который попадают все значения временного ряда, равен $[0; 2]$. Интервалы, которые получаются при разбиении $[0; 2]$ на 2, 4 и 8 интервалов, приведены в таблице 1.7. До двоеточий в ячейках этой таблицы приведёны номера интервалов, а после — их границы. Теперь предположим, что $X^{[3]} = 7, 4, 0, 1, 3, 5$. Ему соответствует $X^{[2]} = 3, 2, 0, 0, 1, 2$. По $X^{[2]}$ можно восстановить $X^{[3]}$, если иметь в наличии последовательность $E = 1, 0, 0, 1, 1, 1$. В самом деле, известно, что $x_1^{[2]} = 3$, значит возможные значения для $x_1^{[3]}$ — это 6 и 7. Поскольку $e_1 = 1$, то выбираем для $x_1^{[3]}$ больший номер, т.е. 7. Аналогично,

$x_2^{[2]} = 2$ и $e_2 = 0$, значит из возможных значений 4 и 5 выбираем меньшее, поэтому $x_2^{[3]} = 4$, что соответствует действительности. Таким образом, зная $X^{[2]}$ и E, $X^{[3]}$ восстанавливается однозначно и без потерь.

Таблица 1.7 — Соответствие между интервалами и их номерами при разбиении отрезка $[0; 2]$ на 2, 4 и 8 интервалов

<table>
<tr><td colspan="3">Количество интервалов</td></tr>
<tr><td>2</td><td>4</td><td>8</td></tr>
<tr><td rowspan="4">0: $[0; 1)$</td><td rowspan="2">0: $[0.0; 0.5)$</td><td>0: $[0.0; 0.25)$</td></tr>
<tr><td>1: $[0.25; 0.5)$</td></tr>
<tr><td rowspan="2">1: $[0.5; 1.0)$</td><td>2: $[0.5; 0.75)$</td></tr>
<tr><td>3: $[0.75; 1.0)$</td></tr>
<tr><td rowspan="4">1: $[1; 2]$</td><td rowspan="2">2: $[1.0; 1.5)$</td><td>4: $[1.0; 1.25)$</td></tr>
<tr><td>5: $[1.25; 1.5)$</td></tr>
<tr><td rowspan="2">3: $[1.5; 2.0]$</td><td>6: $[1.5; 1.75)$</td></tr>
<tr><td>7: $[1.75; 2.0]$</td></tr>
</table>

1.4 Объединение нескольких методов сжатия в один метод прогнозирования

Как было отмечено в параграфе 1.2, современные методы сжатия данных разнообразны и большинство из них имеют различные модификации, разработанные с целью повышения их эффективности. Поэтому в общем случае нельзя заранее сказать, какой метод окажется наиболее подходящим для прогнозирования того или иного временного ряда. В данном параграфе предлагается подход, позволяющий «автоматизировать» процесс выбора лучшего метода. Предположим, что задано некоторое конечное множество методов сжатия $F = \{\varphi_1, \varphi_2, \ldots, \varphi_k\}$. Для построения прогноза на h шагов вперёд для последовательности $X = x_1, x_2, \ldots, x_t$, x_i принадлежит некоторому конечному алфавиту $\mathcal{A}$, модифицируем формулу 1.4, взвешивая кодовые вероятности, получаемые от разных методов из F:

$$P_F(x_{t+1}=a_{i_1},\ldots,x_{t+h}=a_{i_h}|x_1,\ldots,x_t)=\frac{\sum\limits_{i=1}^{k}\gamma_i 2^{-|\varphi_i(x_1,\ldots,x_t,a_{i_1},\ldots,a_{i_h})|}}{\sum\limits_{(b_{j_1},\ldots,b_{j_h})\in\mathcal{A}^h}\sum\limits_{i=1}^{k}\gamma_i 2^{-|\varphi_i(x_1,\ldots,x_t,b_{j_1},\ldots,b_{j_h})|}}, \tag{1.7}$$

где γ_i — неотрицательные весовые коэффициенты, причём $\sum\limits_{i=1}^{k}\gamma_i=1$.

Заметим, что наибольшее влияние на распределение вероятностей для $x_{t+1},\ldots,x_{t+h}$, получаемое по формуле 1.7, оказывает φ_i, обеспечивающий наилучшую степень сжатия для временного ряда X с некоторым его возможным продолжением $b_{j_1},\ldots,b_{j_h}$ на конце. Поэтому можно сказать, что в процессе вычислений по 1.7 φ_i «выбирается» из F для прогнозирования X.

Добавим, что формулы 1.6 и 1.7 могут быть использованы совместно.

1.5 Прогнозирование многомерных временных рядов

На практике возможны ситуации, в которых нужно построить прогноз для нескольких временных рядов, связанных между собой. Например, временные ряды температуры воздуха, атмосферного давления, скорости и направления ветра взаимосвязаны и их совместное прогнозирование может иметь смысл. Обобщим методы из параграфов 1.3–1.4 на многомерный случай. При этом будем рассматривать только временные ряды с конечными целочисленными алфавитами, однако для прогнозирования вещественных рядов, как и в одномерном случае, можно воспользоваться процедурой квантования. Предположим, что даны k временных рядов $X_1, X_2, \ldots, X_k$, $X_j = x_{1j}, x_{2j}, \ldots, x_{tj}$, $j \in \{1, 2, \ldots, k\}$, и их элементы x_{ij} принадлежат алфавиту $\mathcal{A} = \{0, 1, \ldots, n-1\}$. Пусть требуется построить прогноз на h шагов вперёд для каждого ряда. Можно свести эту задачу к одномерному случаю, построив новый ряд $X' = x'_1, x'_2, \ldots, x'_t$ по следующему правилу:

$$x'_i = \sum_{j=1}^{k} x_{ij}|\mathcal{A}|^{j-1}. \tag{1.8}$$

В результате получим ряд с алфавитом $\mathcal{A}' = \{0, 1, \ldots, n^k - 1\}$, по которому можно построить прогноз для $x'_{t+1}, x'_{t+2}, \ldots, x'_{t+h}$ с помощью описанных ранее

методов. Затем, разложив $\hat{x}'_{t+j|t}$ по формуле 1.8, получим прогнозные значения для $X_1, X_2, \ldots, X_k$.

В качестве примера преобразуем временные ряды

$$Y_1 = 3208, -355, 2163, 2807, -154, 1760, -2234, -3706$$

$$Y_2 = 6385, -5729, 1279, 1924, -2513, 1173, 1442, -2837$$

к одномерному ряду целых чисел, который можно непосредственно прогнозировать с помощью методов сжатия данных. Несмотря на то, что Y_1 и Y_2 являются целочисленными, применим процедуру квантования для уменьшения размеров алфавитов. Наименьший элемент первого ряда равен −3706, наибольший — 3208. Для второго ряда это значения −5729 и 6385 соответственно. Для простоты будем рассматривать только разбиения на два интервала: $\mathcal{A} = \{0, 1\}$. Соответствие между получающимися интервалами и их номерами приведено в таблице 1.8.

Таблица 1.8 — Соответствие между интервалами и их номерами при квантовании для рядов Y_1 и Y_2

Ряд	Номер интервала	
	0	1
Y_1	$[-3706; -249)$	$[-249; 3208]$
Y_2	$[-5729; 328)$	$[328; 6385]$

Заменив в Y_1 и Y_2 все элементы номерами содержащих их интервалов, получим следующие два ряда:

$$X_1 = 1,0,1,1,1,1,0,0,$$

$$X_2 = 1,0,1,1,0,1,1,0.$$

Используя 1.8, построим по X_1 и X_2 временной ряд X':

$$X' = x'_1, x'_2, \ldots, x'_8 = 3, 0, 3, 3, 1, 3, 2, 0.$$

Предположим, что прогноз для следующих двух значений X', полученный с помощью какого-либо метода сжатия данных, равен 3, 1. Разложим прогнозные значения по 1.8: $3 = 1 \cdot 2^0 + 1 \cdot 2^1$, и $1 = 1 \cdot 2^0 + 0 \cdot 2^1$. Прогнозными значениями для X_1 являются коэффициенты при 2^0, для X_2 — при 2^1. Таким образом, прогнозом для номеров интервалов следующих двух значений Y_1 является последовательность

1, 1, Y_2 — последовательность 1, 0. Снова используя таблицу 1.8, заменим номера интервалов их серединами, округляя дробные значения до ближайшего чётного целого. В результате получим прогноз 1480, 1480 для Y_1 и 3356, −2700 для Y_2.

1.6 Поиск закономерностей с помощью формальных грамматик

Формальные грамматики широко используются в информатике и имеют разнообразные приложения [63–65]. В [56] было показано, что на основе контекстно-свободных грамматик могут быть построены универсальные коды. С точки зрения прогнозирования этот подход представляет интерес, поскольку коды на основе грамматик могут быть способны выявлять новые виды закономерностей.

Приведём необходимые определения. Под *формальной грамматикой* G понимается четвёрка $(\Sigma, \Gamma, S, \Delta)$, в которой Σ — конечное множество *терминальных* символов, Γ — конечное множество *нетерминальных* символов, причём $\Sigma \cap \Gamma = \varnothing$, $S \in \Gamma$ — специальный символ, который называется начальным, Δ — множество правил вывода вида $\alpha \to \beta$, $\alpha \in (\Sigma \cup \Gamma)^* A (\Sigma \cup \Gamma)^*$, $A \in \Gamma$, $\beta \in (\Sigma \cup \Gamma)^*$. В [66] была предложена классификация грамматик, получившая название *иерархии Хомского*. Ограничения, накладываемые на вид правил вывода грамматики, определяют её принадлежность к тому или иному классу в иерархии. Пусть $\alpha, \beta, \gamma \in (\Sigma \cup \Gamma)^*$, $w \in \Sigma^*$, $A, B \in \Gamma$. Если на вид правил вывода грамматики не накладывается никаких ограничений, за исключением того, что α не является пустой цепочкой, то она относится к типу 0 и порождаемый ею язык называется *рекурсивно-перечислимым*. Если все правила вывода грамматики имеют вид $\alpha A \beta \to \alpha \gamma \beta$, где γ не является пустой цепочкой, то она называется *контекстно-зависимой* и относится к типу 1. Если правила грамматики имеют вид $A \to \alpha$, то она называется *контекстно-свободной* и относится к типу 2. И, наконец, если грамматика состоит из правил вида $A \to w$ и $A \to wB$ (или $A \to Bw$), то она называется *регулярной* и принадлежит к классу 3. Ясно, что с увеличением номера класса строгость ограничений, накладываемых на правила вывода грамматики, возрастает, и грамматика с классом большего номера удовлетворяет ограничениям, накладываемым на классы с меньшим номером. Поэтому обычно под классом грамматики подразумевают класс с максимальным номером, которому удовлетворяет грамматика.

Далее, будем говорить, что последовательность β *непосредственно выводится* из последовательности α в грамматике G и обозначать этот факт как $\alpha \xrightarrow{G} \beta$, если в G существует такое правило вывода $\gamma \to \omega$, что β получается из α заменой вхождения γ на ω. Последовательность β называется *выводимой* из последовательности α в грамматике G и обозначается $\alpha \overset{G}{\Rightarrow} \beta$, если существует последовательность строк $\alpha_1, \alpha_2, \ldots, \alpha_k$ такая, что

$$\alpha = \alpha_1 \xrightarrow{G} \alpha_2, \alpha_2 \xrightarrow{G} \alpha_3, \ldots, \alpha_{k-1} \xrightarrow{G} \alpha_k = \beta.$$

Языком $L(G)$, описываемым грамматикой G, называется множество всех последовательностей терминальных символов, выводимых из S:

$$L(G) = \{\alpha \in \Sigma^* : S \overset{G}{\Rightarrow} \alpha\}.$$

Под *размером грамматики* $|G|$ понимают сумму длин правых частей её правил:

$$|G| = \sum_{(A \to \alpha) \in \Delta} |\alpha|.$$

Основная идея использования грамматик для сжатия данных и, как следствие, для прогнозирования, заключается в следующем. Предположим, что требуется сжать последовательность X. Построим компактную формальную грамматику G такую, что $L(G) = \{X\}$. Тогда вместо передачи или хранения X можно передать или сохранить G. Если никаких ограничений на вид грамматики не накладывается и требуется найти грамматику минимального размера, то данная задача сводится к вычислению колмогоровской сложности слова, которая, как известно, не вычислима. Однако если будем производить поиск G в классе контекстно-свободных грамматик, то задача станет вычислимой, хотя и будет являться NP-трудной [67]. Поэтому на практике, как правило, используют приближённые алгоритмы.

Рассмотрим пример. Предположим, что требуется сжать последовательность $X = 10011000100010011$. Данная последовательность однозначно выводится из КС-грамматики $G = (\{0, 1\}, \{S, X_1, X_2\}, S, \Delta)$, где Δ:

$$S \to X_1 X_2 X_2 X_1 1,$$
$$X_1 \to 1001,$$
$$X_2 \to 1000.$$

Заметим, что длина X равна 17, а $|G| = 13$. Очевидно, что если просто записать правые части правил вывода грамматики в строку без каких-либо разделителей, то восстановить исходное сообщение по такой строке не получится. Наверное, в качестве одного из самых простых вариантов решения этой задачи можно предложить ввести символ-разделитель для правил грамматики, известный декодеру, с его использованием записать все правила в одну строку и затем сжать её, например, арифметическим кодом [59]. Однако существуют более эффективные методы [68].

Далее рассмотрим существующие приближённые алгоритмы построения компактных КС-грамматик G по заданной строке X таких, что $L(G) = \{X\}$. Поскольку необходимо, чтобы X однозначно выводилась из G, на G накладывается ряд ограничений [67]:

1. для каждого нетерминального символа $A \in \Gamma$ в G существует ровно одно правило вывода, в котором A стоит в левой части;
2. G ациклична, т.е. все нетерминальные символы можно упорядочить таким образом, что каждый нетерминал предшествует всем нетерминалам в правой части его правила вывода.

Обозначим через G^* кратчайшую КС-грамматику, однозначно описывающую X. Для сравнения разных алгоритмов используют *порядок аппроксимации* $a(n)$, который определяется как отношение размера грамматики G, построенной рассматриваемым алгоритмом, к размеру кратчайшей грамматики G^* в худшем случае:

$$a(n) = \max_{X \in \Sigma^n} \left(\frac{|G(X)|}{|G^*(X)|} \right).$$

В [67] было показано, что если P $\neq$ NP, то не существует полиномиального по времени алгоритма с порядком аппроксимации лучше, чем 8569/8568. Поэтому все алгоритмы, пригодные для использования на практике, являются приближёнными. Одним из первых был разработан алгоритм Sequitur [57]. Он предлагался не только для сжатия данных, а в целом для выделения иерархических структур в последовательностях с целью их дальнейшего анализа. Несколькими годами позже в [69] был описан похожий, но улучшенный алгоритм, известный как Sequential. В [67] были найдены оценки степени аппроксимации для ряда существующих на тот момент алгоритмов, для Sequential она оказалась равной $O((n/\log n)^{\frac{3}{4}})$.

Хорошо известный алгоритм LZ-78 [48] неявно строит формальную грамматику, для него была получена оценка $O((n/\log n)^{\frac{2}{3}})$. Лучшей верхней оценкой для существующих алгоритмов, проанализированных в [67], является $O((n/\log n)^{\frac{1}{2}})$ для алгоритма Bisection [70]. В [71] были получены более точные оценки степени аппроксимации для алгоритмов Bisection и LZ-78: в случае Bisection $a(n) = \Theta((n/\log n)^{\frac{1}{2}})$, для LZ-78 $a(n) = \Theta((n/\log n)^{\frac{2}{3}})$.

Для примера рассмотрим работу алгоритма Re-Pair (от Recursive Pairing) с порядком аппроксимации $O((n/\log n)^{\frac{2}{3}})$, который был предложен в [72]. Данный алгоритм отличается простотой и его удобно применять на практике.

Предположим, что требуется сжать строку $X = x_1, x_2, \ldots, x_n$. Алгоритм Re-Pair состоит из следующих шагов:

1. $Y = X$, множество правил искомой грамматики является пустым;
2. Найти в Y пару соседних символов (не делая различий между терминальными и нетерминальными символами) a и b, таких, что ab является самой часто встречающейся подстрокой длины 2 в Y. Если не существует пары соседних символов, встречающихся более одного раза, перейти на шаг 5;
3. Ввести новый нетерминальный символ A и добавить в грамматику правило $A \to ab$;
4. Заменить все вхождения ab в Y на A, перейти на шаг 2;
5. Добавить в грамматику правило $S \to Y$.

После выполнения шагов 1–5, полученную грамматику нужно сжать алгоритмом энтропийного кодирования нулевого порядка, таким как арифметический код [59].

В [67] были предложены два новых алгоритма с экспоненциально лучшими порядками аппроксимации — $O(\log^3 n)$ и $O(\log(n/g^*))$, где g^* — размер кратчайшей грамматики для сжимаемой строки. В ряде работ были описаны другие алгоритмы: в [73] предложен алгоритм на основе суффиксных деревьев с аппроксимацией $O(\log_2(n/g^*))$, на основе Re-Pair в [74] также был разработан $O(\log_2(n/g^*))$-алгоритм. Трудоёмкость по памяти описанных алгоритмов $\Omega(n)$. В [75] описан алгоритм, названный LCA (сокращение от lowest common ancestor, обозначающее узел, являющийся общим предком для двух заданных узлов в двоичном дереве и имеющий самый короткий наидлиннейший путь до своих листьев), требующий $O(g^* \log_2 g^*)$ памяти с аппроксимацией $O(\log_2 n \log_2 g^*)$ и линейным временем работы, а в [58] этот алгоритм был улучшен до уров-

ня аппроксимации $O(\log_2 n \log^* n)$, где $\log^* n$ — итеративная логарифмическая функция, с временем работы $O(n \log^* n)$. В [76] предложен простой линейный алгоритм с аппроксимацией $4g^* \log_{3/2}(n/g^*)$.

В данной работе будем использовать готовые программные реализации алгоритмов сжатия, основанных на грамматиках, объединяя их вместе с другими алгоритмами сжатия по формуле 1.7. Отметим, что не для всех упомянутых алгоритмов поиска компактных грамматик такие реализации есть (по крайней мере, общедоступные). В главе 4, посвящённой прогнозированию реальных данных, используются реализация алгоритма Re-Pair, описанная в [77], а также реализация модифицированного алгоритма LCA, степень аппроксимации которого $O(\log^2 n)$ [68].

Глава 2. Прогнозирование временных рядов с помощью многоголовочных автоматов

2.1 Мотивация разработки метода

Несмотря на то, что к настоящему времени было предложено достаточно большое количество разнообразных методов прогнозирования временных рядов, существуют классы закономерностей, которые не могут быть обнаружены известными методами. В данной главе ограничимся рассмотрением рядов с конечными алфавитами (для прогнозирования вещественных временных рядов можно использовать процедуру квантования, описанную в предыдущей главе), и в примерах будем использовать алфавиты вида $\mathcal{A} = \{0, 1, \ldots, n-1\}$, поскольку любой другой конечный алфавит можно преобразовать к данному виду с помощью нумерации его символов. Для начала скажем, что рассмотренные ранее методы сжатия данных, равно как и некоторые другие алгоритмы прогнозирования, могут успешно прогнозировать последовательности с периодическими закономерностями, такие как

$$000111000111000\ldots \tag{2.1}$$

В предыдущей главе было показано, что с помощью методов сжатия данных 2.1 можно корректно экстраполировать.

Другими закономерностями, которые классические методы сжатия успешно выявляют, являются «запрещённые» комбинации символов. Например, в последовательности

$$X = 102333201231101 \tag{2.2}$$

после 1 никогда не встречается 3, поэтому при построении совместного распределения вероятностей для нескольких будущих значений вероятности комбинаций, начинающихся с 3, будут низкими. В качестве примера в таблице 2.1 приведено распределение вероятностей для следующих двух элементов этой последовательности, полученное с помощью архиватора gzip. Как видно из этой таблицы, все комбинации, начинающиеся с 3, сжимаются хуже прочих.

Таблица 2.1 — Построение распределения вероятностей для следующих двух элементов последовательности X (2.2) с помощью gzip

Возможное продолжение Y для X	Размер сжатой последовательности XY, байт	$\approx P_{\text{gzip}}(XY)$
00	49	5.8463×10^{-8}
01	47	0.0038
02	49	5.8463×10^{-8}
03	49	5.8463×10^{-8}
10	47	0.0038
11	49	5.8463×10^{-8}
12	49	5.8463×10^{-8}
13	49	5.8463×10^{-8}
20	47	0.0038
21	47	0.0038
22	47	0.0038
23	46	0.9808
30	49	5.8463×10^{-8}
31	49	5.8463×10^{-8}
32	49	5.8463×10^{-8}
33	49	5.8463×10^{-8}

Для прогнозирования последовательностей с трендами также разработаны различные подходы. Поясним, что многие алгоритмы прогнозирования временных рядов не способны экстраполировать тренд (например, к ним относятся модель авторегрессии, методы сжатия данных и др.), поэтому часто прогнозируют данные, очищенные от тренда, и затем вносят корректировку на тренд в построенные прогнозы. Например, для удаления тренда на практике широко используется процедура взятия n-ой разности. Взятие первой разности (дифференцирование ряда) заключается в вычитании из каждого значения временного ряда предшествующего ему значения, т.е. от ряда $x_1, x_2, \ldots, x_t$ переходят к ряду $x_2 - x_1, x_3 - x_2, \ldots, x_t - x_{t-1}$. При взятии n-ой разности данная процедура повторяется n раз. С помощью взятия разностей также можно удалить из данных и сезонную составляющую, если вычитать из каждого значения ряда не предше-

ствующее ему значение, а значение, зафиксированное m шагов назад, где m — длина цикла в сезонной составляющей.

Однако существуют классы закономерностей, которые не могут быть обнаружены методами из предыдущей главы, равно как и другими известными методами прогнозирования временных рядов. В качестве примера простой последовательности, содержащей закономерность подобного класса, можно привести

$$0100100010000100000\ldots \tag{2.3}$$

в которой серии нулей заканчиваются единицами, и каждая серия на один нуль длиннее предыдущей. Очевидно, что следующими двумя символами 2.3 должны быть 01. Однако, если прогнозное значение выражается через линейную комбинацию k предыдущих значений (как, например, это делается в авторегрессионной модели), то ясно, что когда длина серии нулей будет гораздо больше k, не получится распознать момент, когда вместо 0 должна появиться 1. С помощью методов сжатия данных также не удаётся распознать позицию единицы. Пусть конечное слово X является префиксом 2.3 и заканчивается единицей, т.е. содержит целое число k серий нулей. Попробуем построить прогноз для $(k+1)$-й серии нулей и следующей за ней единицы для некоторых k с помощью различных методов сжатия. Будем использовать следующую схему вычислений. Алгоритму подаётся на вход слово $X = x_1, x_2, \ldots, x_t$, представляющее собой k серий нулей, за которыми следуют единицы, и $x_t = 1$, затем он должен сделать $(k+1)+1$ одношаговых прогнозов. По начальному слову $X = x_1, x_2, \ldots, x_t$ строится прогноз для x_{t+1}, затем правильное значение добавляется в конец X и строится прогноз для x_{t+2} по слову $X = x_1, x_2, \ldots, x_t, x_{t+1}$ и так далее. Для измерения точности будем использовать сумму модулей ошибок, поскольку с увеличением k количество нулей в серии возрастает, а они, как правило, прогнозируются безошибочно и, следовательно, средняя ошибка будет стремиться к 0 с увеличением k. Результаты вычислений для всех используемых в данной работе методов сжатия данных (их краткие описания со ссылками на веб-сайты проектов приведены в параграфе 4.1) содержатся в таблице 2.2. Видно, что каждый из них периодически выдаёт ошибочные одношаговые прогнозы.

В данной главе разрабатывается метод, способный правильно прогнозировать последовательности с подобными закономерностями и подходящий для совместного использования с архиваторами из предыдущей главы с целью повышения его универсальности. Однако отметим, что данный метод также имеет

ограничения и не составляет труда придумать закономерности, которые он не способен обнаружить. Например, $00111000001111111\ldots$ — последовательность, в которой серии из нулей и единиц сменяют друг друга и длина серии с номером i равна i-тому простому числу.

Таблица 2.2 — Суммы ошибок при прогнозировании $(k+1)$-й серии нулей и следующей за ней единицы по k предшествующим сериям нулей и единиц слова 2.3 различными методами сжатия

k	Название архиватора						
	lcacomp	rp	zstd	bzip2	zlib	ppmd	zpaq
20	4	6	2	2	2	1	1
40	19	20	8	1	2	1	1
60	1	31	2	1	0	1	1
80	1	27	1	74	2	1	1
100	1	50	2	3	2	1	1

2.2 Описание метода

Перейдём к описанию предлагаемого метода. Он основан на алгоритме из [78] для прогнозирования полилинейных (multilinear) слов [79] и является его модификацией. Согласно [79], бесконечное слово называется *полилинейным*, если оно может быть записано в виде

$$q\prod_{n\geqslant 0} r_1^{a_1 n+b_1} r_2^{a_2 n+b_2}\ldots r_m^{a_m n+b_m},$$

где $\prod$ обозначает конкатенацию (под *конкатенацией* слов $X = x_1, x_2, \ldots, x_{n_1}$ и $Y = y_1, y_2, \ldots, y_{n_2}$ понимается слово XY длины n_1+n_2, получаемое дописыванием слова Y в конец слова X: $XY = x_1, x_2, \ldots, x_{n_1}, y_1, y_2, \ldots, y_{n_2}$), q — некоторое конечное слово, m — положительное целое, и для любого $i \in \{1, 2, \ldots, m\}$ r_i не пусто, а a_i и b_i — неотрицательные целые числа, такие что $a_i + b_i > 0$.

Ясно, что 2.1 является полилинейным, поскольку его можно записать в виде

$$\prod 000111.$$

Последовательность 2.3 также является полилинейной, т.к. имеет вид

$$\prod_{n\geqslant 0} 0^{n+1}1.$$

Конечные автоматы, рассматриваемые в [78; 79] и используемые в данной главе, несколько отличаются от «классических» автоматов, которые применяются, как правило, для распознавания цепочек языков соответствующих классов. Например, в широко известной книге [80] *детерминированный конечный автомат* (*ДКА*) определяется как пятёрка $(Q, \mathcal{A}, \delta, q_0, F)$, где Q — конечное множество состояний, $\mathcal{A}$ — конечный входной алфавит, $q_0 \in Q$ — начальное состояние, $F \subseteq Q$ — множество допускающих состояний и δ — функция переходов вида $Q \times \mathcal{A} \to Q$. На вход ДКА подаётся конечное слово $\alpha \in \mathcal{A}^*$, записанное на ленту. В начальный момент времени головка автомата указывает на первый символ на ленте, а сам автомат находится в состоянии q_0. За один переход автомат меняет своё текущее состояние q на $\delta(q, a)$, где a — символ α, на который указывает головка автомата, и перемещает головку на одну позицию вправо. Если после прочтения α автомат окажется в состоянии, принадлежащем множеству F, то говорят, что автомат *допускает* α. Для целей прогнозирования ДКА M в [78] определяется как пятёрка $M = (Q, \mathcal{A}, T, \triangleright, q_0)$, где Q — конечное множество состояний, $\mathcal{A}$ — конечный входной алфавит, $\triangleright$ — маркер начала строки, $q_0 \in Q$ — начальное состояние, T — функция переходов вида $Q \times (\mathcal{A} \cup \{\triangleright\}) \to Q \times \mathcal{A}$. На вход ДКА подаётся бесконечное слово α, записанное на ленту после маркера начала строки $\triangleright$. В начальный момент времени головка автомата указывает на маркер начала строки, а сам автомат находится в состоянии q_0. За один переход $(q_n, b) = T(q_c, a)$ автомат меняет своё текущее состояние q_c на новое состояние q_n, а также выдаёт прогноз b для следующего символа в зависимости от того, на какой символ a на ленте указывает его головка, и перемещает головку на одну позицию вправо. Таким образом, автомат формирует последовательность прогнозных значений $M(\alpha)$. Обозначим операцию получения i-го символа слова α как $\alpha[i]$. Если существует $n_0 \geqslant 1$ такое, что $M(\alpha)[i] = \alpha[i]$ для любого $i \geqslant n_0$, то будем говорить, что M *обучается* (masters) α. При сопоставлении определений автоматов видно, что три из пяти компонент остались прежними. Поскольку при прогнозировании автомат не описывает никакой язык, множество допускающих состояний F перестаёт быть нужным. В то же время для целей прогнозирования усложняется функция переходов и вводится специальный начальный символ, не принадлежащий $\mathcal{A}$.

Известно, что возможностей ДКА не хватает для прогнозирования полилинейных слов [78]. Более того, не существует ДКА, способного обучиться любому периодическому слову над алфавитом $\mathcal{A}$ (если $|\mathcal{A}| \geqslant 2$). Поэтому для дальнейшего изложения необходимо дать определения более сложных автоматов. *Многоголовочный детерминированный конечный автомат* (*МДКА*) M — шестёрка $(Q, \mathcal{A}, k, T, \triangleright, q_0)$, где Q — конечное множество состояний, $\mathcal{A}$ — конечный входной алфавит, $k \geqslant 1$ — количество головок, $\triangleright$ — маркер начала строки, $q_0 \in Q$ — начальное состояние, T — функция переходов вида $Q \times (\mathcal{A} \cup \{\triangleright\})^k \to Q \times \{\text{stay}, \text{right}\}^k \times \mathcal{A}$. На вход M подаётся бесконечное слово α, записанное на ленту после маркера начала строки $\triangleright$. В начальный момент времени все головки автомата указывают на маркер начала строки, а сам автомат находится в состоянии q_0. За один переход каждая из k головок автомата либо остаётся на месте, либо передвигается на одну позицию вправо, автомат меняет своё состояние, а также выдаёт прогноз для следующего символа. Если за переход автомат не перемещает крайнюю правую головку вправо, то его прогноз за этот переход отбрасывается. Таким образом, автомат формирует последовательность прогнозных значений $M(\alpha)$, в которую входят прогнозные значения только за те переходы, при которых автомат достигает нового символа на ленте.

Вопрос о том, существует ли МДКА, способный обучиться любому полилинейному слову, остаётся открытым [78]. Но в [78] было показано, что существует интеллектуальный (sensing) МДКА (ИМДКА), который способен начиная с некоторого символа правильно прогнозировать любое полилинейное слово над алфавитом $\mathcal{A}$, и приведён алгоритм его работы. Под *интеллектуальным МДКА* понимается МДКА, у которого функция переходов T принимает дополнительный аргумент, позволяющий определить для каждой пары головок, находятся они на одной позиции на ленте или нет. Для целей данной работы нужно, чтобы этот алгоритм работал с конечными словами и результатом его выполнения была не последовательность одношаговых прогнозов автомата $M(\alpha)$, а оценка вероятности $P_M(X)$ появления входного конечного слова $X = x_1, x_2, \ldots, x_n$, $x_i \in \mathcal{A}$, на выходе неизвестного источника P. Зная $P_M(X)$, длина кодового слова $|\varphi_M(X)|$ в битах для X, нужная для включения автомата в формулу 1.7 вместе с методами сжатия данных, может быть получена как $|\varphi_M(X)| = -\log_2 P_M(X)$.

Для получения искомой оценки вероятности поступим следующим образом. Вероятность слова будем вычислять как произведение условных вероятностей для отдельных его букв:

$$P_M(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P_M(x_i|x_1, x_2, \ldots, x_{i-1}), \tag{2.4}$$

причём для первого символа будем считать $P_M(x_1|\ldots) = P_M(x_1) = 1/|\mathcal{A}|$.

В алгоритме из [78] на некоторых шагах прогнозное значение, которое выдаёт автомат, строго указано (случай 1), в других местах оно может быть любым (случай 2). Будем по-разному вычислять оценки $P_M(x_i|x_1, x_2, \ldots, x_{i-1})$ для случаев 1 и 2.

Предположим, что уже просмотрено i букв X и оценивается вероятность $P_M(x_{i+1}|x_1, \ldots, x_i)$. Пусть, согласно прогнозу автомата, следующим символом будет $a \in \mathcal{A}$ и автомат «уверен» в своём выборе, т.е. имеем дело со случаем 1. Тогда припишем a вероятность

$$P_M(a|x_1, \ldots, x_i) = \frac{C + 1/2}{C + |\mathcal{A}|/2}, \tag{2.5}$$

где C — количество безошибочных предсказаний автомата, в которых он был «уверен», с момента последней ошибки (или с начала слова, если ошибок ещё не было). Всем остальным символам $\mathcal{A}$ припишем одинаковые вероятности:

$$P_M(b|x_1, \ldots, x_i) = \frac{1/2}{C + |\mathcal{A}|/2}, \tag{2.6}$$

где $b \in \mathcal{A} \setminus \{a\}$.

Заметим, что если $C = 0$, то при вычислениях по 2.5–2.6 получим равномерное распределение вероятностей. Как только автомат обучится прогнозированию входного слова, C будет строго возрастать с увеличением i и вычисляемые по 2.5 вероятности будут стремиться к 1, а по 2.6 — к нулю.

Если автомат не знает, какой символ будет следующим (случай 2), то для каждого символа $a \in \mathcal{A}$ оценим его условную вероятность, используя контекстную модель нулевого порядка:

$$P_M(a|x_1, \ldots, x_i) = \frac{\nu_{x_1,\ldots,x_i}(a) + 1/2}{i + |\mathcal{A}|/2}, \tag{2.7}$$

где $\nu_{x_1,\ldots,x_i}(a)$ — количество вхождений буквы a в слово x_1,$x_2, \ldots, x_i$.

Поясним, что в 2.7 используется *предсказатель Кричевского* [81; 82], который обеспечивает минимальную ошибку для источника, у которого символы на выходе независимы и одинаково распределены (i.i.d source). 2.5–2.6 основаны

на этой же формуле, только в качестве числа вхождений символа, предсказанного автоматом, используется количество безошибочных предсказаний автомата, а частоты остальных символов принимаются нулевыми.

Далее, для того, чтобы автомат мог обнаружить конец слова, введём дополнительный символ $\triangleleft \notin \mathcal{A}$ и запишем на ленту последовательность $\alpha = \triangleright x_1 x_2 \dots x_n \triangleleft$. В листинге 2.1 приведён алгоритм для 10-головочного ИМДКА из [78] с описанными модификациями. В случаях, когда автомат может выдать прогноз для следующего символа на ленте (случай 1), этот прогноз явно указывается после слова «guessing» в описании алгоритма. Если прогнозное значение совпадает с фактическим следующим символом на ленте, то вероятность фактического символа оценивается по формуле 2.5, если нет — по 2.6. Если же автомат не может спрогнозировать следующий символ (случай 2), то вместо прогнозного символа в описании алгоритма указано CM0 (сокращение от Context Model) и вероятность фактического следующего символа на ленте оценивается по формуле 2.7.

В рамках данной работы была выполнена программная реализация описанного алгоритма. Приведём несколько примеров прогнозирования полилинейных слов с её помощью. Также попробуем объединить её с другими методами сжатия по формуле 1.7.

Для начала рассмотрим прогнозирование простого периодического слова 2.1. Для удобства будем называть компоненты $r_1^{a_1 n + b_1} r_2^{a_2 n + b_2} \dots r_m^{a_m n + b_m}$ полилинейных слов *блоками*. Например, для 2.1 одним блоком будет являться слово 000111. Будем использовать схему вычислений из параграфа 2.1: по слову из k блоков прогнозировать его $(k+1)$-й блок с помощью одношаговых прогнозов. Ясно, что после некоторого k автомат должен перестать делать ошибки, поэтому если выбрать k достаточно большим, сумма модулей ошибок при прогнозировании следующего блока должна быть равна нулю. Для сравнения, помимо автомата будем вычислять прогноз с помощью архиваторов ppmd и lcacomp, а также используя их комбинацию с автоматом по формуле 1.7 с равными весами. Результаты вычислений приведены в таблице 2.3. Как видно из этой таблицы, автомат и ppmd достаточно быстро начинают безошибочно прогнозировать эту последовательность, но автомат это делает несколько быстрее. При объединении методов наибольший вклад в результат вносит автомат, поскольку длины кодовых слов у него оказываются меньше. В результате комбинированный метод ведёт себя так же эффективно с точки зрения точности, как и автомат.

Листинг 2.1: Модифицированный алгоритм работы автомата, позволяющий оценить вероятность появления на выходе источника слова, записанного на ленту. Головки автомата обозначены как $h_1, h_2, h_3, h_{3a}, h_4, l, r, t, inner, outer$

loop
    TryMove(r), guessing by CM(0)
    Correction
    Matching

procedure Matching
    loop
        move h_{3a} **until** $h_{3a} = h_3$

        while $\alpha[h_1] = \alpha[h_2] =$
                $\alpha[h_3] = \alpha[h_4]$ **do**
            TryMove(h_1, h_2, h_{3a}, h_3)
            TryMove(h_4), guessing $\alpha[h_2]$
        break unless $\alpha[h_2] = \alpha[h_4]$

        while $\alpha[h_2] = \alpha[h_3] = \alpha[h_4]$ **do**
            TryMove(h_2, h_3)
            TryMove(h_4), guessing $\alpha[h_3]$
        break unless $\alpha[h_3] = \alpha[h_4]$

        while $\alpha[h_{3a}] = \alpha[h_3] = \alpha[h_4]$ **do**
            TryMove(h_{3a}, h_3)
            TryMove(h_4), guessing $\alpha[h_{3a}]$
        break unless $\alpha[h_{3a}] = \alpha[h_4]$

        while $h_{3a} \neq h_3$ **and** $\alpha[h_{3a}] = \alpha[h_4]$ **do**
            TryMove(h_{3a})
            TryMove(h_4), guessing $\alpha[h_{3a}]$
        break unless $\alpha[h_{3a}] = \alpha[h_4]$

procedure TryMove($\mathfrak{h}_1, \mathfrak{h}_2, \ldots, \mathfrak{h}_k$)
    for $\mathfrak{h} \in \{\mathfrak{h}_1, \mathfrak{h}_2, \ldots, \mathfrak{h}_k\}$ **do**
        move $\mathfrak{h}$
        if $\alpha[\mathfrak{h}] = \triangleleft$ **then**
            Exit

procedure Correction
    move h_1 **until** $h_1 = h_4$
    AdvanceOne(1)

    move h_2 **until** $h_2 = h_1$
    AdvanceMany(2)

    move h_3 **until** $h_3 = h_2$
    AdvanceMany(3)

    move h_4 **until** $h_4 = h_3$
    AdvanceMany(4)

procedure AdvanceMany(i)
    move *outer* **until** $outer = r$
    while $l \neq outer$ **do**
        AdvanceOne(i)
        TryMove(l, r)

procedure AdvanceOne(i)
    move t **until** $t = h_i$
    TryMove(h_i), guessing by CM(0)
    move $inner$ **until** $inner = r$
    while $l \neq inner$ **do**
        if $\alpha[t] = \alpha[h_i]$ **then**
            TryMove($l, r, outer$)
        else
            move $inner$ **until** $inner = r$
            TryMove(h_i), guessing by CM(0)
        TryMove(t)
        TryMove(h_i), guessing by CM(0)
    while $\alpha[t] = \alpha[h_i]$ **do**
        TryMove(t)
        TryMove(h_i), guessing $\alpha[t]$

Таблица 2.3 — Суммы ошибок одношаговых прогнозов для $(k+1)$-го блока слова 2.1, вычисленных с использованием k предшествующих блоков

k	Автомат	ppmd	lcacomp	Комбинация
3	7	4	7	7
4	0	10	12	0
5	0	2	12	0
6	0	0	16	0
7	0	0	19	0

Перейдём к рассмотрению более сложных с точки зрения прогнозирования слов и попробуем подать на вход тем же методам последовательность 2.3. В данном случае блоками будут её подпоследовательности вида 0^n1. Суммы ошибок при прогнозировании $(k+1)$-го блока по k блокам для разных k приведены в таблице 2.4. Видно, что после обработки 13 блоков автомат начинает безошибочно прогнозировать следующий блок, и комбинированный метод выдаёт аналогичный результат. При этом оба используемых архиватора продолжают ошибаться.

Таблица 2.4 — Суммы ошибок одношаговых прогнозов для $(k+1)$-го блока слова 2.3, вычисленных с использованием k предшествующих блоков

k	Автомат	ppmd	lcacomp	Комбинация
5	1	1	1	1
12	1	1	1	1
13	0	2	5	0
14	0	1	6	0
30	0	1	18	0

Наконец, попробуем построить прогнозы для более сложного полилинейного слова. Например, пусть оно задаётся выражением

$$20\prod_{n\geqslant 0}(012)^{n+3}(01)^n0^2. \tag{2.8}$$

Как и в предыдущем случае, ожидается, что рано или поздно автомат и комбинированный алгоритм начнут безошибочно прогнозировать следующий блок по k предыдущим, в то время как архиваторы продолжат ошибаться. Результаты вычислений приведены в таблице 2.5. Из этой таблицы видно, что в данном случае

комбинированный метод поначалу ведёт себя как ppmd, но с увеличением длины прогнозируемой последовательности начинает вести себя как автомат.

Таблица 2.5 — Суммы ошибок одношаговых прогнозов для $(k+1)$-го блока слова 2.8, вычисленных с использованием k предшествующих блоков

k	Автомат	ppmd	lcacomp	Комбинация
10	13	16	38	16
20	0	30	64	30
30	0	4	129	4
40	0	16	169	0
50	0	24	209	0

Анализируя таблицы 2.3–2.5, можно сделать вывод, что автомат действительно безошибочно прогнозирует полилинейные слова, и с увеличением длины слова точность комбинированного метода становится аналогичной точности автомата.

Глава 3. Адаптивный метод прогнозирования

3.1 Мотивация разработки метода

В параграфе 1.4 был предложен способ объединения различных методов сжатия в один метод прогнозирования. Его очевидным недостатком является высокая трудоёмкость — все последовательности, возникающие при прогнозировании, должны быть сжаты всеми используемыми архиваторами. В данной главе опишем метод сокращения трудоёмкости вычислений, практически не приводящий к потери точности получаемых прогнозов. Назовём его *адаптивным методом*. Также здесь предлагается способ преобразования произвольных методов прогнозирования к методам сжатия данных, что позволяет объединять их с архиваторами и автоматом по формуле 1.7, с возможным использованием предлагаемого адаптивного метода. Получающийся в результате комплекс методов прогнозирования способен использовать сильные стороны отдельных методов, входящих в его состав, и является более универсальным по сравнению с ними.

3.2 Описание метода

Для ускорения вычислений в данной работе предлагается использовать адаптивные по времени коды, которые были описаны в [83]. Рассмотрим сначала, как эти коды позволяют сократить время вычислений при сжатии данных. Предположим, что задано некоторое конечное множество методов сжатия $F = \{\varphi_1, \varphi_2, \ldots, \varphi_k\}$ и требуется сжать последовательность $X = x_1,x_2,\ldots,x_t$ символов из конечного алфавита $\mathcal{A}$, используя для этого $\varphi \in F$, обеспечивающий максимальное сжатие. Ясно, что можно попробовать сжать X всеми алгоритмами из F с целью поиска наилучшего алгоритма φ_s, и затем вместе с сжатым представлением X хранить номер s выбранного алгоритма для того, чтобы поддержать возможность правильного декодирования X. Поскольку всего в заданном множестве существует k номеров алгоритмов и двоичное представление любого номера занимает не более $\lceil \log_2 k \rceil$ бит, получающееся в результате кодовое слово будет

иметь длину $\lceil \log_2 k \rceil + |\varphi_s(X)|$ бит. Серьёзным недостатком подобного «наивного» подхода является большое количество времени, затрачиваемого на поиск φ_s.

В основе адаптивных по времени кодов лежит следующая идея. Вместо того, чтобы сжимать $X = x_1, x_2, \ldots, x_t$ с помощью всех архиваторов из F, можно сжать только небольшой фрагмент X, например, префикс $x_1, x_2, \ldots, x_r$, $r << t$, всеми архиваторами для выбора оптимального $\hat{\varphi}_s$ на этом фрагменте, и затем сжать X целиком только с его помощью. Ясно, что при этом φ_s и $\hat{\varphi}_s$ могут не совпасть, поэтому будем говорить, что $\hat{\varphi}_s$ является *близким к* оптимальному для X. Обозначим через v_i время, которое требуется φ_i для сжатия одной буквы X, пусть $v = \max\limits_{i \in \{1,2,\ldots,k\}} \{v_i\}$. Тогда времени $T = tv$ будет достаточно для того, чтобы сжать X любым методом из F. Обозначим через δT дополнительное время, которое допустимо потратить на поиск $\hat{\varphi}_s$, где δ — положительная константа. В таком случае суммарное время на выбор алгоритма и сжатие данных будет ограничено сверху величиной $\hat{T} = T + \delta T = T(1 + \delta)$. Любой метод сжатия, который следует этой схеме, называется *адаптивным по времени* [83] и обозначается как $\hat{\Phi}^{\delta}_{compr}$. Если для адаптивного по времени метода (или кода) выполняется условие

$$\lim_{t \to \infty} \frac{|\hat{\Phi}^{\delta}_{compr}(x_1, x_2, \ldots, x_t)|}{t} = \min_{i \in \{1,2,\ldots,k\}} \lim_{t \to \infty} \frac{|\varphi_i(x_1, x_2, \ldots, x_t)|}{t},$$

то этот метод называется *универсальным по времени*.

В [83] был предложен следующий простой универсальный по времени метод сжатия:

1. Вычисляем $r = \lfloor \delta T / kv \rfloor$;
2. Сжимаем $x_1, x_2, \ldots, x_r$ всеми $\varphi \in F$, затем находим $\hat{\varphi}_s \in F$, обеспечивающий наилучшее сжатие;
3. Сжимаем $x_1, x_2, \ldots, x_t$ с помощью $\hat{\varphi}_s$;
4. Формируем кодовое слово $< s > \hat{\varphi}_s(x_1, x_2, \ldots, x_t)$, где $< s >$ — двоичное представление s.

В данной работе этот алгоритм применяется для целей прогнозирования. Заметим, что поскольку для выбора близкого к оптимальному архиватора используется только префикс входной последовательности, для всех $|\mathcal{A}|^h$ временных рядов, получаемых при вычислениях по 1.4, он будет одинаковым. Поэтому можно сжимать только префикс исходного ряда X, что позволяет сократить время вычислений даже при $r = t$. Это происходит за счёт того, что вместо сжатия $|\mathcal{A}|^h$ временных рядов с помощью k архиваторов и последующего объединения

результатов по формуле 1.7 теперь сжимается только один временной ряд с использованием всех φ из F, а затем $|\mathcal{A}|^h$ рядов лишь с помощью $\hat{\varphi}_s$.

Добавим, что в адаптивном методе можно выбрать не один, а несколько лучших архиваторов, и затем объединить их в один метод прогнозирования по формуле 1.7.

В качестве примера исследуем эффективность адаптивного метода при прогнозировании последовательности

$$X = 123212321232123212321232123212. \tag{3.1}$$

Пусть $F = \{\text{zlib}, \text{bzip2}, \text{zstd}\}$ (краткое описание этих архиваторов приведено в параграфе 4.1). Рассматриваемая последовательность содержит $t = 30$ элементов, будем использовать для выбора близкого к оптимальному архиватора первые $r = 10$ из них. Поскольку $T = tv$ и $r = \lfloor \delta T / kv \rfloor$ получаем, что $r = \lfloor \delta t / k \rfloor$ и, следовательно, $\delta = 1$. Сожмём 10 первых значений X с помощью каждого $\varphi \in F$. Для zlib размер сжатого представления указанной подпоследовательности получился равным 112 бит, для bzip2 — 320 бит, для zstd — 152 бита. Согласно описанному алгоритму, можно построить прогноз на требуемое количество шагов вперёд используя только zlib. Построим прогнозы на 2 шага для второй половины X, используя схему вычислений, аналогичную той, что была описана в параграфе 2.1: по $x_1, x_2, \ldots, x_{15}$ построим прогноз для x_{16} и x_{17}, затем на основе $x_1, x_2, \ldots, x_{16}$ построим прогноз для x_{17} и x_{18} и так далее. Средние абсолютные ошибки (MAE) для всех трёх методов сжатия и их комбинации приведены в таблице 3.1.

Таблица 3.1 — Средние абсолютные ошибки, полученные при прогнозировании второй половины последовательности 3.1 с помощью различных методов сжатия

Номер шага	1	2
zlib	0	0
bzip2	0.43	0.57
zstd	0	0
Комбинация	0	0

Как видно из таблицы 3.1, в данном случае действительно не теряем в точности, используя для прогнозирования только zlib.

3.3 Преобразование методов прогнозирования к методам сжатия данных

Как отмечалось в параграфе 1.2, в настоящее время существует большое количество разнообразных методов прогнозирования и заранее неизвестно, какой метод окажется наиболее точным при прогнозировании определённого временного ряда. В данном параграфе предлагается способ преобразования произвольных методов прогнозирования к методам сжатия данных, что позволит использовать их в формуле 1.7, а также в адаптивном методе из предыдущего параграфа.

Опишем предлагаемое преобразование. Пусть дан некоторый метод прогнозирования π. По последовательности чисел $Y = y_1, y_2, \ldots, y_t$ метод π должен быть способен выдавать прогнозное значение $\hat{y}_{t+1|t}$ для следующего члена последовательности: $\hat{y}_{t+1|t} = \pi(y_1, y_2, \ldots, y_t)$. Если Y — вещественный временной ряд, то применим к нему процедуру квантования из параграфа 1.3 и получим целочисленный ряд X, элементы которого принадлежат алфавиту $\mathcal{A} = \{0, 1, \ldots, n-1\}$, где n — количество интервалов при квантовании. Если же Y — целочисленный ряд, то какие-либо преобразования выполнять не обязательно, т.е. $X = Y$, а $\mathcal{A}$ — множество всех значений X. Как и в случае с автоматом, требуется получить оценку вероятности $P_\pi(x_1, x_2, \ldots, x_t)$ появления X на выходе источника с помощью π, поскольку затем эту вероятность можно будет конвертировать в длину кодового слова как $-\log_2 P_\pi(X)$. Как и ранее, будем вычислять вероятность появления слова X через произведение условных вероятностей его букв:

$$P_\pi(x_1, x_2, \ldots, x_t) = \prod_{i=1}^{t} P_\pi(x_i | x_1, x_2, \ldots, x_{i-1}), \tag{3.2}$$

причём для первого символа будем считать $P_\pi(x_1 | \ldots) = P_\pi(x_1) = 1/|\mathcal{A}|$.

Условные вероятности с помощью π будем вычислять следующим образом. Если $\pi(x_1, x_2, \ldots, x_{i-1})$, округлённое до ближайшего символа алфавита $\mathcal{A}$, совпадает с x_i, то

$$P_\pi(x_i | x_1, x_2, \ldots, x_{i-1}) = \frac{(i-1) + 1/2}{(i-1) + |\mathcal{A}|/2}.$$

В противном случае,

$$P_\pi(x_i | x_1, x_2, \ldots, x_{i-1}) = \frac{1/2}{(i-1) + |\mathcal{A}|/2}.$$

Сделаем допущение, что π необходимо знать более одного предыдущего значения для того, чтобы дать прогноз следующего символа. Это означает, что несколько первых условных вероятностей в 3.2 вычислить не получится. Пусть π достаточно i_0 значений для того, чтобы начать делать прогнозы. Тогда вероятность для первых i_0 букв вычислим с помощью предсказателя Кричевского:

$$K(x_1, x_2, \ldots, x_{i_0}) = \prod_{i=1}^{i_0} \frac{\nu_{x_1,\ldots,x_{i-1}}(x_i) + 1/2}{(i-1) + |\mathcal{A}|/2},$$

где $\nu_{x_1,\ldots x_{i-1}}(x_i)$ — количество вхождений значения x_i в слово x_1,$x_2, \ldots, x_{i-1}$. После этого формула 3.2 примет следущий вид:

$$P_\pi(x_1, x_2, \ldots, x_t) = K(x_1, x_2, \ldots, x_{i_0}) \prod_{i=i_0+1}^{t} P_\pi(x_i | x_1, x_2, \ldots, x_{i-1}). \qquad (3.3)$$

В данной работе описанный подход был реализован для аддитивной модели Хольта-Уинтерса (Holt-Winters) [11], которая относится к методам экспоненциального сглаживания и позволяет учесть тренд и сезонность. Предположим, что $Y = y_1, y_2, \ldots, y_t$ — вещественный временной ряд, и требуется вычислить $\hat{y}_{t+h|t}$. В модели Хольта-Уинтерса предполагается, что каждое значение ряда содержит три составляющие: уровень l, тренд b и сезонную составляющую s. Согласно этой модели, прогнозное значение вычисляется по следующей формуле:

$$\begin{aligned}
\hat{y}_{t+h|t} &= l_t + hb_t + s_{t+h-m(k+1)}, \\
l_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}), \\
b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}, \\
s_t &= \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},
\end{aligned}$$

где l_i, b_i, s_i — значения соответствующих компонент в момент времени i, α, β, γ — коэффициенты, которые нужно оценить, m — длина цикла в сезонной составляющей (например, для ежемесячных данных $m = 12$), k — целая часть от $(h-1)/m$. Существует несколько разных подходов к тому, как выбрать начальные значения для l, b и s. Приведём для примера относительно простой вариант, предложенный в [84]:

$$\begin{aligned}
l_m &= (y_1 + y_2 + \ldots + y_m)/m, \\
b_m &= [(y_{m+1} + y_{m+2} + \ldots + y_{m+m}) - (y_1 + y_2 + \ldots + y_m)]/m^2, \\
s_i &= y_i - l_m,\ i = 1, 2, \ldots, m.
\end{aligned}$$

Другой, несколько более сложный подход к оценке начальных значений, может быть найден в [17].

Рассмотрим пример. Вычислим оценку вероятности для последовательности

$$X = 1, 2, 3, 3, 2, 2, 3, 1$$

с помощью модели Хольта-Уинтерса. Для этого будем использовать программную реализацию этой модели из пакета statsmodels для Python, её описание доступно по ссылке `https://www.statsmodels.org/dev/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html`. Для оценки параметров модели воспользуемся автоматической процедурой, реализованной в данном пакете, максимизирующей логарифм функции правдоподобия. Фрагмент кода, с помощью которого будем вычислять $\pi(x_1, x_2, \ldots, x_{i-1})$, приведён в листинге 3.1. В этом фрагменте выбрана произвольная дата в качестве начала отсчёта и считается, что данные являются ежедневными с длиной периода в 1 день.

Листинг 3.1: Программа для вычисления прогнозов на 1 шаг вперёд с помощью модели Хольта-Уинтерса с описанной модификацией

```
import pandas as pd
from statsmodels.tsa.api import ExponentialSmoothing

def bound(val: int, min_: int, max_: int) -> int:
    if val < min_:
        return min_
    if val > max_:
        return max_
    return val

# Загружаем временной ряд как список в переменную ts.

arbitrary_date = "1/1/2000"
index = pd.date_range(start=arbitrary_date, periods=len(ts), freq='D')
pd_ts = pd.Series(ts, index)
fit = ExponentialSmoothing(pd_ts, trend='add').fit(optimized=True)

prediction = bound(int(round(fit.forecast(1).values[0])),
    min(ts), max(ts))
```

Для первых четырёх символов будем оценивать вероятности с помощью предсказателя Кричевского (т.е. $i_0 = 4$), а для остальных — с помощью модели Хольта-Уинтерса. Алфавитом $\mathcal{A}$ в данном случае является множество $\{1, 2, 3\}$. Получаем

$$K(1233) = \frac{0+1/2}{0+3/2} \cdot \frac{0+1/2}{1+3/2} \cdot \frac{0+1/2}{2+3/2} \cdot \frac{1+1/2}{3+3/2} \approx 0.003175.$$

Модель Хольта-Уинтерса генерирует прогнозные значения, приведённые в таблице 3.2.

Таблица 3.2 — Прогнозные значения для ряда X, полученные с помощью модели Хольта-Уинтерса

Элемент x_i	1	2	3	3	2	2	3	1
$\pi(x_1, \ldots, x_{i-1})$	-	-	-	-	2	1	2	2

В результате получаем

$$P_\pi(12332231) = K(1232) \cdot \frac{4+1/2}{4+3/2} \cdot \frac{1/2}{5+3/2} \cdot \frac{1/2}{6+3/2} \cdot \frac{1/2}{7+3/2} \approx 7.84 \times 10^{-7}.$$

Глава 4. Экспериментальное исследование

В данной главе предложенные ранее методы применяются для прогнозирования реальных временных рядов. Сначала описывается используемая методология вычислений, затем строятся прогнозы для некоторых физических, социальных и экономических показателей.

4.1 Методология

Начнём главу с описания схемы вычислений, которая использовалась в экспериментальном исследовании разработанных методов. Поскольку применяемый подход позволяет использовать различные программы сжатия данных для построения прогнозов, были выбраны следующие 7 программ, в которых реализованы разные подходы к сжатию:

1. *zlib* — библиотека для сжатия данных, в которой реализована схема DEFLATE, представляющая собой комбинацию алгоритма LZ77 [47] и кода Хаффмана. Адрес сайта проекта: `https://zlib.net/`;
2. *bzip2* — программа для сжатия данных, в которой реализованы преобразование Барроуза-Уилера (Burrows–Wheeler transform, BWT) [50] и код Хаффмана. Доступна по адресу `https://www.sourceware.org/bzip2`;
3. *ppmd* — вариант алгоритма предсказания по частичному совпадению (prediction by partial matching, PPM) [53]. В данной работе использовалась реализация, исходный код которой может быть найден по адресу `https://github.com/Shelwien/ppmd_sh`;
4. *rp* — алгоритм сжатия данных на основе контекстно-свободных грамматик. Была использована реализация [77], доступная по адресу `https://github.com/nicolaprezza/Re-Pair`;
5. *lcacomp* — алгоритм сжатия данных на основе контекстно-свободных грамматик, описание которого может быть найдено в [68], а программная реализация доступна по ссылке `https://code.google.com/archive/p/lcacomp/`;

6. *zstd* — алгоритм сжатия данных, разработанный компанией Facebook. Исходный код архиватора доступен на github: `https://github.com/facebook/zstd`;
7. *zpaq* — архиватор, который позволяет инкрементально добавлять файлы в архив. В нём реализован набор различных методов сжатия, таких как LZ-77, BWT, CM (Context Mixing, смешивание контекстов). Сайт проекта: `http://mattmahoney.net/dc/zpaq.html`.

Помимо этих архиваторов, был использован автомат из параграфа 2.2 (далее в данной главе будем называть метод прогнозирования на основе автомата как *automaton*), а также, в некоторых вычислениях, модель Хольта-Уинтерса с модификацией из параграфа 3.3. При объединении алгоритмов по формуле 1.7, а также прогнозов, полученных с использованием различного числа интервалов при квантовании в формуле 1.6, были использованы равные веса, т.е. априорно не отдавалось предпочтение какому-либо алгоритму или разбиению. Для представления одного элемента временного ряда в памяти использовался 1 байт.

Для удаления трендов применялось взятие первой разности, а иногда и второй (в таких случаях будем это явно указывать). Также для уменьшения влияния выбросов использовалось сглаживание:

$$\bar{x}_i = (2x_i + x_{i-1} + x_{i-2})/4. \tag{4.1}$$

Помимо самих прогнозных значений, далее для них иногда приводятся доверительные интервалы. Для их построения вычислялись прогнозы для уже известных значений соответствующих рядов и затем оценивались стандартные отклонения для ошибок построенных прогнозов (отдельно для каждого шага). Предположим, что рассматривается временной ряд $X = x_1, x_2, \ldots, x_t$ и требуется построить прогноз на h шагов вперёд. Выберем некоторое значение $s \in \{2, 3, \ldots, t-h\}$. Затем для каждого ряда

$$X_s, X_{s+1}, \ldots, X_{t-h}, \tag{4.2}$$

где $X_i = x_1, x_2, \ldots, x_i$, построим прогноз на h шагов вперёд $\hat{X}_i^h = \hat{x}_{i+1|i}, \hat{x}_{i+2|i}, \ldots, \hat{x}_{i+h|i}$ и найдём ошибки для каждого шага: $r_{ij} = x_{i+j} - \hat{x}_{i+j|i}$, $j \in \{1, 2, \ldots, h\}$. После этого вычислим стандартные отклонения для ошибок: $\sigma_j = \sqrt{\frac{1}{t-h-s+1} \sum_{i=s}^{t-h} (r_{ij} - \bar{r}_j)^2}$, где $\bar{r}_j = (\sum_{i=s}^{t-h} r_{ij})/(t-h-s+1)$. Доверительный интервал для x_{t+j} выберем равным $[\hat{x}_{t+j|t} - 2\sigma_j; \hat{x}_{t+j|t} + 2\sigma_j]$, что приблизительно соответствует доверительной вероятности 0.954 для нормального распределения.

Для оценки точности построенных прогнозов чаще всего использовались средние абсолютные ошибки (mean absolute error, MAE) для каждого шага $j \in \{1, 2, \ldots, h\}$, вычисляемые как $\frac{1}{t-h-s+1} \sum_{i=s}^{t-h} |r_{ij}|$. Реже использовались средние относительные ошибки, которые находились как $\frac{1}{t-h-s+1} \sum_{i=s}^{t-h} \left| \frac{r_{ij}}{x_{i+j}} \right|$.

Рассмотрим пример. Предположим, что требуется построить прогноз на 2 шага вперёд для ряда

$$X = 1.1, 1.2, 1.3, 1.2, 1.4.$$

Пусть $s = 2$. Тогда в соответствии с изложенной выше схемой вычислений, нужно построить прогнозы на 2 шага для рядов

$$X_2 = 1.1, 1.2$$

и

$$X_3 = 1.1, 1.2, 1.3.$$

Пусть вычисленный прогноз для X_2 равен 1.3, 1.4, а для X_3 — 1.4, 1.5. Тогда стандартные отклонения для шагов 1 и 2 равны

$$\sigma_1 = \sqrt{((0 + 0.1)^2 + (-0.2 + 0.1)^2)/2} = 0.1$$

и

$$\sigma_2 = \sqrt{((-0.2 + 0.15)^2 + (-0.1 + 0.15)^2)/2} = 0.05.$$

Пусть вычисленный прогноз для X равен 1.3, 1.5, тогда доверительный интервал для первого шага получим как $[1.3 - 2 \cdot 0.1; 1.3 + 2 \cdot 0.1] = [1.1; 1.5]$, для второго шага — $[1.5 - 2 \cdot 0.05; 1.5 + 2 \cdot 0.05] = [1.4; 1.6]$.

Поскольку количество сжимаемых последовательностей экспоненциально возрастает с числом шагов, на которое строится прогноз, при прогнозировании на большое количество шагов вперёд (больше 4-х в данной работе) с целью сокращения трудоёмкости вычислений использовался следующий подход. Предположим для простоты, что количество элементов t в прогнозируемом временном ряде и число шагов h, на которое строится прогноз, являются чётными числами. Разобъём временной ряд X на два временных ряда $X_{\text{odd}} = x_1, x_3, \ldots, x_{t-1}$ и $X_{\text{even}} = x_2, x_4, \ldots, x_t$. Затем будем отдельно прогнозировать $x_{t+1}, x_{t+3}, \ldots, x_{t+h-1}$ по X_{odd} и $x_{t+2}, x_{t+4}, \ldots, x_{t+h}$ по X_{even}. Таким образом, количество последовательностей, которые необходимо сжать (без учёта использования нескольких разбиений при

квантовании), сократится с $|\mathcal{A}|^h$ до $2|\mathcal{A}|^{h/2}$. Ясно, что аналогично можно разбить X на 3, 4 и т.д. временных ряда, а также воспользоваться данным подходом если t и/или h являются нечётными числами. В этих случаях длины получаемых временных рядов могут различаться, как и количества шагов, на которые по ним строятся прогнозы. Очевидным недостатком подобного подхода является снижение точности, поскольку при декомпозиции одного ряда на несколько отдельных временных рядов некоторые закономерности могут быть утрачены.

4.2 Прогнозирование временных рядов из M3 Competition

Перейдём к описанию результатов вычислений на реальных временных рядах, проведённых с использованием предлагаемых методов. Начнём с описания результатов прогнозирования рядов из соревнования M3 Competition (сокращённо M3C), которое было проведено в 2000 году и основной целью которого являлось сравнение точности различных методов прогнозирования на реальных данных [19]. В этом соревновании были представлены 3003 временных ряда, длина которых составляла от 14 до 144 наблюдений. Присутствовали ежегодные, ежемесячные и ежеквартальные данные, а также небольшое количество рядов, не относящихся ни к одной из вышеперечисленных категорий (в [19] эта категория данных обозначена как «другие»). Каждому участнику соревнования требовалось с помощью метода, в котором у него есть экспертные знания, построить прогноз на 6 значений вперёд для ежегодных данных, на 8 значений вперёд для ежеквартальных данных и данных из категории «другие», и на 18 значений вперёд для ежемесячных данных. В итоге были получены прогнозы, построенные с помощью 24 методов прогнозирования, точность которых была оценена организаторами соревнования. На веб-сайте Международного института прогнозистов (`https://forecasters.org/resources/time-series-data/m3-competition/`) размещены временные ряды из M3C вместе с их реальными значениями за периоды времени, на которые требовалось построить прогнозы (эти значения не были доступны участникам соревнования, но использовались организаторами для оценки точности). Поэтому сейчас имеется возможность построить аналогичные прогнозы с использованием предлагаемых методов и сравнить их точность с точностью методов из M3C.

В данном параграфе для оценки точности прогнозов будем использовать симметричные средние абсолютные процентные ошибки (symmetric mean absolute percentage error, sMAPE). Предположим, что прогнозы на h шагов вперёд были построены для временных рядов $X_1, X_2, \ldots, X_n$, каждый из которых имеет вид $X_i = x_{i1}, x_{i2}, \ldots, x_{it_i}, i \in \{1, 2, \ldots, n\}$. Тогда sMAPE для шага $j \in \{1, 2, \ldots, h\}$ определяется как

$$\text{sMAPE}(j) = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{x}_{i(t_i+j)|t_i} - x_{i(t_i+j)}|}{(\hat{x}_{i(t_i+j)|t_i} + x_{i(t_i+j)})/2} \cdot 100.$$

Поясним, что sMAPE использовалась в M3C и её значения для участвовавших методов опубликованы в [19]. Поскольку все временные ряды из M3C не содержат отрицательных значений, данная величина не может быть отрицательной.

В вычислениях использовались три метода сжатия: zlib, ppmd и rp, а также их комбинация. Заметим, что данные алгоритмы основаны на разных принципах (в zlib используется алгоритм LZ-77 совместно с кодом Хаффмана, в ppmd реализован алгоритм предсказания по частичному совпадению, а rp сжимает данные за счёт построения компактной КС-грамматики). Также был использован метод STL [85] для выделения сезонной составляющей, которая принималась постоянной. К данным была применена процедура сглаживания по формуле 4.1. Кроме того, для всех категорий временных рядов осуществлялся переход к первой разности. В процессе экспериментального исследования была предпринята попытка выбора такого порядка разности, при котором среднеквадратичное отклонение для ряда будет наименьшим. За исключением одного случая (ежемесячные данные), такой подход приводил к небольшому ухудшению точности прогноза по сравнению с постоянным выбором первой разности.

Добавим, что при построении всех прогнозов применялась описанная в параграфе 4.1 процедура уменьшения трудоёмкости вычислений за счёт декомпозиции исходного временного ряда на несколько рядов. При прогнозировании ежегодных, ежеквартальных и «других» данных каждый ряд разбивался на два ряда, для ежемесячных — на шесть рядов.

Результаты вычислений приведены в таблицах 4.1–4.4. Формат этих таблиц позаимствован из [19], но с тем отличием, что в них не приводятся значения ошибок по отдельным методам, участвовавшим в соревновании. Вместо этого, в

строках «Лучший M3C» и «Худший M3C» содержатся, соответственно, наименьшие и наибольшие средние ошибки для каждого шага среди всех участвовавших в M3C методов. В строках, соответствующих методам сжатия, рядом с названиями методов приводятся максимальные количества интервалов, использовавшиеся при квантовании. Отметим, что при этом во всех случаях вычисления проводились по формуле 1.6, т.е. если возле названия метода указано «16 интервалов», это означает, что были использованы разбиения на 2, 4, 8 и 16 интервалов, при этом в 1.6 веса ω_i для них были одинаковыми. Комбинация методов zlib, ppmd и rp, полученная по 1.7, обозначена в таблицах как zlib+ppmd+rp.

Анализируя представленные таблицы, можно сказать, что при прогнозировании ежегодных данных и данных из категории «другие» метод на основе архиваторов показал точность, сравнимую с точностью методов, участвовавших в соревновании. На ежеквартальных и ежемесячных данных его точность оказалась сравнительно низкой но, тем не менее, при прогнозировании первых четырёх значений рядов сопоставимой с методами из M3 Competition. При этом увеличение максимального количества интервалов при квантовании с 16 до 32 ни в одном из случаев не привело к существенному повышению точности прогнозов. Точность комбинированного метода во всех случаях практически не уступает точности лучшего из отдельных методов.

Таблица 4.1 — Результаты прогнозирования ежегодных данных из M3 Competition

Метод	sMAPE для номера шага h						Среднее		Кол-во рядов
	1	2	3	4	5	6	1–4	1–6	
Лучший M3C	7.6	12.1	16.1	18.2	20.8	22.7	13.65	16.42	645
Худший M3C	10.7	15.2	20.8	24.1	28.1	31.2	17.57	21.59	645
zlib (16 интервалов)	9.9	14.9	20.8	22.9	28.0	28.2	17.13	20.79	645
ppmd (16 интервалов)	10.1	14.5	20.6	22.4	27.3	27.2	16.76	20.25	645
rp (16 интервалов)	10.5	14.9	19.5	21.9	26.2	27.4	16.70	20.06	645
zlib+ppmd+rp (16 интервалов)	10.4	14.5	19.4	21.8	26.5	27.3	16.51	19.97	645
zlib+ppmd+rp (32 интервала)	10.3	14.5	19.3	21.8	26.4	27.3	16.49	19.95	645

Таблица 4.2 — Результаты прогнозирования ежеквартальных данных из M3 Competition

Метод	sMAPE для номера шага h							Среднее			Кол-во ря-дов
	1	2	3	4	5	6	8	1–4	1–6	1–8	
Лучший M3C	4.8	6.6	7.4	8.8	9.4	10.9	12.0	7.0	8.04	8.96	756
Худший M3C	7.7	8.9	9.1	10.7	11.8	13.7	15.7	8.86	9.6	10.96	756
zlib (16 интерв.)	5.8	7.6	9.0	11.5	14.0	14.4	17.0	8.47	10.39	12.02	756
ppmd (16 интерв.)	5.8	7.5	8.8	10.6	12.7	13.3	15.4	8.16	9.77	11.12	756
rp (16 интервалов)	6.5	9.0	10.0	12.0	13.5	13.6	15.2	9.35	10.75	11.96	756
zlib+ppmd+rp (16 интервалов)	5.8	7.5	8.8	10.5	13.0	13.2	15.0	8.16	9.80	11.11	756
zlib+ppmd+rp (32 интервала)	5.8	7.5	8.8	10.5	13.0	13.2	14.9	8.16	9.81	11.11	756

Таблица 4.3 — Результаты прогнозирования ежемесячных данных из M3 Competition

Метод	sMAPE для номера шага h										Среднее			Кол-во ря-дов
	1	2	3	4	5	6	8	12	15	18	1–4	1–8	1–18	
Лучший M3C	11.2	10.7	11.7	12.4	11.8	12.2	12.6	13.2	16.2	18.2	11.54	12.06	13.85	1428
Худший M3C	15.3	13.8	15.7	17.0	15.3	15.6	17.4	17.5	22.2	24.3	15.39	15.89	18.4	1428
zlib (16 интервалов)	14.4	14.9	17.3	19.4	20.9	17.9	20.4	19.0	25.6	26.5	16.51	18.42	21.23	1428
ppmd (16 интервалов)	14.0	14.1	15.7	20.2	16.8	20.6	17.7	18.0	24.6	25.5	17.23	18.76	19.95	1428
rp (16 интерв.)	15.6	15.7	17.9	19.7	21.2	18.5	19.1	20.7	26.4	26.9	15.44	17.26	21.55	1428
zlib+ppmd+rp (16 интервалов)	14.0	14.1	15.8	19.6	21.2	18.3	19.1	20.6	25.9	26.7	15.86	18.02	21.13	1428
zlib+ppmd+rp (32 интервала)	14.0	14.1	15.8	19.6	21.2	18.4	19.1	20.6	26.0	26.7	15.86	18.02	21.13	1428
ppmd (16 интервалов, подбор поряд-ка разности)	13.4	13.2	14.7	17.5	19.1	16.4	17.5	17.6	22.3	24.4	14.70	16.52	18.87	1428

Таблица 4.4 — Результаты прогнозирования прочих (other) данных из M3 Competition

Метод	sMAPE для номера шага h							Среднее			Кол-во ря-дов
	1	2	3	4	5	6	8	1–4	1–6	1–8	
Лучший M3C	1.6	2.7	3.8	4.3	5.3	5.1	6.0	3.17	3.86	4.38	174
Худший M3C	2.7	3.8	5.4	6.3	7.8	7.6	9.2	4.38	5.49	6.3	174
zlib (16 интервалов)	2.5	3.4	4.9	6.3	8.3	7.7	8.9	4.25	5.49	6.33	174
ppmd (16 интервалов)	2.4	3.2	4.7	5.1	7.1	6.9	8.2	3.85	4.90	5.68	174
rp (16 интервалов)	2.7	3.9	5.8	6.9	8.0	8.5	10.3	4.84	5.97	6.87	174
zlib+ppmd+rp (16 интервалов)	2.4	3.2	4.7	5.1	7.3	6.8	8.1	3.85	4.91	5.70	174
zlib+ppmd+rp (32 интервала)	2.4	3.2	4.7	5.0	7.2	6.8	8.1	3.84	4.90	5.69	174

4.3 Прогнозирование физических данных

Перейдём к прогнозированию физических данных, и для начала рассмотрим временной ряд солнечных пятен. Под *солнечными пятнами* понимают временные области на Солнце, которые холоднее и выглядят темнее, чем окружающая их поверхность. Количество и размер солнечных пятен характеризуют 11-летние циклы солнечной активности [86], которые, в свою очередь, оказывают существенное влияние на жизнь на Земле. Например, с увеличением солнечной активности возрастают интенсивности ультрафиолетового и рентгеновского излучений, исходящих от Солнца, что приводит к повышению температуры и плотности верхних слоёв атмосферы. Помимо циклов, многие другие характеристики солнечной активности связаны с пятнами. К ним относятся поток радиоизлучения с длиной волны 10.7 см (the 10.7 cm radio flux), полное солнечное излучение (TSI, the total solar irradiance — мера солнечной энергии по

всем длинам волн на единицу площади, падающей на верхние слои атмосферы Земли) и другие.

Хотя о солнечных пятнах было известно в Китае ещё около 2000 лет назад, в Европе до 17 века записи об их наблюдениях практически не встречаются. Исключениями являются две записи, сделанные в России в 14 веке во времена лесных пожаров [86; 87]. Систематические ежедневные наблюдения пятен проводятся с 1849 года благодаря деятельности швейцарского астронома Дж. Р. Вольфа. Вольф предложил подсчитывать количество групп солнечных пятен и затем вычислять показатель

$$R = k(10g + n),$$

где k — корректирующий коэффициент, специфичный для наблюдателя и телескопа, g — количество групп солнечных пятен, n — количество отдельных солнечных пятен. Значения R часто называют *числами Вольфа* или *международными числами солнечных пятен*. Несмотря на то, что возможности современного оборудования позволяют точно подсчитать количество пятен на Солнце, наблюдения до сих пор ведутся «на глаз» по старой методике. Причинами этого являются короткая история наблюдений с использованием современных средств, а также уникальность и ограниченность времени эксплуатации приборов. Из-за этого наблюдения с использованием современной техники ограничиваются одним или, в лучшем случае, несколькими солнечными циклами, в то время как наблюдения с помощью телескопа существуют со времён Галилея, т.е. более 400 лет [88].

Добавим, что Вольф восстановил значения солнечных пятен за предшествующие 100 лет, до 1749 года. Несмотря на то, что он использовал данные от нескольких наблюдателей, значительная часть значений отсутствовала, и поэтому ему пришлось использовать приближённые значения, вычисленные на основе измерений геомагнитной активности. Ясно, что это сказалось на точности, поэтому данные с 1749 года по 1849 год считаются менее надёжными [86]. В настоящее время вычислением международного числа солнечных пятен занимается Центр анализа данных солнечного влияния (Solar Influences Data Analysis Center, SIDC) [88], являющийся подразделением Королевской обсерватории Бельгии. График среднемесячных несглаженных чисел солнечных пятен приведён на рисунке 4.1.

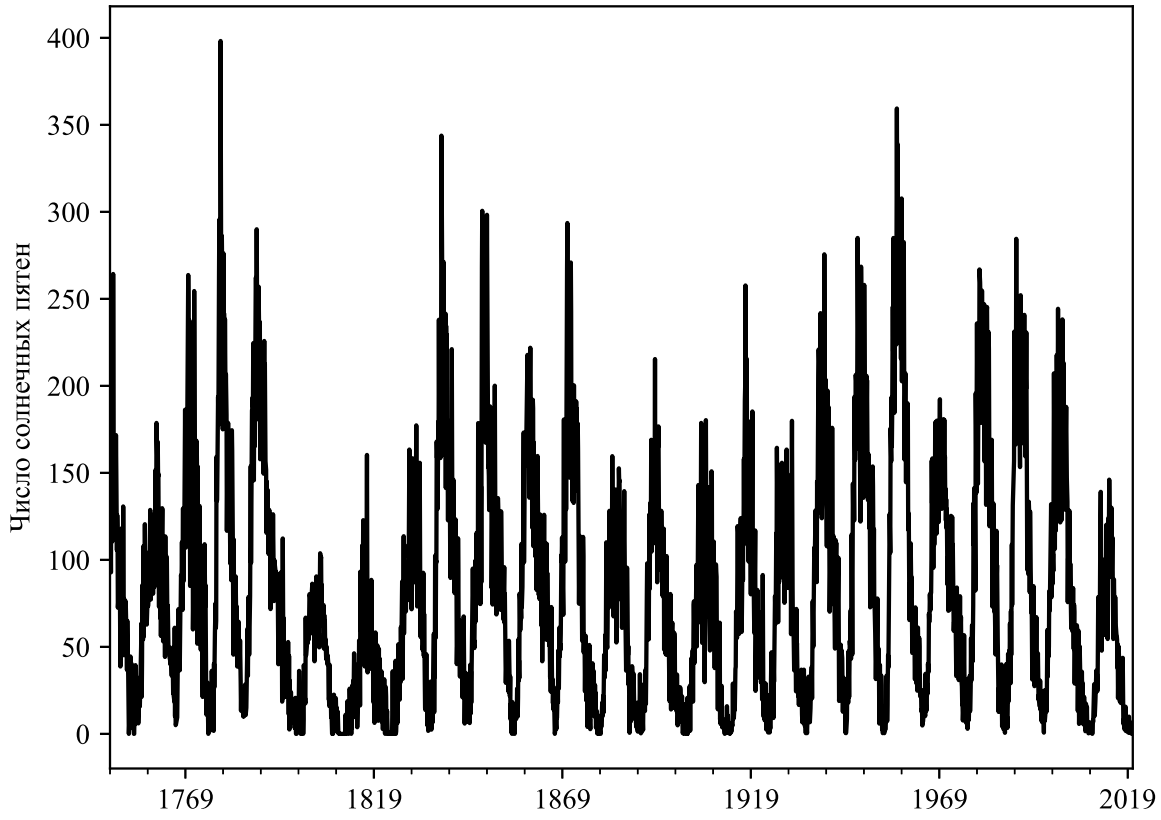


Рисунок 4.1 — Среднемесячные несглаженные числа солнечных пятен

Важным для настоящего исследования является то, что Служба космической погоды (The Space Weather Services, SWS) Австралийского метеорологического бюро на сайте `http://listserver.ips.gov.au/pipermail/ips-ssn-predictions/` каждый месяц публикует свой прогноз для ряда среднемесячных чисел солнечных пятен. Более того, доступны для загрузки прогнозы, сделанные в прошлые месяцы. Это предоставляет возможность сравнить точность прогнозов, вычисляемых с помощью предлагаемых методов, с точностью прогнозов бюро. В июле 2015 года в SIDC была произведена рекалибровка данных, и SWS перешла на использование новой версии в феврале 2016 года. Следуя схеме вычислений, изложенной в параграфе 4.1, вычислим прогнозные значения на 4 шага вперёд начиная с этого месяца, т.е. s положим равным 3205 в 4.2. Будем использовать 7 архиваторов, перечисленных в параграфе 4.1, а также автомат. Применим адаптивный метод из параграфа 3.2, но для сравнения точности построим прогноз и с помощью простого объединения всех 8 методов по формуле 1.7 с равными весами, которое далее для краткости будем называть как *комбинирован-*

ный метод. Отметим, что SIDC публикует как несглаженные, так и сглаженные данные. Прогнозные значения вычислим по несглаженным данным, и точность построенных прогнозов и прогнозов SWS сравним также с несглаженными данными.

При использовании адаптивного метода необходимо выбрать размер префикса ряда, который будет сжат всеми архиваторами для поиска наиболее эффективного из них. В целях эксперимента, попробуем провести вычисления с 10-ю различными вариантами выбора размера этого префикса — будем сжимать от 10% до 100% значений ряда с шагом 10%. Это позволит определить, какого процента значений ряда достаточно для того, чтобы выбранный архиватор обеспечивал точность, равную точности комбинированного метода. Максимальное количество интервалов при квантовании, которое будем рассматривать, сначала выберем равным 8. Также возьмём от ряда первую разность. При таком подходе, для любой доли значений ряда в качестве лучшего метода выбирается ppmd. Значения средней абсолютной ошибки для ppmd, комбинированного метода, а также прогнозов SWS приведены в таблице 4.5.

Таблица 4.5 — Средние абсолютные ошибки при прогнозировании временного ряда солнечных пятен с использованием до 8 интервалов при квантовании

Метод	Номер шага			
	1	2	3	4
ppmd	8.0	11.3	12.7	15.9
Комбинированный метод	8.0	11.3	12.7	15.9
SWS	8.3	9.1	9.7	9.8

Увеличим максимальное количество интервалов при квантовании до 16. В этом случае при использовании от 10% до 40% значений ряда в качестве лучшего метода выбирается ppmd, для 50%–100% значений лучшим методом становится zstd. Средние абсолютные ошибки для ppmd, zstd, комбинированного метода, а также прогнозов SWS приведены в таблице 4.6. Из этой таблицы видно, что увеличение количества интервалов при квантовании привело к повышению точности прогнозов, а также что ppmd оказался немного точнее, чем zstd. Уменьшим s с 3205 до 2800 т.е. построим больше прогнозов для уже известных значений. В этом случае для шагов 1–3 zstd оказался несколько точнее, чем ppmd (средние абсолютные ошибки приведены в таблице 4.7). Для шага 4, ppmd оказался более точным. Таким образом, при использовании адаптивного метода, в зависимости

от выбранной доли значений ряда, прогноз должен быть вычислен либо с помощью zstd, либо с помощью ppmd. При этом заметим, что комбинированный метод показывает точность, аналогичную zstd, поэтому использование адаптивного метода не приводит к потере точности. При сравнении MAE прогнозов, полученных с помощью комбинированного метода, с MAE прогнозов SWS видно, что при прогнозировании на 1 шаг вперёд комбинированный метод оказался несколько более точным, в остальных случаях прогноз SWS точнее.

Сравним время работы комбинированного и адаптивного методов. Программа была запущена 5 раз с использованием всех 8-ми алгоритмов, и 5 раз с использованием адаптивного алгоритма, при этом для выбора лучшего метода сжималось 50% значений ряда. Среднее время, требуемое для вычисления одного прогноза на 4 шага вперёд (использовалось 1 ядро процессора), оказалось равным 9945.46 с. в первом случае и 564.63 с. во втором. Таким образом, использование адаптивного метода в данном случае позволяет уменьшить время вычислений более чем в 17 раз без потерь в точности прогнозов.

Таблица 4.6 — Средние абсолютные ошибки, полученные при прогнозировании временного ряда солнечных пятен с использованием до 16 интервалов при квантовании

Метод	Номер шага			
	1	2	3	4
ppmd	7.2	9.2	10.1	10.2
zstd	8.1	10.3	11.8	13.3
ppmd+zstd	8.1	10.3	11.8	13.3
Комбинированный метод	8.1	10.3	11.8	13.3
SWS	8.3	9.1	9.7	9.8

Таблица 4.7 — Средние абсолютные ошибки, полученные при прогнозировании временного ряда солнечных пятен с использованием до 16 интервалов при квантовании (количество прогнозных значений увеличено на 405 по сравнению с таблицей 4.6)

Метод	Номер шага			
	1	2	3	4
ppmd	16.5	19.1	20.4	21.6
zstd	16.4	18.7	20.0	22.1

Далее рассмотрим прогнозирование временного ряда планетарного K-индекса (Planetary K-index, Kp-index). K-индекс получил своё название от немецкого слова «Kennziffer» (характеристическое число) и служит характеристикой возмущений горизонтальной компоненты магнитного поля Земли [89]. Его возможными значениями являются числа от 0 до 9, причём величины выше либо равные 5 соответствуют геомагнитным штормам. Значение индекса получается из наибольших отклонений горизонтальных компонент, зафиксированных на магнитометре за 3 часа [90]. K-индекс является локальным, поскольку вычисляется по данным одной обсерватории. Характеристикой геомагнитной активности всей Земли может служить планетарный K-индекс (обозначается как Kp). Его получают путём взвешенного усреднения значений K-индекса от нескольких обсерваторий. Значения Kp с интервалом в 3 часа могут быть найдены на сайте Центра предсказания космической погоды (Space Weather Prediction Center, SWPC) Национального управления океанических и атмосферных исследований (National Oceanic and Atmospheric Administration, NOAA) США `https://www.swpc.noaa.gov/products/planetary-k-index`. Поскольку геомагнитная активность влияет на деятельность организаций, работающих в сферах электроэнергетики и космонавтики, на наблюдателей полярного сияния, а также на пользователей радиосвязи, при которой радиосигналы отражаются или проходят через ионосферу Земли, SWPC рассылает оповещения о геомагнитных штормах и использует значения Kp в качестве индикатора о необходимости рассылки очередного оповещения [90]. График данного временного ряда приведён на рисунке 4.2.

Построим прогнозы для исторических значений Kp-индекса с помощью архиваторов zlib, rp и метода Хольта-Уинтерса. Будем использовать данные за период с 16.11.2020 по 07.12.2020. Прогнозы будем строить на 4 шага вперёд, дифференцировать ряд не будем. s в 4.2 выберем равным $t/2$, где t — длина временного ряда за указанный период, равная 172-м, т.е. строить прогнозы будем для второй половины ряда. При этом для оценки коэффициентов в модели Хольта-Уинтерса, как и в параграфе 3.3, воспользуемся методом максимизации логарифмической функции правдоподобия, реализованном в пакете `statsmodels` для Python. В вызове конструктора класса `ExponentialSmoothing` значение параметра `seasonal` установим равным `'add'` (для включения в модель аддитивной сезонной составляющей), длину периода `seasonal_periods` зададим равной 8, значения остальных параметров оставим установленными по

умолчанию. Средние абсолютные ошибки прогнозов, построенных с помощью перечисленных методов и их комбинации с равными весами, приведены в таблице 4.8. Видно, что при прогнозировании данного ряда точнее оказалась модель Хольта-Уинтерса, но при этом комбинированный метод практически не уступает ей в точности.

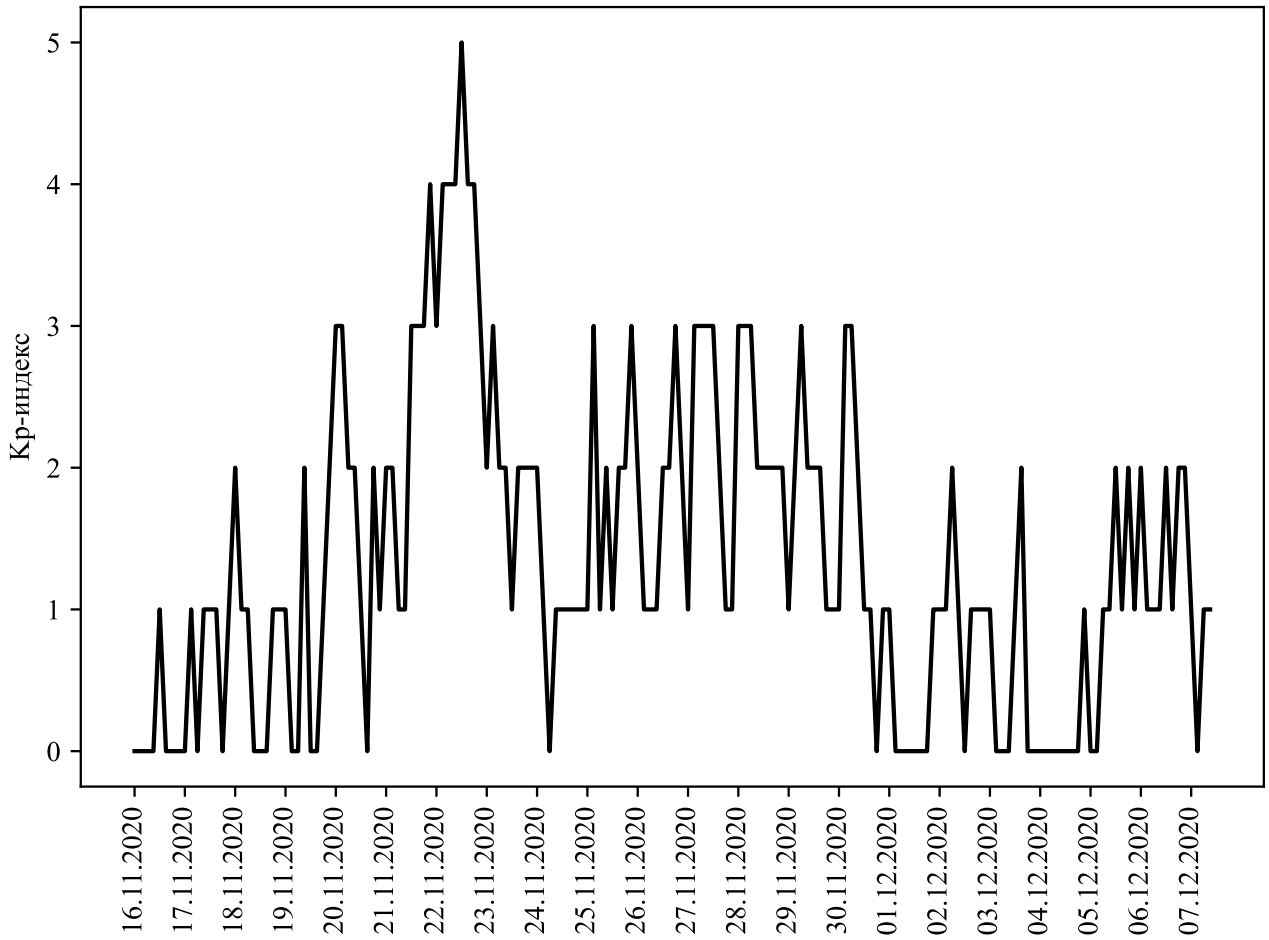


Рисунок 4.2 — График временного ряда Кр-индекса за период с 16.11.2020 по 07.12.2020

Таблица 4.8 — Средние абсолютные ошибки, полученные при прогнозировании второй половины временного ряда Кр-индекса

Метод	Номер шага			
	1	2	3	4
zlib	0.80	0.92	1.0	1.08
rp	1.06	1.34	1.08	1.46
holt-winters	0.58	0.71	0.76	0.72
zlib+rp+holt-winters	0.59	0.73	0.73	0.75

Наконец, рассмотрим прогнозирование временного ряда T-индекса. Данный временной ряд тесно связан с рядом солнечных пятен и используется для предсказания максимальной частоты распространения высокочастотных (ВЧ) волн. Максимальную частоту, при которой радиоволны отражаются от ионосферы, называют *максимальной применимой частотой* (МПЧ). Это значение определяет диапазон частот, которые могут быть использованы для радиосвязи. Известно, что МПЧ меняется с течением времени и зависит от фазы солнечного цикла, индикатором которой служит количество солнечных пятен. Поэтому временной ряд солнечных пятен может быть использован для предсказания МПЧ. Но проблема заключается в том, что есть и другие факторы, влияющие на МПЧ, такие как наличие или отсутствие геомагнитных штормов, а также уровень ультрафиолетовой солнечной радиации в крайней части спектра (solar extreme ultraviolet radiation), который не всегда меняется пропорционально числу солнечных пятен. С другой стороны, МПЧ может быть измерена прибором, который называется *ионозонд*. T-индекс отражает эквивалентное число солнечных пятен, соответствующее МПЧ, полученной в результате ионосферного зондирования. Иными словами, это число солнечных пятен, при котором наблюдалась бы измеренная МПЧ в случае отсутствия прочих факторов [91]. Временной ряд ежемесячных значений T-индекса можно найти на веб-сайте `http://listserver.ips.gov.au/mailman/listinfo/ips-tindex-predictions` метеорологического бюро Австралии. График этого ряда приведён на рисунке 4.3.

На указанном сайте метеорологического бюро (почти) каждому месяцу, начиная с ноября 2000 года, соответствует свой файл. В этом файле приведены исторические значения T-индекса, начиная с января 1938 года и заканчивая месяцем, предшествующим месяцу файла. Помимо этого, в каждом файле приводится прогноз на несколько месяцев вперёд, но количество месяцев варьируется от файла к файлу. В рамках данной работы для каждого месяца с апреля 2011 по апрель 2016 года (в этом временном интервале отсутствуют пропущенные значения) был построен прогноз на 18 значений вперед и затем проведено сравнение его точности с точностью прогноза метеорологического бюро. Вычисления были проведены следующим образом. На основании данных из файла за некоторый месяц (например, январь 2014 года) строился прогноз на 18 шагов вперёд. При этом использовалась та же методология, что и при прогнозировании данных M3C: исходный временной ряд разбивался на 6 рядов, т.е. фактически строились 6 прогнозов на 3 шага вперёд, использовалось сглаживание по формуле 4.1, взятие первой разности и

STL. Затем по файлу, на 18 месяцев более позднему, рассчитывалась абсолютная ошибка построенного прогноза и прогноза бюро, содержащегося в исходном файле, для каждого из шагов 1–18. После окончания обработки всех файлов вычислялась средняя ошибка для каждого шага. Результаты расчётов приведены в таблице 4.9. Видно, что при прогнозировании на 1 шаг вперёд предлагаемый метод оказался точнее метода метеорологического бюро, в остальных случаях точность прогнозов бюро оказалась выше. При анализе точности прогнозов отдельных методов сжатия для различных шагов видно, что лучший метод меняется в зависимости от номера шага. Так, например, для шага 1 наиболее точным оказался ppmd, для шага 4 — rp, для шага 5 — zlib. При этом точность комбинированного метода в среднем по шагам 1–18 оказалась лучше, чем точность каждого из методов по отдельности. Увеличение количества интервалов с 16 до 32 привело к небольшому уменьшению ошибки для отдельных шагов, но в среднем по всем шагам точность прогнозов ухудшилась.

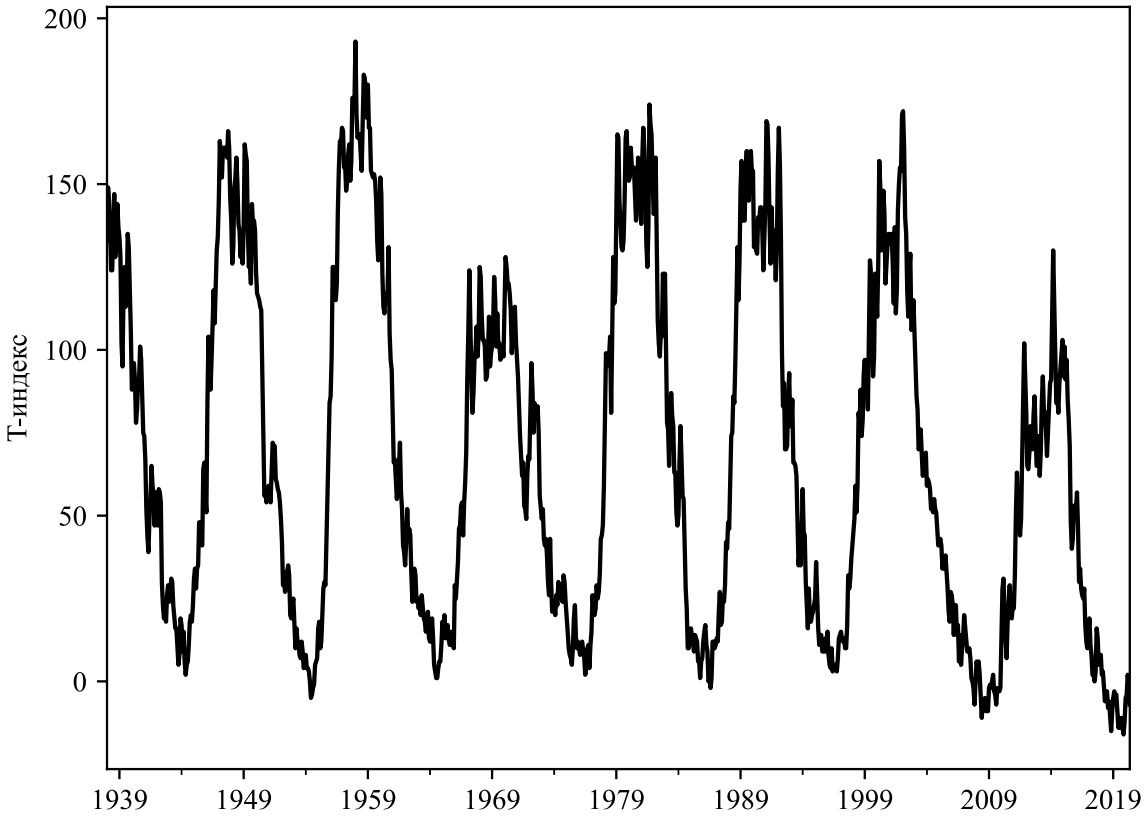


Рисунок 4.3 — График временного ряда Т-индекса

Таблица 4.9 — Результаты прогнозирования временного ряда T-индекса

Метод	Средняя абсолютная ошибка для номера шага h										Среднее			Кол-во ря-дов
	1	2	3	4	5	6	8	12	15	18	1–4	1–8	1–18	
Прогноз метеорологического бюро	13.0	14.2	14.8	15.7	16.5	18.0	21.2	23.5	25.6	27.0	14.43	16.69	21.09	61
zlib (16 интервалов)	12.3	16.0	18.3	20.4	19.1	21.5	24.9	24.5	28.8	26.5	16.75	19.70	23.68	61
ppmd (16 интервалов)	11.9	15.2	17.4	20.9	21.5	22.7	18.8	25.7	30.6	25.5	16.34	18.56	22.87	61
rp (16 интервалов)	12.5	18.1	18.6	18.3	24.9	24.5	22.6	33.4	38.3	41.9	16.86	21.10	28.2	61
zlib+ppmd+rp (16 интервалов)	11.9	15.2	17.4	20.8	21.5	22.0	18.8	24.9	29.4	24.2	16.31	18.46	22.57	61
zlib+ppmd+rp (32 интервала)	11.8	15.2	17.4	21.2	21.5	22.0	18.8	25.4	28.7	26.1	16.39	18.51	22.67	61

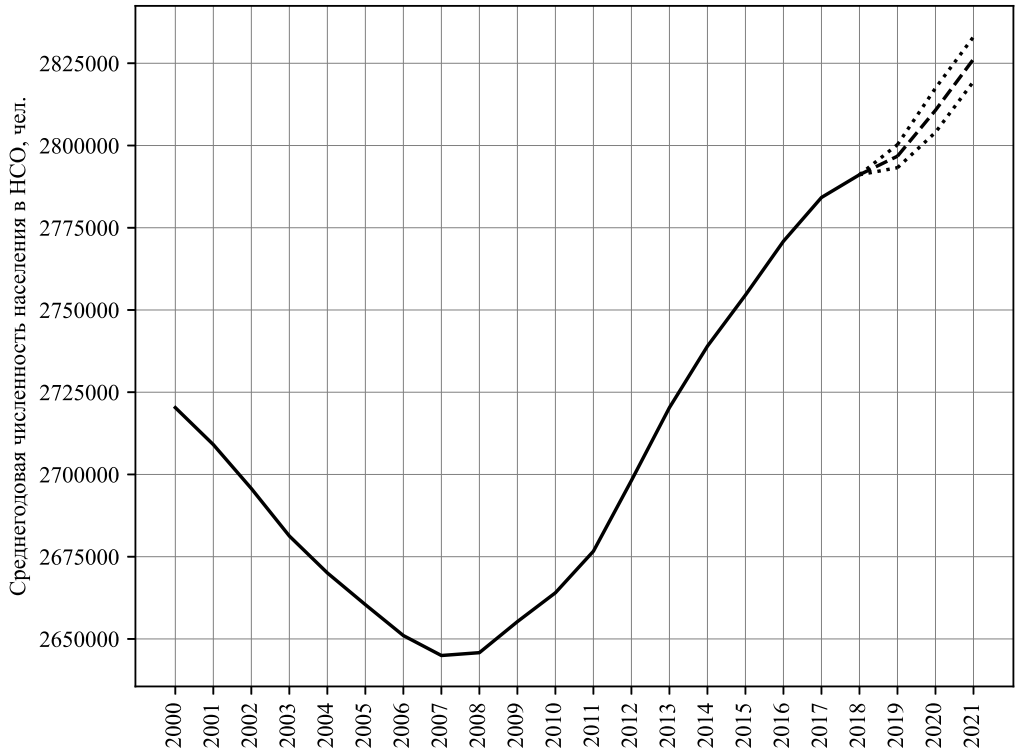
4.4 Прогнозирование социально-экономических показателей Новосибирской области

Теперь используем описанные ранее методы для прогнозирования некоторых демографических, социальных и экономических показателей Новосибирской области (НСО). Временные ряды, которые здесь рассматриваются, могут быть найдены на официальном сайте Федеральной службы государственной статистики по НСО (`https://novosibstat.gks.ru`, дата обращения 15.03.2020). Для построения прогнозов будем совместно использовать zlib, bzip2, ppmd, rp и автомат. При вычислениях во всех случаях s в 4.2 примем равным $t/2$, где t — длина ряда, т.е. построим прогнозы для второй половины значений рядов. Будем брать от данных первую разность, а в некоторых случаях и вторую (будем это явно указывать). Также применим сглаживание по формуле 4.1.

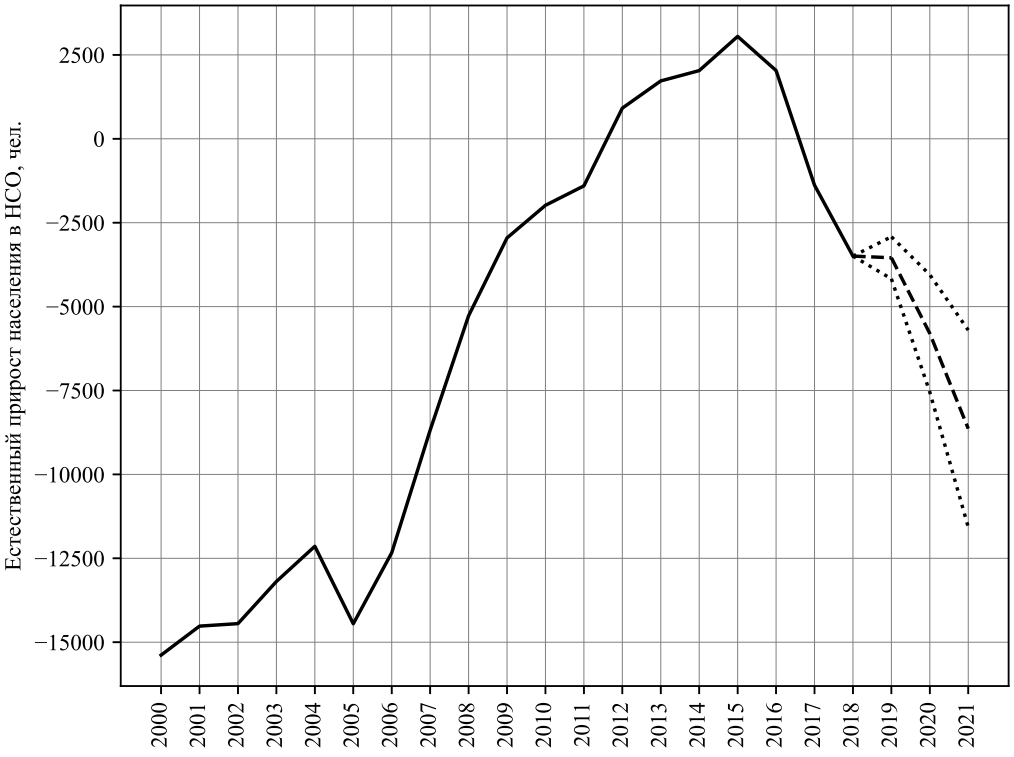
Перейдём к описанию результатов вычислений. Поскольку многие показатели НСО взаимосвязаны, начнём со случаев, в которых применение многомерного прогнозирования привело к повышению точности прогнозов. Рассмотрим ряды среднегодовой численности и естественного прироста населения в НСО. Их графики приведены на рисунке 4.4. Также на этом рисунке приведены прогнозы для каждого ряда на 3 года вперёд, полученные с помощью многомерного прогнозирования. Для их построения применялась процедура взятия второй разности, при квантовании использовались разбиения отрезков, в которые попадают значения рядов, на 2, 4 и 8 интервалов (вычисления проводились по формуле 1.6 с равными весами). Отметим, что на момент обращения к сайту Федеральной службы государственной статистики по НСО (15.03.2020), последние опубликованные значения для данных показателей были за 2018 год. В таблице 4.10 приведены прогнозные значения с доверительными интервалами, полученные при многомерном и одномерном прогнозировании, а в таблице 4.11 средние относительные ошибки для каждого шага (1–3), полученные при прогнозировании зафиксированных значений (начиная с 2011 года). Из этой таблицы видно, что средние относительные ошибки при многомерном прогнозировании оказались ниже, чем при одномерном.

Таблица 4.10 — Прогнозные значения и доверительные интервалы, полученные путём одномерного и многомерного прогнозирования рядов среднегодовой численности и естественного прироста населения в НСО

Тип прогнозирования	Ряд	Год		
		2019	2020	2021
Одномерное	Среднегод. числ. населения	2795721 [2791892;2799550]	2807615 [2800208;2815021]	2820005 [2811159;2828851]
	Естеств. прирост населения	-2969 [-3557;-2381]	-5296 [-7153;-3439]	-8572 [-11764;-5381]
Многомерное	Среднегод. числ. населения	2796757 [2793253;2800261]	2810730 [2804002;2817458]	2826252 [2819460;2833043]
	Естеств. прирост населения	-3545 [-4171;-2919]	-5801 [-7551;-4051]	-8637 [-11571;-5702]



а)



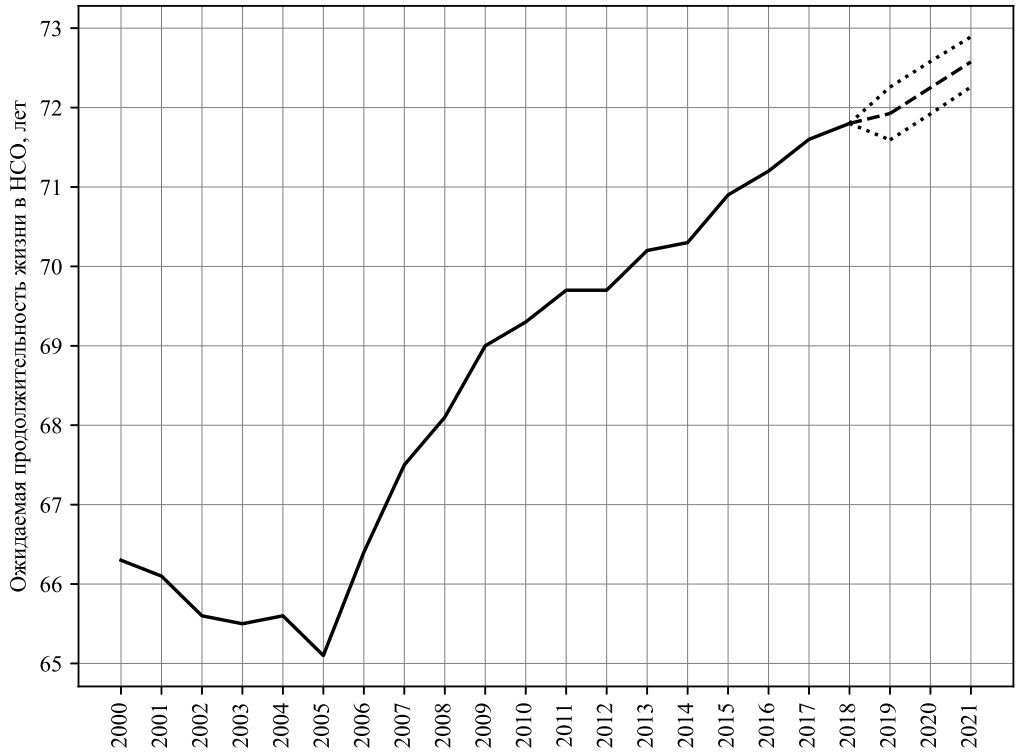
б)

Рисунок 4.4 — Временные ряды среднегодовой численности населения (а) и естественного прироста населения (б) в НСО. Непрерывной линией показаны зафиксированные значения, штриховой — прогнозные значения, пунктирными линиями ограничены доверительные интервалы (уровень доверия 95%)

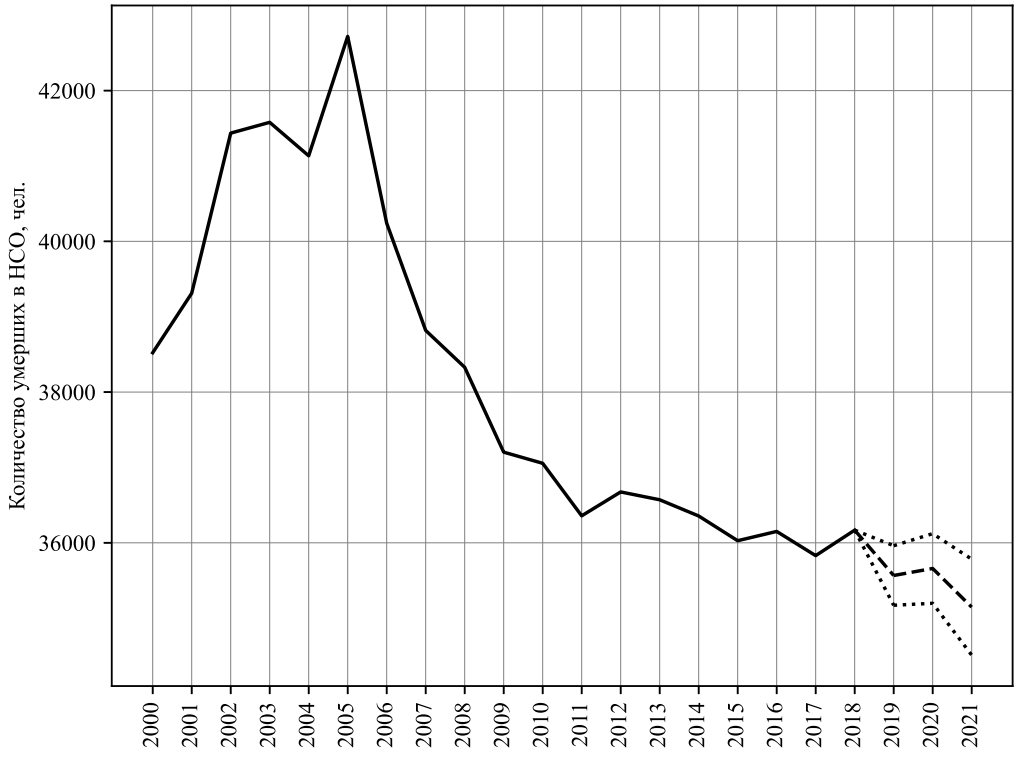
Таблица 4.11 — Средние относительные ошибки, полученные при одномерном и многомерном прогнозировании рядов среднегодовой численности и естественного прироста населения в НСО

Тип прогнозирования	Ряд	Шаг 1	Шаг 2	Шаг 3
Одномерное	Среднегодовая числ. населения	0.004	0.005	0.007
	Естественный прирост населения	0.143	0.356	0.670
Многомерное	Среднегодовая числ. населения	0.004	0.003	0.006
	Естественный прирост населения	0.138	0.278	0.545

В качестве другого примера рассмотрим прогнозирование рядов ожидаемой продолжительности жизни и количества умерших в НСО. Их графики, вместе с прогнозными значениями на 3 шага вперёд и доверительными интервалами, приведены на рисунке 4.5. В данном случае была взята первая разность, в остальном вычисления проводились аналогично предыдущему случаю. В таблице 4.12 вычисленные прогнозные значения и интервалы представлены в числовой форме. Средние относительные ошибки, полученные при многомерном и одномерном прогнозировании известных значений, приведены в таблице 4.13. Как видно из этой таблицы, средняя относительная ошибка для количества умерших в НСО оказалась ниже при многомерном прогнозировании, в то время как для ожидаемой продолжительности жизни однозначный вывод сделать нельзя.



а)



б)

Рисунок 4.5 — Временные ряды ожидаемой продолжительности жизни (а) и количества умерших (б) в НСО. Непрерывной линией показаны зафиксированные значения, штриховой — прогнозные значения, пунктирными линиями ограничены доверительные интервалы (уровень доверия 95%)

Таблица 4.12 — Прогнозные значения и доверительные интервалы, полученные путём одномерного и многомерного прогнозирования рядов ожидаемой продолжительности жизни и количества умерших в НСО

Тип прогнозирования	Ряд	Год		
		2019	2020	2021
Одномерное	Ож. прод. жизни	72.27 [72.05;72.49]	72.25 [72.05;72.45]	72.92 [72.67;73.17]
	Кол-во умерших	35846 [35478;36214]	35740 [35276;36205]	35438 [34897;35978]
Многомерное	Ож. прод. жизни	71.93 [71.59;72.26]	72.25 [71.92;72.58]	72.58 [72.26;72.89]
	Кол-во умерших	35566 [35171;35961]	35660 [35198;36121]	35147 [34510;35784]

Таблица 4.13 — Средние относительные ошибки, полученные при одномерном и многомерном прогнозировании рядов ожидаемой продолжительности жизни и количества умерших в НСО

Тип прогнозирования	Ряд	Шаг 1	Шаг 2	Шаг 3
Одномерное	Ож. прод. жизни	0.005	0.005	0.009
	Количество умерших	0.014	0.020	0.036
Многомерное	Ож. прод. жизни	0.006	0.006	0.006
	Количество умерших	0.013	0.015	0.017

Далее, приведём в таблице 4.14 прогнозы на 4 года вперёд для некоторых других показателей НСО, полученные с помощью одномерного прогнозирования (увеличение количества шагов по сравнению с предыдущими примерами связано с тем, что трудоёмкость вычислений при одномерном прогнозировании ниже, чем при многомерном). Отметим, что при прогнозировании рядов валового регионального продукта и количества браков бралась вторая разность, во всех остальных случаях использовалась первая разность. Как и ранее, совместно рассматривались разбиения множеств возможных значений рядов на 2, 4 и 8 интервалов при квантовании.

Таблица 4.14 — Прогнозные значения на 4 шага вперёд и доверительные интервалы для некоторых других показателей НСО

Показатель	Шаг 1	Шаг 2	Шаг 3	Шаг 4
Прожиточный минимум, руб.	2020	2021	2022	2023
	11405 [10777;12033]	11752 [11106;12398]	12098 [11394;12802]	12445 [11897;12992]
Валовой рег. продукт, млн. руб.	2019	2020	2021	2022
	1274481 [1243953;1305009]	1380703 [1323329;1438077]	1499047 [1451156;1546939]	1627328 [1558722;1695934]
Количество браков	2019	2020	2021	2022
	17190 [16592;17787]	14585 [13482;15689]	11803 [9786;13819]	8590 [5988;11191]
Количество разводов	2019	2020	2021	2022
	12699 [11721;13677]	11904 [10954;12855]	11799 [10631;12967]	11001 [9355;12646]
Средний возраст матери	2019	2020	2021	2022
	28.94 [28.85;29.03]	29.1 [28.96;29.24]	29.27 [29.1;29.43]	29.44 [29.29;29.58]
Дебитор. задол. организац., млн. руб.	2019	2020	2021	2022
	349623 [345045;354201]	372745 [359634;385856]	397037 [394762;399312]	418228 [407720;428737]
Кредитор. задол. организац, млн. руб.	2019	2020	2021	2022
	458611 [446726;470496]	480336 [473660;487011]	492491 [486058;498924]	514131 [472756;555506]
Средн. цены на рынке жилья, руб. за 1 кв.м.	2020	2021	2022	2023
	56743 [53363;60123]	60273 [55523;65022]	63766 [58357;691741]	66952 [61477;72428]

Поясним, что приведённые прогнозы были опубликованы в [92], и на момент их вычисления (15.03.2020) последние зафиксированные значения были доступны за 2018–2019 годы. По этой причине, прогнозы не могут учитывать влияние пандемии COVID-19 — на исторических данных в период с 2000 по 2019 год подобные явления в НСО не происходили. На момент написания этого текста (19.02.2022) некоторые новые зафиксированные значения для всех прогнозируемых показателей стали известны, поэтому приведём их вместе с прогнозами в таблице 4.15. Анализируя эту таблицу, можно сказать, что наиболее точными получились прогнозы для среднегодовой численности населения, величины прожиточного минимума, количеств браков и разводов. Для естественного прироста

населения и ожидаемой продолжительности жизни заметны большие ошибки прогнозов на 2020 год.

Таблица 4.15 — Прогнозные и зафиксированные значения для показателей НСО. Для каждого показателя в первой строке приведены годы, для которых строился прогноз, во второй — зафиксированные и (в скобках) прогнозные значения показателя. Прочерком проставлены неизвестные значения

Показатель	Шаг 1	Шаг 2	Шаг 3	Шаг 4
Среднегод. числ. населения	2019	2020	2021	—
	2795777 (2796757)	2792003 (2810730)	— (2826252)	— (—)
Естеств. прирост населения	2019	2020	2021	—
	−5582 (−3545)	−13974 (−5801)	— (−8637)	— (—)
Ож. прод. жизни	2019	2020	2021	—
	72.3 (71.93)	70.3 (72.25)	— (72.58)	— (—)
Кол-во умерших	2019	2020	2021	—
	35605 (35566)	42833 (35660)	— (35147)	— (—)
Прожиточный минимум, руб.	2020	2021	2022	2023
	11845 (11405)	12284 (11752)	12775 (12098)	— (12445)
Валовой рег. продукт, млн. руб.	2019	2020	2021	2022
	1409192 (1274481)	— (1380703)	— (1499047)	— (1627328)
Количество браков	2019	2020	2021	2022
	19405 (17190)	15979 (14585)	— (11803)	— (8590)
Количество разводов	2019	2020	2021	2022
	13281 (12699)	12235 (11904)	— (11799)	— (11001)
Средний возраст матери	2019	2020	2021	2022
	28.7 (28.94)	28.7 (29.1)	— (29.27)	— (29.44)
Дебитор. задол. организац., млн. руб.	2019	2020	2021	2022
	321259.2 (349623)	357335.9 (372745)	— (397037)	— (418228)
Кредитор. задол. организац, млн. руб.	2019	2020	2021	2022
	413088.3 (458611)	455794.1 (480336)	— (492491)	— (514131)
Средн. цены на рынке жилья, руб. за 1 кв.м.	2020	2021	2022	2023
	63610.77 (56743)	73620.15 (60273)	— (63766)	— (66952)

Глава 5. Описание программного комплекса

В данной главе приведено описание программной реализации разработанных методов, исходный код которой доступен на сайте github по ссылке `https://github.com/kchirikhin/itp`.

5.1 Требования к программному комплексу

Прежде чем переходить непосредственно к описанию программной реализации, сформулируем основные требования, которые учитывались при её разработке.

- *Расширяемость.* Поскольку в предлагаемых методах используется набор алгоритмов сжатия данных необходимо, чтобы в программную реализацию было легко добавлять новые архиваторы. Также нужно предусмотреть возможность добавления новых методов прогнозирования с модификацией, описанной в параграфе 3.3;
- *Кроссплатформенность.* Желательно, чтобы программа могла выполняться под управлением любой широко используемой в настоящее время операционной системы (Linux, Windows, Max OS X);
- *Производительность.* Время работы программы преимущественно зависит от скорости обработки последовательностей используемыми методами прогнозирования. Тем не менее, программная реализация не должна вносить существенных накладных расходов при вычислениях, количество запусков процедур сжатия должно стремиться к минимально необходимому. Полезно предусмотреть возможность параллельного выполнения вычислений;
- *Интегрируемость с существующими системами анализа данных.* В настоящее время существует несколько широко используемых сред для анализа данных. Одними из самых популярных являются Python (`https://www.python.org/`) и R (`https://www.r-project.org/`). В этих средах реализованы различные методы визуализации, предварительной обработки данных, чтение данных из разных форматов и т.д.

Желательно, чтобы программная реализация могла использоваться в одной или нескольких из подобных систем.

Чтобы удовлетворить перечисленным требованиям, основная часть программы была выполнена в виде библиотеки, при разработке которой использовался язык программирования C++. Эта библиотека была названа `itp_core` (itp — сокращение от information-theoretic predictor). Данную библиотеку можно использовать самостоятельно при разработке программ на C++, а также возможна реализация доступа к ней из другой среды программирования или анализа данных. В рамках настоящей работы был разработан пакет `itp` для языка программирования Python, основанный на `itp_core`.

5.2 Описание библиотеки itp_core

В данной библиотеке реализованы: вычисление прогнозных значений для временного ряда по формулам 1.4–1.7, квантование (параграф 1.3), адаптивный метод прогнозирования из параграфа 3.2, процедура декомпозиции временного ряда на k более коротких рядов для их прогнозирования по отдельности (параграф 4.1), взятие n-ых разностей, а также базовые классы для работы с произвольными методами прогнозирования с модификацией, описанной в параграфе 3.3. Структура библиотеки приведена на рисунке 5.1.

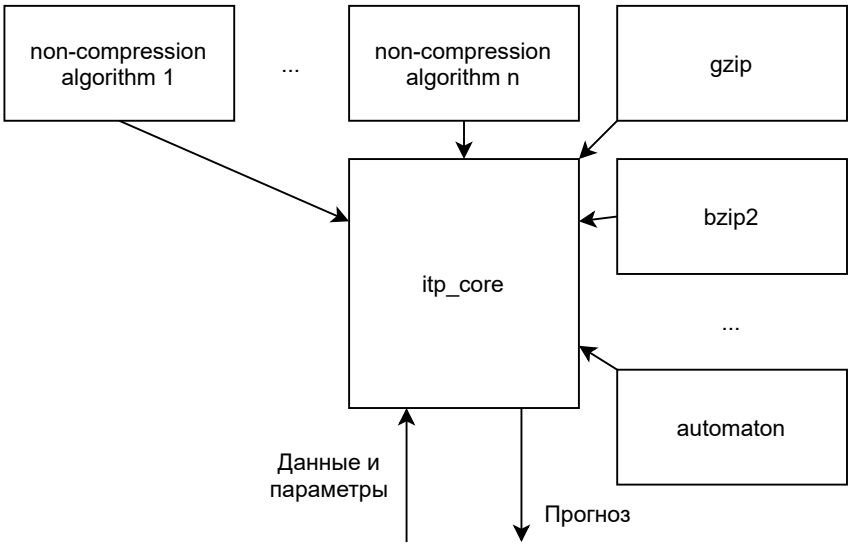


Рисунок 5.1 — Структура библиотеки `itp_core`

В данном параграфе не будем описывать подробности реализации itp_core, вместо этого опишем назначения и интерфейсы классов, о которых необходимо знать пользователю данной библиотеки для непосредственной работы с ней, а также для её расширения.

Реализации методов сжатия данных, такие как zlib и bzip2, подключаются к itp_core как статические библиотеки. Для их интеграции предусмотрен интерфейс ICompressor — для каждого метода сжатия, интегрируемого в itp, нужно создать класс, реализующий данный интерфейс, в котором будет сосредоточен весь необходимый для взаимодействия с этим методом код. Объявление ICompressor приведено в листинге 5.1.

Листинг 5.1: Объявление класса ICompressor

```
class ICompressor
{
public:
    using SizeInBits = size_t;
    using Continuations = std::vector<Continuation<Symbol>>;

    virtual ~ICompressor() = default;

    virtual SizeInBits Compress(const unsigned char* data,
        size_t size,
        std::vector<unsigned char>* output_buffer) = 0;

    virtual std::vector<SizeInBits> CompressContinuations(
        const std::vector<Symbol>& historical_values,
        const Continuations& possible_endings) = 0;

    virtual void SetTsParams(Symbol alphabet_min_symbol,
        Symbol alphabet_max_symbol) = 0;
};
```

Метод Compress принимает на вход массив с данными, которые нужно сжать (т.е. временной ряд), количество элементов в массиве, а также буфер, в который следует поместить сжатые данные (его содержимое далее в библиотеке нигде не используется, и вызывающий код многократно передаёт один и тот же

буфер для сокращения количества вызовов оператора `new` для выделения динамической памяти). В качестве результата этот метод возвращает размер сжатого представления данных в битах. Метод `CompressContinuations` поочерёдно дописывает каждую из последовательностей из аргумента `possible_endings` в конец `historical_values`, сжимает сформированные «расширенные» временные ряды и возвращает полученные в результате длины кодовых слов в битах. Таким образом, один вызов этого метода заменяет собой несколько вызовов `Compress`. Такой подход эффективнее с точки зрения производительности для методов прогнозирования, не основанных на сжатии, т.к. позволяет один раз построить одношаговые прогнозы для `historical_values` вместо `possible_endings.size()` раз. Метод `SetTsParams` является вспомогательным и используется для того, чтобы передать архиватору размер алфавита. Для классических методов сжатия эта информация не используется, но она нужна автомату и методам прогнозирования, которые были преобразованы в методы сжатия.

Для интеграции методов прогнозирования, не являющихся методами сжатия, предусмотрен класс `NonCompressionAlgorithmAdaptor`, реализующий интерфейс `ICompressor` и логику из параграфа 3.3. Конструктор `NonCompressionAlgorithmAdaptor` в качестве единственного аргумента принимает объект с интерфейсом `INonCompressionAlgorithm`, в котором сосредоточено взаимодействие с конкретным методом прогнозирования. Объявление интерфейса `INonCompressionAlgorithm` приведено в листинге 5.2.

В методе `GiveNextPrediction` по переданной последовательности символов интегрируемым методом прогнозирования строится прогноз на один шаг вперёд с указанием, «уверен» ли этот метод в своём прогнозе. Метод может быть не уверен в прогнозе если, например, длина данных недостаточна для вычисления прогноза и было возвращено значение по умолчанию. И аналогично `ICompressor`, предусмотрен метод `SetTsParams`, в котором задаётся размерность алфавита для передаваемых впоследствии в `GiveNextPrediction` временных рядов.

Все экземпляры доступных методов сжатия/прогнозирования вместе с указателем на область динамической памяти, в которую будут помещаться результаты сжатия, содержатся в объекте класса `CompressorsPool`. Объявление этого класса приведено в листинге 5.3.

Листинг 5.2: Объявление класса INonCompressionAlgorithm

```
enum class ConfidenceLevel : unsigned char
{
    kConfident = 0,
    kNotConfident = 1
};

class INonCompressionAlgorithm
{
public:
    using Guess = std::pair<Symbol, ConfidenceLevel>;

    virtual ~INonCompressionAlgorithm() = default;
    virtual Guess GiveNextPrediction(const unsigned char* data,
        size_t size) = 0;
    virtual void SetTsParams(Symbol alphabet_min_symbol,
        Symbol alphabet_max_symbol) = 0;
};
```

Метод RegisterCompressor регистрирует экземпляр алгоритма сжатия (или прогнозирования) для его последующего использования. Оставшиеся три метода Compress, CompressContinuations и SetAlphabetDescription повторяют сигнатуру и назначение одноимённых методов класса ICompressor, и их реализация является тривиальной — управление передаётся соответствующему методу архиватора, найденному по указанному имени. Класс CompressorsPool в первую очередь необходим для реализации возможности добавления новых методов прогнозирования на этапе выполнения кода. Например, в данной работе поддержана возможность создания производных классов от NonCompressionAlgorithm на языке программирования Python. При этом объекты таких классов нужно регистрировать в itp_core, и затем их можно использовать аналогично методам сжатия данных, присутствовавшим в библиотеке на этапе компиляции.

Листинг 5.3: Объявление класса `CompressorsPool`

```
class CompressorsPool
{
public:
    void RegisterCompressor(std::string name,
        std::unique_ptr<ICompressor> compressor);

    ICompressor::SizeInBits Compress(
        const std::string& compressor_name,
        const unsigned char* data, size_t size);

    std::vector<ICompressor::SizeInBits> CompressContinuations(
        const std::string& compressor_name,
        const std::vector<Symbol>& historical_values,
        const ICompressor::Continuations& possible_continuations);

    void SetAlphabetDescription(
        AlphabetDescription alphabet_description);

private:
    std::unordered_map<std::string, std::unique_ptr<ICompressor>>
        compressor_instances_;
    std::vector<unsigned char> output_buffer_;
};
```

Для пользователей `itp_core` работа с ней должна происходить через взаимодействие с объектом класса `InformationTheoreticPredictor`, объявление которого приведено в листинге 5.4. Метод `ForecastReal` предназначен для прогнозирования вещественных временных рядов. Первым параметром в него передаётся прогнозируемый ряд, вторым — комбинации методов сжатия, с помощью которых нужно построить прогнозы. Например, если передать в качестве `concatenated_compressor_groups` массив строк `{"gzip_bzip2_lcacomp", "automaton"}`, то это будет означать, что нужно построить прогноз два раза: один раз с помощью комбинации методов `gzip`, `bzip2` и `lcacomp`, а второй раз — с помощью `automaton`. Если один и тот же метод входит в несколько групп, все последовательности будут сжаты им только один раз, что сокращает объём вычислений. Третьим параметром (`horizon`) передаётся количество шагов, на которое нужно построить прогноз. Четвёртый

параметр difference — порядок разности. Параметр quanta_count задаёт количество интервалов при квантовании, sparse определяет, на какое количество временных рядов должен быть разбит исходный ряд (параграф 4.1). Методы ForecastMultialphabet, ForecastMultialphabetVec, ForecastDiscrete и ForecastDiscreteVec похожи на ForecastReal, поэтому назначение одноимённых параметров у них совпадает. ForecastMultialphabet реализует вычисления по формуле 1.6, в то время как в ForecastReal используется только одно разбиение при квантовании. ForecastMultialphabetVec способен прогнозировать многомерные данные, элементами его первого параметра history являются не скалярные, а векторные значения. ForecastDiscrete и ForecastDiscreteVec предназначены для прогнозирования целочисленных рядов, процедура квантования в данных методах не используется, а параметры, задающие количество интервалов при квантовании, отсутствуют. Возвращаемыми значениями всех перечисленных методов являются отображения названий групп методов сжатия ("gzip_bzip2_lcacomp" и "automaton" из примера выше) на вектора прогнозных значений. Метод RegisterNonCompressionAlgorithm необходим для регистрации новых алгоритмов прогнозирования. Заметим, что объект класса InformationTheoreticPredictor содержит указатель на объект типа CompressorsPool, в который и добавляются эти алгоритмы.

Ещё одной функцией, относящейся к интерфейсу itp_core, является SelectBestCompressors, её объявление приведено в листинге 5.5. Данная функция предназначена для поддержки адаптивного метода прогнозирования из параграфа 3.2. Как видно из листинга, большая часть параметров этой функции совпадает с параметрами методов для прогнозирования класса InformationTheoreticPredictor. Функция является шаблонной и может работать с любым типом временных рядов, с которым может работать InformationTheoreticPredictor. Новыми параметрами являются part_to_consider, который задаёт долю значений временного ряда, используемую для поиска лучших методов прогнозирования, и target_number — количество искомых методов. Названия найденных методов возвращаются в виде вектора.

Листинг 5.4: Объявление класса `InformationTheoreticPredictor`

```
class InformationTheoreticPredictor
{
public:
    map<string, vector<itp::Double>> ForecastReal(
        const vector<itp::Double>& time_series,
        const ConcatenatedCompressorNamesVec&
            concatenated_compressor_groups, size_t horizon,
        size_t difference, size_t quanta_count, int sparse);

    map<string, vector<itp::Double>> ForecastMultialphabet(
        const vector<double>& history,
        const itp::ConcatenatedCompressorNamesVec&
            concatenated_compressor_groups, size_t horizon,
        size_t difference, size_t max_quanta_count, int sparse);

    map<string, vector<vector<double>>>
    ForecastMultialphabetVec(
        const vector<std::vector<double>>& history,
        const itp::ConcatenatedCompressorNamesVec&
            concatenated_compressor_groups, size_t horizon,
        size_t difference, size_t max_quanta_count, int sparse);

    map<string, vector<itp::Double>>
    ForecastDiscrete(
        const vector<itp::Symbol>& history,
        const ConcatenatedCompressorNamesVec&
            concatenated_compressor_groups, size_t horizon,
        size_t difference, int sparse);

    map<string, vector<itp::VectorDouble>>
    ForecastDiscreteVec(
        const vector<itp::VectorSymbol>& history,
        const ConcatenatedCompressorNamesVec&
            concatenated_compressor_groups, size_t horizon,
        size_t difference, int sparse);

    void RegisterNonCompressionAlgorithm(const string& name,
        itp::INonCompressionAlgorithm* non_compression_algorithm);

private:
    shared_ptr<CompressorsPool> compressors_;
};
```

Листинг 5.5: Объявление функции `SelectBestCompressors`

```
template<typename T>
itp::CompressorNames SelectBestCompressors(
    const std::vector<T>& history,
    const itp::CompressorNames& compressor_names,
    const size_t difference,
    const std::vector<size_t>& quanta_count,
    const Share part_to_consider,
    const size_t target_number);
```

5.3 Описание пакета itp

Для более удобной работы с библиотекой `itp_core` был реализован пакет для Python, названный `itp`. Использование этого пакета позволяет работать со стандартными средствами обработки и визуализации данных Python. Пакет обладает следующими возможностями:

- Предоставляет доступ ко всей функциональности `itp_core` из Python;
- Позволяет автоматически строить прогнозы для исторических значений рядов с целью оценки точности моделей и построения доверительных интервалов;
- Поддерживает возможность добавления в `itp_core` новых методов прогнозирования, изначально не основанных на сжатии, из Python;
- Предоставляет возможность параллельного построения нескольких прогнозов с использованием библиотеки MPI (Message Passing Interface);
- Предоставляет средства для визуализации исторических данных, прогнозов и доверительных интервалов, в том числе при работе в параллельном режиме;
- Реализует модификацию из параграфа 3.3 для метода Хольта-Уинтерса.

Далее опишем основные компоненты пакета. Как и в предыдущем параграфе, не будем рассматривать их реализацию, а ограничимся описанием назначений и интерфейсов основных классов.

Для организации доступа к `itp_core` из Python была использована библиотека pybind11 (`https://github.com/pybind/pybind11`). С её помощью

в Python были экспортированы классы `InformationTheoreticPredictor`, `NonCompressionAlgorithm` и `INonCompressionAlgorithm`, а также функции `select_best_compressors_multialphabet` и `select_best_compressors_discrete`, являющиеся специализациями шаблонной функции `SelectBestCompressors`.

Для взаимодействия с классом `InformationTheoreticPredictor` в пакете `itp` предусмотрен класс `ItpAccessor`. Он повторяет интерфейс `InformationTheoreticPredictor`, но объект данного класса, при регистрации через него новых алгоритмов прогнозирования, реализованных в Python, запоминает ссылки на их объекты у себя в памяти, чтобы сделать время жизни этих объектов не менее коротким, чем время жизни данного экземпляра `ItpAccessor`.

Для реализации процедуры построения прогнозов для уже известных значений некоторого ряда в параллельном режиме, а также просто для возможности параллельного построения нескольких прогнозов, в `itp` добавлен механизм задач. Всего в пакете присутствуют два типа задач прогнозирования: `BasicTask` и `TrainingTask`. Первый из них соответствует случаю, в котором требуется построить прогноз на h шагов вперёд для одного временного ряда. Второй тип задачи соответствует ситуации, в которой нужно построить набор прогнозов для уже известных значений ряда. Обозначим прогнозируемый временной ряд как $x_1, x_2, \ldots, x_t$. `TrainingTask` строит прогноз на h шагов вперёд по всем $x_1, x_2, \ldots, x_i$, $i \in \{t_0, t_0 + 1, t - h\}$ для заданного $t_0 \in \{1, 2, \ldots, t - h\}$.

Опишем реализацию механизма задач. Для этого сначала рассмотрим несколько вспомогательных классов, для некоторых из которых пользователь может предоставить собственные реализации. Первым таким классом является абстрактный класс `ITimeSeriesTransformator`. В производных от него классах могут быть реализованы различные методы преобразований временных рядов, такие как сглаживание. Интерфейс данного класса приведён в листинге 5.6. Если класс, производный от `ITimeSeriesTransformator`, реализует обратимое преобразование данных (например, взятие разности с некоторым лагом для удаления сезонной составляющей), то ему необходимо реализовать оба метода `transform` и `inverse_transform`. Если же класс реализует только прямое преобразование (например, сглаживание), то в нём может быть переопределён только метод `transform`. Тривиальной реализацией интерфейса `ITimeSeriesTransformator` в пакете `itp` явля-

ется класс EmptyTimeSeriesTransformator. Также присутствует класс BasicSmoothingTimeSeriesTransformator, реализующий сглаживание по формуле 4.1.

Листинг 5.6: Объявление абстрактного класса ITimeSeriesTransformator

```
class ITimeSeriesTransformator(ABC):
    @abstractmethod
    def transform(self, time_series: TimeSeries) -> TimeSeries:
        pass

    @abstractmethod
    def inverse_transform(self, result: Forecast) -> Forecast:
        pass
```

Ещё одним вспомогательным классом является IElementaryTask, его объявление приведено в листинге 5.7. Он является абстракцией элементарной задачи, которая непосредственно может быть выполнена с помощью одного из методов класса InformationTheoreticPredictor библиотеки itp_core. Задача типа BasicTask соответствует одной элементарной задаче, в то время как задача типа TrainingTask декомпозируется на $t - h - t_0 + 1$ элементарных задач. Как видно из листинга, класс содержит всего один метод, вызов которого должен непосредственно приводить к запуску вычислений.

Листинг 5.7: Объявление абстрактного класса IElementaryTask

```
class IElementaryTask:
    @abstractmethod
    def run(self) -> None:
        pass
```

Число классов, производных от IElementaryTask, соответствует числу методов прогнозирования класса InformationTheoreticPredictor. Реализации этих классов достаточно просты и однотипны. Для примера рассмотрим DiscreteUnivariateElemetaryTask, соответствующий задаче прогнозирования одномерного целочисленного временного ряда. Объявление этого класса приведено в листинге 5.8. Как видно из этого листинга, в конструктор передаются параметры, необходимые для вызова соответствующего метода класса

ItpAccessor, ссылка на экземпляр ItpAccessor, который будет выполнять процедуру прогнозирования (в этом экземпляре могут быть зарегистрированы методы прогнозирования, реализованные на Python, или сюда может быть передан mock-объект при запуске модульных тестов), а также ссылка на преобразователь данных transformator. В методе run сначала вызывается transformator.preprocess, затем запускается процедура прогнозирования, затем вызывается transformator.postprocess.

Листинг 5.8: Объявление класса DiscreteUnivariateElemetaryTask

```
class DiscreteUnivariateElemetaryTask(IElementaryTask):
    def __init__(self, time_series: TimeSeries,
                 compressors: List[ConcatenatedCompressorGroup],
                 horizon: int, difference: int, sparse: int,
                 itp_accessor: ItpAccessor = None,
                 transformator: ITimeSeriesTransformator = None):
        pass

    def run(self):
        pass
```

Для обработки результатов вычислений в itp используются классы, производные от ITaskResult. Например, для TrainingTask в таком классе вычисляются средние ошибки прогнозирования для каждого шага и доверительные интервалы для будущих значений. Интерфейс ITaskResult приведён в листинге 5.9. Как видно из этого листинга, результатом любой задачи является прогноз для будущих значений, который можно получить по названию группы алгоритмов, с помощью которых он строился (вспомним, что в InformationTheoreticPredictor при прогнозировании передаётся список групп алгоритмов, с помощью которых нужно построить прогноз), а также есть возможность получить прогнозируемый временной ряд.

Классы, производные от ITaskResult и соответствующие задачам BasicTask и TrainingTask, существенно отличаются между собой. Для представления результатов выполнения задачи вида BasicTask разработан класс BasicTaskResult. Он содержит новый метод set_results_of_computations, в который передаются прогнозируемый временной ряд и вычисленный прогноз. Как следствие, реализация

абстрактных методов из ITaskResult для него становится тривиальной. Для представления результатов выполнения TrainingTask разработан класс TrainingTaskResult. Он также содержит метод set_results_of_computations, но в него, помимо прогнозируемого временного ряда и вычисленного прогноза, передаются массивы тренировочных временных рядов, вычисленных для них прогнозов и соответствующие прогнозам наблюдаемые значения. На основе этой информации в TrainingTaskResult вычисляются средние абсолютные ошибки прогнозов на каждый шаг, средние относительные ошибки для каждого шага, а также доверительные интервалы для будущих значений. В случае необходимости проведения каких-либо иных вычислений, пользователь может определить свой класс, расширяющий TrainingTaskResult или заменяющий его.

Листинг 5.9: Объявление класса ITaskResult

```
class ITaskResult:
    @abstractmethod
    def forecast(self, compressors_group) -> TimeSeries:
        pass

    @abstractmethod
    def history(self) -> TimeSeries:
        pass
```

Перейдём теперь непосредственно к рассмотрению основных классов. Базовым классом для задач всех типов является ITask, его объявление приведено в листинге 5.10. Метод get_elementary_tasks возвращает список элементарных задач, на которые декомпозируется задача. Метод set_results_of_computations предназначен для передачи в объект задачи результатов вычислений. Гарантируется, что результат с номером i соответствует элементарной задаче с номером i. Метод history позволяет получить временной ряд, для которого была создана задача.

Далее опишем класс BasicTask, его объявление приведено в листинге 5.11. В конструктор данного класса передаётся объект, в который будет помещён результат выполнения задачи, тип для элементарной задачи, прогнозируемый временной ряд, а также прочие параметры для конструктора элементарной задачи. Такой интерфейс не очень удобен для пользователя —

вместо явного перечисления ожидаемых параметров в конструкторе использованы `args` и `kwargs`, при работе с ним необходимы знания о деталях реализации в виде элементарных задач. Поэтому в пакете предусмотрены дочерние классы `DiscreteUnivariateBasicTask`, `RealUnivariateBasicTask` и `RealMultivariateBasicTask`, в конструкторах которых списки допустимых параметров указаны явно и типы элементарных задач отсутствуют.

Листинг 5.10: Объявление абстрактного класса `ITask`

```
class ITask:
    @abstractmethod
    def get_elementary_tasks(self) -> List[IElementaryTask]:
        pass

    @abstractmethod
    def set_results_of_computations(self,
            elementary_results: List[Forecast]) -> ITaskResult:
        pass

    @abstractmethod
    def history(self) -> TimeSeries:
        pass
```

Объявление класса `TrainingTask` приведено в листинге 5.12. В конструкторе данного класса присутствуют несколько параметров, которых нет в конструкторе `BasicTask`: `training_start_index`, `compressors` и `horizon`. `training_start_index` задаёт номер элемента временного ряда, начиная с которого будут строиться прогнозы на `horizon` шагов вперёд. Параметр `compressors` содержит комбинации алгоритмов, с помощью которых будут построены прогнозы. `compressors` и `horizon` нужны для внутреннего пользования реализации `TrainingTask`, поэтому здесь они указаны явно (в `BasicTask` они передаются через `args`/`kwargs`). Чтобы избавить пользователя от необходимости передавать тип элементарной задачи и сообщить ему список всех доступных параметров, созданы классы `DiscreteUnivariateTrainingTask`, `RealUnivariateTrainingTask` и `RealMultivariateTrainingTask`, производные от `TrainingTask` и содержащие только конструкторы.

Листинг 5.11: Объявление класса `BasicTask`

```
class BasicTask(ITask):
    def __init__(self, task_result_handler: IBasicTaskResult,
            elementary_task_type: Type,
            time_series: TimeSeries,
            *args, **kwargs):
        pass

    def get_elementary_tasks(self) -> List[IElementaryTask]:
        pass

    def set_results_of_computations(self, results)
            -> IBasicTaskResult:
        pass

    def history(self) -> TimeSeries:
        pass
```

Листинг 5.12: Объявление класса `TrainingTask`

```
class TrainingTask(ITask):
    def __init__(self,
            task_result_handler: ITrainingTaskResult,
            elementary_task_type: Type,
            time_series: TimeSeries,
            training_start_index: int,
            compressors: List[ConcatenatedCompressorGroup],
            horizon: int, *args, **kwargs):
        pass

    def get_elementary_tasks(self) -> List[IElementaryTask]:
        pass

    def set_results_of_computations(self, results)
            -> ITrainingTaskResult:
        pass

    def history(self) -> TimeSeries:
        pass
```

Для визуализации результатов выполнения задач предусмотрены классы `TextVisualizer` и `PlotVisualizer`, реализующие интерфейс `IVisualizer`, приведённый в листинге 5.13. Единственный метод `visualize`, который содержит этот интерфейс, принимает в качестве входного параметра результат выполнения задачи. Реализация этого метода в `TextVisualizer` печатает на экран прогнозные значения и, при наличии, доверительные интервалы, средние абсолютные и относительные ошибки прогнозов для каждого шага. Для `PlotVisualizer` данный метод строит график, на котором изображены исторические значения, прогнозные значения с доверительными интервалами (при наличии), а также легенда. График строится с использованием библиотеки matplotlib и по умолчанию сохраняется в файл формата eps. Примером графика, полученного с использованием `PlotVisualizer`, является рисунок 4.5.

Листинг 5.13: Объявление класса `IVisualizer`

```
class IVisualizer:
    @abstractmethod
    def visualize(self, task_result: ITaskResult) -> None:
        pass
```

Далее рассмотрим реализацию пула задач. В пакете присутствует две реализации пула: `SequentialTaskPool` и `MpiTaskPool`. Оба класса являются производными от абстрактного класса `TaskPool`, объявление которого приведено в листинге 5.14.

Листинг 5.14: Объявление класса `TaskPool`

```
class TaskPool(ABC):
    @abstractmethod
    def add_task(self, task: ITask,
            *visualizers: Tuple[Visualizer]):
        pass

    @abstractmethod
    def execute(self):
        pass
```

Метод `add_task` предназначен для добавления задач в пул. Его первый аргумент должен содержать ссылку на объект добавляемой задачи, а второй — ссылки на объекты, с помощью которых нужно визуализировать результаты выполнения данной задачи. Метод `execute` непосредственно запускает процедуру вычислений. Он блокирует вызывающий поток до завершения выполнения всех задач пулом. В обеих реализациях пула при вызове данного метода формируется единый список элементарных задач, в который входят элементарные задачи всех зарегистрированных через `add_task` задач. `SequentialTaskPool` выполняет все задачи последовательно, а `MpiTaskPool` использует пакет mpi4py, который предоставляет доступ к реализации стандарта MPI (Message Passing Interface) из Python, с целью организации параллельных вычислений. Элементарные задачи равномерно распределяются по доступным вычислительным узлам, их результаты затем собираются в корневом узле (по умолчанию таким узлом является узел с номером 0) и затем передаются в объекты визуализации.

Также в пакете `itp` содержится реализация преобразования модели Хольта-Уинтерса к методу сжатия, предназначенная для использования в `itp_core`. Она представлена классом `HoltWinters`, который является производным от `INonCompressionAlgorithm`. Его реализация основана на использовании класса `statsmodels.tsa.api.ExponentialSmoothing` из пакета `statsmodels`. Все параметры для модели Хольта-Уинтерса, которые описаны в документации к `ExponentialSmoothing`, поддерживаются и при работе с `HoltWinters` — они могут быть переданы через конструктор или специально предназначенный для их изменения метод `ResetMethodParameters`.

Заключение

В соответствии с поставленной целью и задачами исследования были получены следующие результаты.

1. Разработан метод прогнозирования временных рядов, основанный на использовании сжатия данных с целью объединения нескольких методов прогнозирования в один, а также для использования алгоритмов искусственного интеллекта, реализованных в методах сжатия (таких, как построение компактной контекстно-свободной грамматики, из которой однозначно выводятся сжимаемые данные). В предлагаемом методе произвольные алгоритмы прогнозирования временных рядов преобразуются к методам сжатия и могут быть использованы совместно с универсальными кодами для вычисления прогнозов;
2. Разработан алгоритм прогнозирования временных рядов на основе многоголовочных конечных автоматов. Данный алгоритм способен правильно прогнозировать полилинейные слова, другие методы прогнозирования временных рядов этого делать не способны. За счёт объединения этого алгоритма с универсальными кодами, в том числе основанными на грамматических моделях, получен метод прогнозирования, способный находить «сложные» скрытые нестационарные закономерности в данных и использовать их для повышения точности прогнозов;
3. Построен адаптивный метод прогнозирования, базирующийся на универсальных по времени кодах и позволяющий существенно сократить время вычислений при объединении нескольких методов прогнозирования в один с помощью теоретико-информационного подхода. Использование данного метода при прогнозировании ряда солнечных пятен позволило сократить время вычислений более чем в 17 раз без потери точности прогнозов;
4. Выполнена программная реализация всех предложенных методов. Она включает в себя библиотеку itp_core, в которой поддержана работа с 7 алгоритмами сжатия данных, основанными на разных принципах, а также с предложенным в рамках настоящей работы алгоритмом на основе многоголовочных автоматов. На базе itp_core разработан пакет itp для Python, в

котором реализовано преобразование для модели Хольта-Уинтерса, позволяющее рассматривать её как метод сжатия и использовать в itp_core, поддержана возможность организации параллельных вычислений, а также содержатся методы анализа и визуализации прогнозов;

5. С использованием пакета itp построены прогнозы для социально-экономических показателей Новосибирской области, временного ряда солнечных пятен, временных рядов T-индекса и планетарного K-индекса, а также рядов из M3 Competition. Проведено сравнение точности прогнозов, полученных с использованием предлагаемых методов, с точностью прогнозов для тех же данных, построенных сторонними исследователями и организациями, а также выполнено сравнение прогнозных значений с реальными значениями, неизвестными на момент прогнозирования. Результаты экспериментального исследования показывают, что предлагаемые методы обладают высокой точностью и могут быть использованы для прогнозирования временных рядов экономических, социальных, физических и других показателей.

В дальнейших исследованиях можно рассмотреть более сложные подходы к выбору близкого к оптимальному алгоритма в адаптивном методе. Например, использовать для этой цели методы многомерной оптимизации.

Список литературы

1. *Ghysels E.*, *Marcellino M.* Applied economic forecasting using time series methods. — New York : Oxford University Press, 2018. — 616 p.
2. *Franses P. H.*, *Dijk D.*, *Opschoor A.* Time Series Models for Business and Economic Forecasting (2nd ed.) — Cambridge : Cambridge University Press, 2014. — 311 p.
3. *Kim K.* Financial time series forecasting using support vector machines // Neurocomputing. — 2003. — Vol. 55, no. 1. — P. 307–319.
4. *Chimmula V. K. R.*, *Zhang L.* Time series forecasting of COVID-19 transmission in Canada using LSTM networks // Chaos, Solitons & Fractals. — 2020. — Vol. 135. — P. 1–6.
5. *Baum C. F.*, *Hurn S.* Environmental Econometrics Using Stata. — College Station, Texas : Stata Press, 2021. — 416 p.
6. Time Series Analysis for the Social Sciences / J. M. Box-Steffensmeier [et al.]. — Cambridge : Cambridge University Press, 2014. — 297 p.
7. *Brown R. G.* Statistical forecasting for inventory control. — New York : McGraw/Hill, 1959. — 232 p.
8. *Brown R. G.* Smoothing, forecasting and prediction of discrete time series. — New Jersey : Prentice-Hall, 1963. — 468 p.
9. *Holt C. C.* Forecasting seasonals and trends by exponentially weighted moving averages // International Journal of Forecasting. — 2004. — Vol. 20, no. 1. — P. 5–10.
10. *Winters P. R.* Forecasting sales by exponentially weighted moving averages // Management Science. — 1960. — Vol. 6, no. 3. — P. 324–342.
11. *Hyndman R.*, *Athanasopoulos G.* Forecasting: principles and practice. — OTexts, 2018. — 382 p.
12. *Gardner E.*, *McKenzie E.* Forecasting trends in time series // Management science. — 1985. — Vol. 31, no. 10. — P. 1237–1246.

13. *Snyder R. D.* Recursive estimation of dynamic linear models // Journal of the Royal Statistical Society. Series B (Methodological). — 1985. — Vol. 47, no. 2. — P. 272–276.

14. *Ord J.*, *Koehler A.*, *Snyder R.* Estimation and prediction for a class of dynamic nonlinear statistical models // Journal of the American Statistical Association. — 1997. — Vol. 92, no. 440. — P. 1621–1629.

15. A state space framework for automatic forecasting using exponential smoothing methods / R. Hyndman [et al.] // International Journal of Forecasting. — 2002. — Vol. 18, no. 3. — P. 439–454.

16. *Taylor J.* Exponential smoothing with a damped multiplicative trend // International Journal of Forecasting. — 2003. — Vol. 19, no. 4. — P. 715–725.

17. Forecasting with exponential smoothing: the state space approach / R. Hyndman [et al.]. — Berlin/Heidelberg, Germany : Springer-Verlag, 2008. — 375 p.

18. *De Silva A.*, *Hyndman R. J.*, *Snyder R.* The vector innovations structural time series framework: a simple approach to multivariate forecasting // Statistical Modelling. — 2010. — Vol. 10, no. 4. — P. 353–374.

19. *Makridakis S.*, *Hibon M.* The M3-Competition: results, conclusions and implications // International Journal of Forecasting. — 2000. — Vol. 16, no. 4. — P. 451–476.

20. *Yule G.* On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers // Philosophical Transactions of the Royal Society of London Series A, Containing Papers of a Mathematical or Physical Character. — 1927. — Vol. 226, no. 636–646. — P. 267–298.

21. *Walker G.* On periodicity in series of related terms // Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character. — 1931. — Vol. 131, no. 818. — P. 518–532.

22. *Слуцкий Е.* Сложение случайных причин как источник циклических процессов // Вопросы конъюнктуры. — 1927. — Т. 3, № 1. — С. 34–64.

23. *Wold H.* A study in the analysis of stationary time series. — Uppsala : Almqvist & Wiksells, 1938. — 214 p.

24. *Whittle P.* Hypothesis testing in time series analysis. — Uppsala : Almqvist & Wiksells, 1951. — 120 p.

25. *Box G. E.*, *Jenkins G.* Time series analysis: forecasting and control. — San Francisco : Holden-Day, 1970. — 553 p.

26. *Akaike H.* A new look at the statistical model identification // IEEE Transactions on Automatic Control. — 1974. — Vol. 19, no. 6. — P. 716–723.

27. *Schwarz G.* Estimating the dimension of a model // Annals of Statistics. — 1978. — Vol. 6, no. 2. — P. 461–464.

28. *Parzen E.* ARARMA models for time series analysis and forecasting // Journal of Forecasting. — 1982. — Vol. 1, no. 1. — P. 67–82.

29. *Quenouille M.* The analysis of multiple time-series. — London : Griffin, 1957. — 105 p.

30. *Granger C.*, *Joyeux R.* An introduction to long-memory time series models and fractional differencing // Journal of Time Series Analysis. — 1980. — Vol. 1, no. 1. — P. 15–29.

31. *Box G.*, *Tiao G.* A new look at the statistical model identification // Intervention analysis with applications to economic and environmental problems. — 1975. — Vol. 70, no. 349. — P. 70–79.

32. *Engle R.* Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation // Econometrica. — 1982. — Vol. 50, no. 4. — P. 987–1007.

33. *Bollerslev T.* Generalized autoregressive conditional heteroskedasticity // Econometrica. — 1986. — Vol. 31, no. 3. — P. 307–327.

34. *Bollerslev T.* Glossary to ARCH (GARCH) // CREATES Research paper. — 2008. — Vol. 49. — P. 1–46.

35. *Bauwens L.*, *Laurent S.*, *Rombouts J.* Multivariate GARCH models: a survey // Journal of Applied Econometrics. — 2006. — Vol. 21, no. 1. — P. 79–109.

36. *Irie B.*, *Miyake S.* Capabilities of three-layered perceptrons // IEEE 1988 International Conference on Neural Networks. — San Diego, CA, USA, 1988. — P. 641–648.

37. *Zhang G.*, *Patuwo B.*, *Hu M.* Forecasting with artificial neural networks: The state of the art // International Journal of Forecasting. — 1998. — Vol. 14, no. 1. — P. 35–62.

38. *De Gooijer J.*, *Hyndman R.* 25 years of time series forecasting // International Journal of Forecasting. — 2006. — Vol. 22, no. 3. — P. 443–473.

39. *Hewamalage H.*, *Bergmeir C.*, *Bandara K.* Recurrent neural networks for time series forecasting: Current status and future directions // International Journal of Forecasting. — 2021. — Vol. 37, no. 1. — P. 388–427.

40. *Tealab A.* Time series forecasting using artificial neural networks methodologies: A systematic review // Future Computing and Informatics Journal. — 2018. — Vol. 3, no. 2. — P. 334–340.

41. *Makridakis S.*, *Spiliotis E.*, *Assimakopoulos V.* The M4 Competition: Results, findings, conclusion and way forward // International Journal of Forecasting. — 2018. — Vol. 34, no. 4. — P. 802–808.

42. *Smyl S.* A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting // International Journal of Forecasting. — 2020. — Vol. 36, no. 1. — P. 75–85.

43. *Рябко Б. Я.* Прогноз случайных последовательностей и универсальное кодирование // Проблемы передачи информации. — 1988. — Т. 24, № 2. — С. 3–14.

44. *Ryabko B.*, *Astola J.*, *Malyutov M.* Compression-based methods of statistical analysis and prediction of time series. — Cham, Switzerland : Springer International Publishing, 2016. — 144 p.

45. *Приставка П. А.* Экспериментальное исследование метода прогнозирования, основанного на универсальных кодах // Вестник СибГУТИ. — 2010. — № 4. — С. 26–35.

46. *Лысяк А. С.*, *Рябко Б. Я.* Методы прогнозирования временных рядов с большим алфавитом на основе универсальной меры и деревьев принятия решений // Вычислительные технологии. — 2014. — Т. 19, № 2. — С. 76–93.

47. *Ziv J.*, *Lempel A.* A universal algorithm for sequential data compression // IEEE Transactions on Information Theory. — 1977. — Vol. 23, no. 3. — P. 337–343.

48. *Ziv J.*, *Lempel A.* Compression of individual sequences via variable-rate coding // IEEE Transactions on Information Theory. — 1978. — Vol. 24, no. 5. — P. 530–536.

49. *Welch T.* Technique for high-performance data compression // Computer. — 1984. — Vol. 17, no. 6. — P. 8–19.

50. *Burrows M.*, *Wheeler D.* A block-sorting lossless data compression algorithm : Tech. Rept. 124. — Palo Alto, CA, USA, 1994. — 18 p.

51. *Manzini G.* The Burrows-Wheeler transform: theory and practice // International Symposium on Mathematical Foundations of Computer Science. — Szklarska Poreba, Poland, 1999. — P. 34–47.

52. *Рябко Б. Я.*, *Пестунов А. И.* "Стопка книг"как новый статистический тест для случайных чисел // Проблемы передачи информации. — 2004. — Т. 40, № 1. — С. 73–78.

53. *Cleary J.*, *Witten I.* Data compression using adaptive coding and partial string matching // IEEE Transactions on Communications. — 1984. — Vol. 32, no. 4. — P. 396–402.

54. *Witten I.*, *Bell T.* The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression // IEEE Transactions on Information Theory. — 1991. — Vol. 37, no. 4. — P. 1085–1094.

55. *Cleary J.*, *Teahan W.* Unbounded length contexts for PPM // The Computer Journal. — 1997. — Vol. 40, no. 2/3. — P. 67–75.

56. *Kieffer J.*, *Yang E.* Grammar-based codes: A new class of universal lossless source codes // IEEE Transactions on Information Theory. — 2000. — Vol. 46, no. 3. — P. 737–754.

57. *Nevill-Manning C.*, *Witten I.* Identifying hierarchical structure in sequences: A linear-time algorithm // Journal of Artificial Intelligence Research. — 1997. — Vol. 7. — P. 67–82.

58. A space-saving approximation algorithm for grammar-based compression / H. Sakamoto [et al.] // IEICE Transactions on Information and Systems. — 2009. — Vol. 92, no. 2. — P. 158–165.

59. *Witten I. H.*, *Neal R. M.*, *Cleary J. G.* Arithmetic coding for data compression // Communications of the ACM. — 1987. — Vol. 30, no. 6. — P. 520–540.

60. *Gallager R. G.* Information theory and reliable communication. — New York : John Wiley & Sons, 1968. — 588 p.

61. *Cover T. M.* Elements of information theory. — Hoboken, New Jersey : Wiley-Interscience, 2006. — 776 p.

62. *Ryabko B.* Applications of Kolmogorov complexity and universal codes to nonparametric estimation of characteristics of time series // Fundamenta Informaticae. — 2008. — Vol. 83, no. 1/2. — P. 177–196.

63. Compilers: Principles, Techniques, and Tools, 2nd Edition / A. V. Aho [et al.]. — Boston, MA, USA : Pearson Education, 2007. — 1040 p.

64. *Jurafsky D.*, *Martin J. H.* Speech and Language Processing, 2nd Edition. — New Jersey : Prentice Hall, 2008. — 1032 p.

65. *Rozenberg G.*, *Salomaa A.* Lindenmayer systems: impacts on theoretical computer science, computer graphics, and developmental biology. — Berlin/Heidelberg, Germany : Springer-Verlag, 1992. — 523 p.

66. *Chomsky N.* On certain formal properties of grammars // Information and Control. — 1959. — Vol. 2, no. 2. — P. 137–167.

67. The smallest grammar problem / M. Charikar [et al.] // IEEE Transactions on Information Theory. — 2005. — Vol. 51, no. 7. — P. 2554–2576.

68. *Maruyama S.*, *Sakamoto H.*, *Takeda M.* An online algorithm for lightweight grammar-based compression // Algorithms. — 2012. — Vol. 5, no. 2. — P. 214–235.

69. *Yang E.*, *Kieffer J.* Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform. I. Without context models // IEEE Transactions on Information Theory. — 2000. — Vol. 46, no. 3. — P. 755–777.

70. Universal lossless compression via multilevel pattern matching / J. Kieffer [et al.] // IEEE Transactions on Information Theory. — 2000. — Vol. 46, no. 4. — P. 1227–1245.

71. *Hucke D.*, *Lohrey M.*, *Reh C. P.* The smallest grammar problem revisited // International Symposium on String Processing and Information Retrieval. — Beppu, Japan, 2016. — P. 35–49.

72. *Larsson N.*, *Moffat A.* Off-line dictionary-based compression // Proceedings of the IEEE. — 2000. — Vol. 88, no. 11. — P. 1722–1732.

73. *Rytter W.* Application of Lempel–Ziv factorization to the approximation of grammar-based compression // Theoretical Computer Science. — 2003. — Vol. 302, no. 1. — P. 211–222.

74. *Sakamoto H.* A fully linear-time approximation algorithm for grammar-based compression // Journal of Discrete Algorithms. — 2005. — Vol. 3, no. 2–4. — P. 416–430.

75. *Sakamoto H.*, *Kida T.*, *Shimozono S.* A space-saving linear-time algorithm for grammar-based compression // International Symposium on String Processing and Information Retrieval. — Padua, Italy, 2004. — P. 218–229.

76. *Jeż A.* A really simple approximation of smallest grammar // Theoretical Computer Science. — 2016. — Vol. 616. — P. 141–150.

77. *Bille P.*, *Gørtz I. L.*, *Prezza N.* Space-efficient re-pair compression // 2017 Data Compression Conference (DCC). — Snowbird, Utah, USA, 2017. — P. 171–180.

78. *Smith T.* Prediction of infinite words with automata // Theory of Computing Systems. — 2018. — Vol. 62, no. 3. — P. 653–681.

79. *Smith T.* On infinite words determined by stack automata // IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013). — IIT Guwahati, India, 2013. — P. 413–424.

80. *Hopcroft J. E.*, *Ullman J. D.* Introduction to automata theory, languages, and computation. — Addison-Wesley, 1979. — 418 p.

81. *Кричевский Р.* Связь между избыточностью кодирования и достоверностью сведений об источнике // Проблемы передачи информации. — 1968. — Т. 4, № 3. — С. 48–57.

82. *Кричевский Р. Е.* Сжатие и поиск информации. — М. : Радио и связь, 1989. — 168 с.

83. *Ryabko B.* Time-Universal data compression // Algorithms. — 2019. — Vol. 12, no. 6. — P. 1–10.

84. *Wheelwright S.*, *Makridakis S.*, *Hyndman R. J.* Forecasting: methods and applications. — New York : John Wiley & Sons, 1998. — 656 p.

85. STL: A seasonal-trend decomposition / R. B. Cleveland [et al.] // Journal of Official Statistics. — 1990. — Vol. 6, no. 1. — P. 3–73.

86. *Hathaway D.* The solar cycle // Living reviews in solar physics. — 2015. — Vol. 12, no. 1. — P. 1–87.

87. *Clark D.*, *Stephenson F.* An interpretation of the pre-telescopic sunspot records from the Orient // Quarterly Journal of the Royal Astronomical Society. — 1978. — Vol. 19. — P. 387–410.

88. *SILSO World Data Center*. The International Sunspot Number [Электронный ресурс] // International Sunspot Number Monthly Bulletin and online catalogue. — Royal Observatory of Belgium, Brussels, Belgium, 1749–2020. — Режим доступа: http://www.sidc.be/silso/ (дата обращения: 31.02.2022).

89. *Bartels J.*, *Heck N.*, *Johnston H.* The three-hour-range index measuring geomagnetic activity // Terrestrial Magnetism and Atmospheric Electricity. — 1939. — Vol. 44, no. 4. — P. 411–454.

90. *Space Weather Prediction Center*. Planetary K-index [Электронный ресурс] // National Oceanic and Atmospheric Administration. — Boulder CO, USA, 2022. — Режим доступа: https://www.swpc.noaa.gov/products/planetary-k-index (дата обращения: 31.01.2022).

91. *Thompson R.* T Index FAQ [Электронный ресурс] // Space Weather Services. — Bureau of Meteorology, Australia, 2022. — Режим доступа: https://www.sws.bom.gov.au/Educational/5/2/1 (дата обращения: 31.01.2022).

92. *Чирихин К. С.*, *Рябко Б. Я.* Применение методов искусственного интеллекта и сжатия данных для прогнозирования социальных, экономических и демографических показателей Новосибирской области // Вычислительные технологии. — 2020. — Т. 25, № 5. — С. 80–90.

93. *Chirikhin K.*, *Ryabko B.* Compression-Based Methods of Time Series Forecasting // Mathematics. — 2021. — Vol. 9, no. 3. — P. 1–11.

94. *Чирихин К. С.* Теоретико-информационный метод интеграции различных алгоритмов прогнозирования временных рядов // Тезисы XXI Всероссийской конференции молодых учёных по математическому моделированию и информационным технологиям. — Новосибирск, 2020. — С. 44.

95. *Chirikhin K.* Application of time-universal codes to time series forecasting // 34th Annual European Simulation and Modelling Conference, ESM 2020. — Toulouse, France, 2020. — P. 60–64.

96. *Чирихин К. С.* Реализация алгоритма прогнозирования временных рядов на основе методов сжатия данных и искусственного интеллекта // Тезисы

Российской научно-технической конференции «Обработка информации и математическое моделирование». — Новосибирск, 2020. — С. 195–199.

97. *Чирихин К. С.* Применение методов сжатия данных и искусственного интеллекта для прогнозирования демографических и экономических показателей Новосибирской области // Распределенные информационно-вычислительные ресурсы. Цифровые двойники и большие данные.(DICR-2019). — Новосибирск, 2019. — С. 238–243.

98. *Chirikhin K. S.*, *Ryabko B. Y.* Application of articial intelligence and data compression methods to time series forecasting // Applied Methods of Statistical Analysis. Statistical Computation and Simulation - AMSA’2019: Proceedings of the International Workshop. — Novosibirsk, 2019. — P. 553–560.

99. *Chirikhin K. S.*, *Ryabko B. Y.* Application of data compression techniques to time series forecasting // The 39th International Symposium on Forecasting. — Thessaloniki, Greece, 2019. — P. 12.

100. *Чирихин К. С.*, *Рябко Б. Я.* Экспериментальное исследование точности методов прогноза, базирующихся на архиваторах // Вестник Новосибирского государственного университета. Серия: Информационные технологии. — 2018. — Т. 16, № 3. — С. 145–158.

Приложение А. Акты о внедрении

Акт о внедрении результатов диссертации в ФИЦ ИВТ

УТВЕРЖДАЮ
И.о. директора ФИЦ ИВТ
д.ф-м.н. С.Б. Медведев
«11» апреля 2022 г.

АКТ ВНЕДРЕНИЯ
результатов диссертационной работы Чирихина К.С.
«Использование методов теории информации и искусственного интеллекта для разработки и исследования высокоточных методов прогнозирования временных рядов»

Комиссия в составе: д.т.н., проф. Фионов А.Н. (председатель), д.ф.-.м.н., проф. Чубаров Л.Б., к.т.н. Ракитский А.А., констатирует, что результаты диссертационной работы Чирихина Константина Сергеевича использованы при выполнении работ по теме № FWNW-2021-0005 (Регистрационный номер 122010800027-7) «Разработка, создание и исследование распределенных информационных систем для поддержки принятия решений и автоматизации процессов» и включены в отчёт (Регистрационный номер 222020900300-8). С помощью предложенных в рамках диссертационной работы методов были построены прогнозы динамики совокупности экономических и демографических показателей Новосибирской области, при этом методы подтвердили свою высокую эффективность.

Председатель комиссии /Фионов А.Н./
Члены комиссии /Чубаров Л.Б./
/Ракитский А.А./

«11» апреля 2022 г.

Акт о внедрении результатов диссертации в учебный процесс на кафедре КС НГУ

«УТВЕРЖДАЮ»
ректор НГУ
академик РАН, д.ф-м.н.,
проф. М.П. Федорук
«17» 03 2022 г.

АКТ ВНЕДРЕНИЯ
результатов диссертационной работы Чирихина К.С.
«Использование методов теории информации и искусственного интеллекта для разработки и исследования высокоточных методов прогнозирования временных рядов»

Комиссия в составе: председатель комиссии Лаврентьев М.М., д.ф-м.н., проф., декан ФИТ и члены комиссии Пищик Б.Н., к.т.н., и.о. заведующего кафедрой Компьютерных систем ФИТ, Рябко Б.Я., д.т.н., профессор кафедры Компьютерных систем ФИТ констатирует, что результаты диссертационной работы Чирихина К.С. внедрены в учебный процесс на кафедре Компьютерных систем в курсе «Информационные технологии на основе методов криптографии» (аспирантура) по направлению подготовки 09.06.01 Информатика и вычислительная техника.

Председатель комиссии /Лаврентьев М.М./
Члены комиссии /Пищик Б.Н./
/Рябко Б.Я./

«10» 03 2022 г.

Акт о внедрении результатов диссертации в учебный процесс на кафедре ПМиК СибГУТИ

"УТВЕРЖДАЮ"
И.о. ректора
ФГБОУ ВО СибГУТИ
Б.Г.Хаиров
«14» 03 2022 г.

АКТ ВНЕДРЕНИЯ
результатов диссертационной работы Чирихина К.С.
«Использование методов теории информации и искусственного интеллекта для разработки и исследования высокоточных методов прогнозирования временных рядов»

Комиссия в составе:

И.о. директора института ИВТ, к.т.н., доцент	Приставка П.А. (председатель)
Зав. каф. ПМиК, д.т.н., доцент	Нечта И.В.
Доцент кафедры ПМиК, к.т.н.	Ракитский А.А.

констатирует, что результаты диссертационной работы Чирихина К.С. внедрены в учебный процесс на кафедре ПМиК в программе курса «Методы машинного обучения» (бакалавриат) по направлению подготовки 09.03.01 «Информатика и вычислительная техника».

Председатель комиссии /Приставка П.А./
Члены комиссии /Нечта И.В./
/Ракитский А.А./

Приложение Б. Свидетельство о государственной регистрации программы для ЭВМ

Свидетельство о государственной регистрации пакета для Python «Information-theoretic predictor»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2022613783

Пакет для Python «Information-theoretic predictor»

Правообладатель: *Чирихин Константин Сергеевич (RU)*

Автор(ы): *Чирихин Константин Сергеевич (RU)*

Заявка № 2022612407
Дата поступления 18 февраля 2022 г.
Дата государственной регистрации
в Реестре программ для ЭВМ *15 марта 2022 г.*

Руководитель Федеральной службы
по интеллектуальной собственности

Ю.С. Зубов