Курсовая работа защищена с оценкой ____

Ученый секретарь кафедры Теории вероятностей Илларионов Е. А.

Московский государственный университет имени М.В. Ломоносова

Механико-математический факультет

Кафедра Теории вероятностей

КУРСОВАЯ РАБОТА

Исследование вариантов архитектур нейронных сетей с памятью на основе нечеткой логики в задаче прогнозирования финансовых временных рядов

Выполнена студенткой 509 группы

Маховой Анастасией Геннадьевной

Научный руководитель:

д.ф.-м.н М.И. Кумсков

Москва, 2025

Аннотация

В данной работе рассматриваются архитектуры нейронных сетей с памятью, использующие принципы нечеткой логики для прогнозирования финансовых временных рядов. Нечеткая логика применяется как на этапе предобработки данных, так и внутри модели, обеспечивая гибкое представление входных характеристик для более точного прогноза.

Прогнозирование финансовых временных рядов представляет собой сложную задачу из-за высокой волатильности и наличия нелинейных зависимостей с историческими данными. Для ее решения используется LSTM (Long Short-Term Memory) — модель, зарекомендовавшая себя в работе с временными рядами.

На этапе предобработки применяется фаззификация входных данных, позволяющая представить их в виде нечетких множеств. Это способствует снижению чувствительности модели к шуму и выделению значимых паттернов в данных.

Внутри модели фаззификация осуществляется через функции активации, которые рассматриваются как функции принадлежности к нечетким множествам. Это позволяет нейросети проводить операции нечеткой логики, регулируя наклон и смещение точки нуля функций активации. Тем самым достигается адаптивная настройка модели под специфику конкретных финансовых данных, что позволяет лучше улавливать сложные зависимости, снижая влияние шума в исходном ряде.

Результаты вычислительных экспериментов показывают, что предложенный подход демонстрирует лучшую предсказательную способность, чем стандартная LSTM-модель.

Ключевые слова: Финансовые временные ряды, функция активации, нечеткая логика, нейронные сети с памятью

Содержание

Введение

Прогнозирование финансовых временных рядов — одна из ключевых задач в экономике и финансах. Оно применяется в алгоритмической торговле, управлении инвестициями, оценке рисков и макроэкономическом анализе. Однако такие особенности финансовых временных рядов как нестационарность, наличие нелинейных зависимостей и шума [1] делают эту задачу сложной для традиционных статистических моделей, таких как ARIMA и GARCH[2].

В последние годы значительное внимание привлекают методы глубокого обучения, особенно рекуррентные нейросети (RNN) и их усовершенствованные версии, такие как нейросети с долгой краткосрочной памятью (Long Short-Term Memory, LSTM)[38]. Эти модели способны улавливать сложные временные зависимости и эффективно работать с временными рядами. Однако стандартные LSTM-модели чувствительны к шуму в данных и используют фиксированные функции активации, что ограничивает их адаптивность к специфике временного ряда.

В данной работе предлагается использование нечеткой логики (fuzzy logic) для улучшения прогнозирования финансовых временных рядов. Нечеткая логика позволяет гибко обрабатывать данные, учитывая их неопределенность и варьируемость. Ее применение возможно на нескольких этапах работы нейросети:

Фаззификация входных данных — преобразование числовых значений в нечеткие множества, что позволяет сглаживать шум. В работе [3] рассматриваются варианты представления нечетких временных рядов: распределение нечеткости моделируемое лингвистической переменной, распределение нечеткости во времени или по переменной значения. Колеблющиеся нечеткие множества(Hesitant Fuzzy Set (HFS)) применяются для прогнозирования баллов для поступления в университет Алабамы и цен акций индийских компаний в работе [4].

Использование нечетких функций активации – изменение наклона и сдвига функций активации, что позволяет подстраиваться под особенности конкретного временного ряда. В предыдущей работе авторов данной статьи[6] было показано, что функции активации *cloglog* и *loglog*, которые можно рассмотреть как преобразования сигмоидальной функции активации, улучшают прогноз. В литературе встречаются свежие работы о параметрических [7] и нечетких [8] функциях активации.

Основной целью работы является исследование архитектур нейронных сетей с памятью, использующих принципы нечеткой логики для повышения точности прогнозирования финансовых временных рядов. Для достижения этой цели в работе решаются следующие задачи:

- Изучение теоретических основ нечеткой логики и ее применения в машинном обучении.
- Разработка и тестирование метода фаззификации входных данных.
- Внедрение модифицированных функций активации на основе нечеткой логики в LSTM-сеть.
- Сравнительный анализ точности предложенного метода с классическими моделями (стандартной LSTM, ARIMA и др.).

Статья организована следующим образом. В разделе 1 рассматриваются основные теоретические аспекты, необходимые в работе: принципы нечеткой логики, роль функции активации, особенности финансовых временных рядов. В разделе 2 приводится постановка задачи. Принцип работы LSTM описан в разделе 3. В разделе 4 дан обзор на варианты функций потерь, алгоритмов оптимизации, метрик оценки качества и функций активации – важнейших компонентов, необходимых для работы нейросети. Кроме того, в этой главе отмечены те функции и алгоритмы, которые подходят для решения задачи прогнозирования финансовых временных рядов. Раздел 5 содержит перечень используемого программного обеспечения. В разделе 6 представлена разработанная модель, приведены результаты вычислительных экспериментов и их анализ. В Заключении подводятся итоги исследования и рассматриваются возможные направления дальнейшей работы.

1 Основные определения и нотация

1.1 Финансовые временные ряды

Определение 1.1. Временной ряд X представляет собой последовательность хронологически упорядоченных вещественных векторов:

$$X = (x_{t_1}, x_{t_2}, \cdots, x_{t_n}), \quad x_{t_i} \in \mathbb{R}^m.$$

Число n обозначается как $|X|$ и называется **длиной ряда**.
Постоянный шаг по времени назовем **тиком**: $\Delta t = t_i - t_{i-1}$

Определение 1.2. Финансовый временной ряд имеет ряд особенностей, в частности, нелинейный характер формирования значений финансовых индексов и последействие, выражающееся в том, что многие индексы, цены и тд. "помнят"прошлое.[1]

1.2 Задачи прогнозирования финансовых временных рядов

Задачи о прогнозировании финансовых временных рядов можно разделить на 2 группы в зависимости от ожидаемых выходных данных[38]:

прогноз цены – необходимо предсказать стоимость на некоторое время вперед.

прогноз тренда – направления движения графика стоимости:

- 2-class problem – предсказать восходящий и нисходящий тренд
- 3-class problem - предсказать восходящий, нисходящий и боковой тренды

Входные данные для прогнозирования можно брать из:

- **Цен за предыдущие периоды**
- **Фундаментального анализа** Благодаря методам фундаментального анализа можно проанализировать справедливую стоимость на данный момент и предсказать повышение/понижение спроса
- **Технического анализа** Анализ графиков на характерные предпосылки к изменению направления движения стоимости. На рисунке 1 изображены основные виды графиков, которые используются для отображения цен во времени.

- **Текстовой информации** Обработка новостей и определение эмоционального настроя и мнений на основе постов и комментариев в социальных сетях помогают спрогнозировать поведение инвесторов.

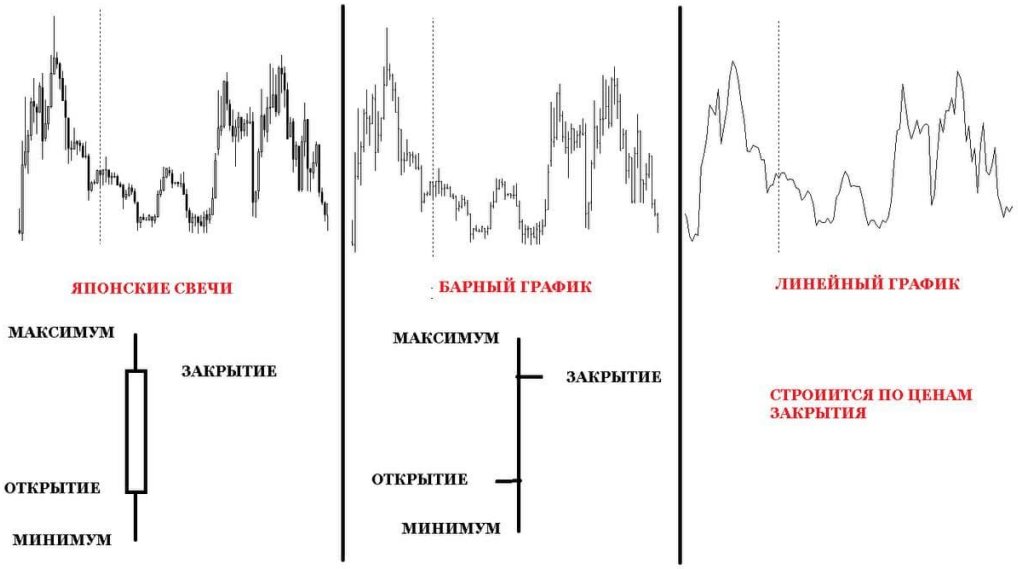

Рис. 1: виды графиков

1.3 Роль функции активации

Основа искусственных нейронных сетей — это *нейрон* и *функция активации*

Определение 1.3. Нейрон – элемент, который принимает, обрабатывает и передает сигнал другим элементам нейросети.

Определение 1.4. Функция активации – это математическая функция, которая по нескольким входным сигналам определяет один выходной сигнал.

Функция активации определяет, как нейрон будет реагировать на входные данные: для входов вычисляется взвешенная сумма со смещением, выполняется преобразование с помощью функции активации, и в результате получается выходной сигнал.

$$y_j = \phi(\sum_i w_{ij} * x_i + b_j),$$

где x_i - входной сигнал, y_j - выходной сигнал, ϕ – функция активации, w_{ij} – весовой коэффициент и b_j – смещение.

Каждый нейрон по сути выполняет нелинейное преобразование входного сигнала в выходной по формуле выше. Саму нейросеть можно рассматривать как композицию взвешенных нейронов [12].

1.4 Нечеткая логика

Определение 1.5. нечеткое множество (fuzzy set) - это пара (U, m),
где U - универсальное множество,
$m : U \to [0, 1]$ - **функция принадлежности** (membership function).

Для каждого $x \in U$ величина $m(x)$ называется **степенью принадлежности** x к нечеткому множеству (U, m)

Для конечного $U = \{x_1, ..., x_n\}$ нечеткое множество (U, m) обозначается как

$$\{m(x_1)/x_1, ..., m(x_n)/x_n\}$$

Пусть $x \in U$, тогда:

$$m(x) = \begin{cases} 0 & x \text{ не лежит в нечетком множестве} \\ 1 & x \text{ полностью принадлежит нечеткому множеству (full member)} \\ \text{иначе} & x \text{ частично принадлежит нечеткому множеству (fuzzy member)} \end{cases}$$

1.4.1 Операции над нечеткими множествами

Пересечение двух нечетких множеств (нечеткое «И»): $m_{A \cap B}(x) = \min(m_A(x), m_B(x))$
Объединение двух нечетких множеств (нечеткое «ИЛИ»): $m_{A \cup B}(x) = \max(m_A(x), m_B(x))$

1.4.2 Переменные

Для описания нечетких множеств вводятся понятия нечеткой и лингвистической переменных.

Определение 1.6. Нечеткая переменная описывается набором (N,U,A),
где N — это название переменной,
U — универсальное множество (область рассуждений),
A — нечеткое множество на U.

Значениями **лингвистической переменной** могут быть нечеткие переменные, т.е. лингвистическая переменная находится на более высоком уровне, чем нечеткая переменная. Каждая лингвистическая переменная состоит из:

- названия;
- множества своих значений, которое также называется базовым терм-множеством T. Элементы базового терм-множества представляют собой названия нечетких переменных;
- универсального множества U;
- синтаксического правила G, по которому генерируются новые термы с применением слов естественного или формального языка;
- семантического правила P, которое каждому значению лингвистической переменной ставит в соответствие нечеткое подмножество множества U.

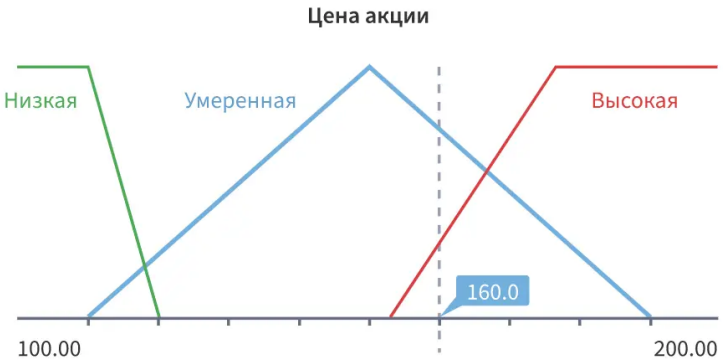

Рис. 2: Пример лингвистической переменной “Цена акции”. Нечеткие переменные: “низкая” , “умеренная” и “высокая”. Универсальное множество $U = [100, 200]$. Степень принадлежности цены 160 к множеству “низкая” - нулевая т.е. $m_{\text{низкая}}(160) = 0$, для множества “умеренная” $m_{\text{умеренная}}(160) = 0.75$ и для множества “высокая” $m_{\text{высокая}}(160) = 0.40$.

1.4.3 Функции принадлежности

Существует свыше десятка типовых форм кривых для задания функций принадлежности. Наибольшее распространение получили: треугольная, трапецеидальная и гауссова функции принадлежности.

Треугольная функция принадлежности определяется тройкой чисел (a, b, c), и ее значение в точке x вычисляется согласно выражению:

$$m(x) = \begin{cases} 1 - \frac{b-x}{b-a} & a \leq x \leq b \\ 1 - \frac{x-b}{c-b} & b \leq x \leq c \\ 0 & x \notin (a; c) \end{cases}$$

Аналогично для задания трапецеидальной функции принадлежности необходима четверка чисел (a, b, c, d):

$$m(x) = \begin{cases} 1 - \frac{b-x}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ 1 - \frac{x-b}{c-b} & c \leq x \leq d \\ 0 & x \notin (a; d) \end{cases}$$

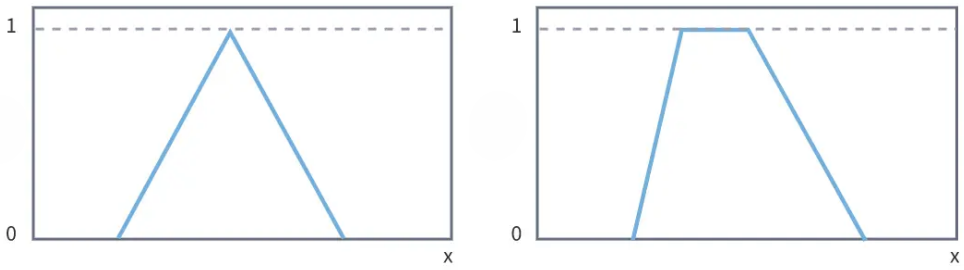

Рис. 3: Слева: треугольная функция активации. Справа: трапециевидная функция активации

Функция принадлежности гауссова типа описывается формулой:

$$m(x) = \exp(-(\frac{x-c}{\sigma})^2)$$

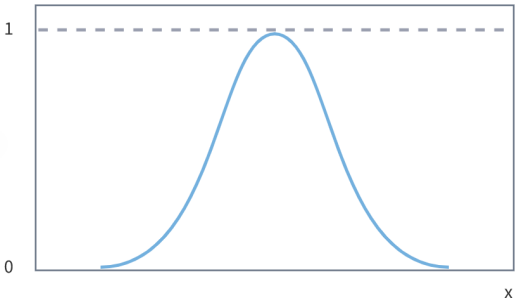

Рис. 4: Функция принадлежности гауссова типа.

Кроме того, функцией принадлежности могут быть и функции активации, чьи значения лежат в $[0, 1]$, наример, s-образные функции *sigmoid*, *cloglog*, *loglog* и их вариации со смещенным центром нуля и измененным наклоном.

1.4.4 Нечеткий логический вывод

Основой для проведения операции нечеткого логического вывода является *база правил*, содержащая нечеткие высказывания и функции принадлежности для соответствующих лингвистических термов.

Механизм логического вывода включает четыре этапа: введение нечеткости (фазификация), нечеткий вывод, композиция и приведение к четкости, или дефазификация (см. рисунок 2). Алгоритмы нечеткого вывода различаются, главным образом, видом используемых правил, логических операций и разновидностью метода дефазификации. Известны модели нечеткого вывода Мамдани, Сугено, Ларсена, Цукамото.[13]

Рис. 5: Механизм логического вывода.

1.4.5 Применение

В результате объединения нескольких технологий искусственного интеллекта появился специальный термин — «мягкие вычисления» (soft computing), который ввел Л. Заде в 1994 году. В настоящее время мягкие вычисления объединяют такие области как: нечеткая логика, искусственные нейронные сети, вероятностные рассуждения и эволюционные алгоритмы. Они дополняют друг друга и используются в различных комбинациях для создания гибридных интеллектуальных систем: Нечеткие нейронные сети (fuzzy-neural networks), Адаптивные нечеткие системы (adaptive fuzzy systems), Нечеткие запросы к базам данных (fuzzy queries) и др.

1.5 Выводы

В этом разделе даны основные определения о временных рядах, функциях активации и нечеткой логике, которые помогут понять постановку задачи.

2 Постановка задачи

Нам дан одномерный финансовый временной ряд цен

$$X = (x_{t_1}, x_{t_2}, \cdots, x_{t_n}), \quad x_{t_i} \in \mathbb{R}$$

с тиком $\Delta t = \{$минута, час, день$\}$

Нужно спрогнозировать значения ряда на k тиков вперед: $Y = (y_{t_{n+1}}, y_{t_{n+2}}, \cdots, y_{t_{n+k}})$

Общий вид модели будет напоминать схему логического вывода 2, где за нечеткий логический вывод будет отвечать нейронная сеть с модифицированными функциями активации, которая будет производить операции над нечеткими множествами с функциями принадлежности отождествленными с функциями активации.

Рис. 6: ЗДЕСЬ БУДЕТ ОБЩАЯ СХЕМА, ПОХОЖАЯ НА ЭТУ

2.1 Фазификация

Ввести нечеткость в финансовый временной ряд можно несколькими способами:

Гауссианы. Для каждого момента времени t_i сопоставим известному значению ряда x_i нечеткое множество, задающееся функцией принадлежности гауссовского типа

$$m(x) = \exp(-(\frac{x-c}{\sigma})^2)$$

с параметрами $c = x_i$ и $\sigma = \hat{\sigma}(t_i, l)$– волатильность ряда за период длины l: $[t_{i-l}, t_i]$. То есть вершина гауссовского купола будет как раз в x_i и его “ширина” будет соответствовать “ширине” обозреваемого ряда.

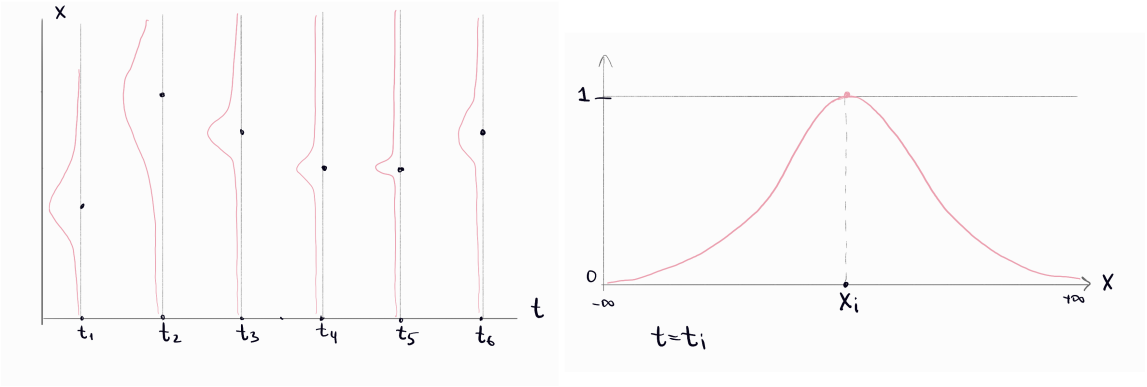

Рис. 7: Слева: черные точки – значение временного ряда x_i, красные “купола” - условное обозначение нечеткого временного ряда: каждому моменту времени t_i сопоставлено нечеткое множество. Справа “фотография” в фиксированный момент t соответствующего нечеткого множества.

Probabilistic Fuzzy Sets Этот метод описан в работе [5].

1. определим область рассуждений $U = [X_{min} - \sigma, X_{max} + \sigma], X_{min}$ и X_{max} – минимальные и максимальные значения ряда за обозреваемый период, σ – дисперсия.
2. Разобьем U на m равных интервалов e_i и зададим на них треугольными функциями принадлежности нечеткие множества F_i

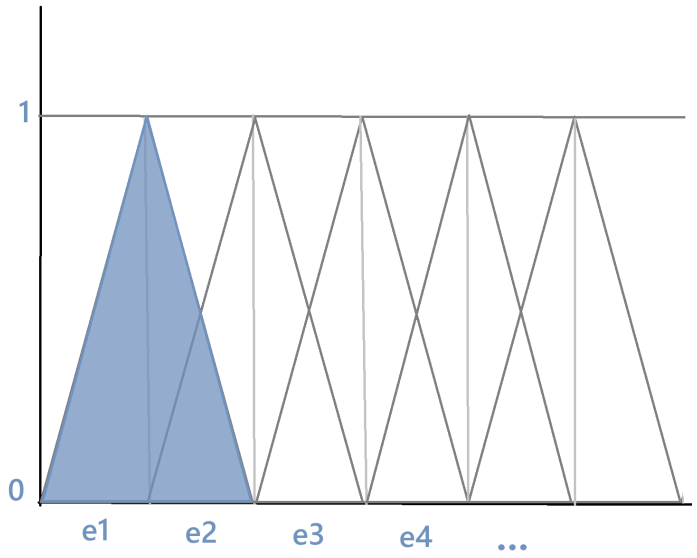

Рис. 8: Разбили U на отрезки e_i и на них, через треугольные функции принадлежности, постоим нечеткие множества F_i.

Лингвистическая переменная Можем перейти от задачи прогнозирования цены к задаче классификации: прогнозированию уровня цены (см. пример на рис.**??**), тренда или даже эмоционального настроя участников рынка. В этом случае через треугольные и трапециевидные функции принадлежности зададим по соответствующим лингвистическим переменным нечеткие множества.

2.2 Нечеткость и функция активации

Пусть $\phi_\theta(y)$ - это функция активации с параметрами θ, которые определяют форму функции активации (например, точка нуля, наклон). Наша задача – найти оптимальные параметры функции активации θ^*, минимизирующие ошибку прогноза финансового временного ряда:

$$\theta^* = \arg\min_\theta \mathcal{L}(X, y, \theta)$$

где $\mathcal{L}(X, y, \theta)$ – функция потерь, показывающая как хорошо нейросеть с параметрами θ (т.е. параметрами функции активации в том числе) прогнозирует y по входным данным X.

Как было отмечено выше, s-образные функции активации такие как *sigmoid* и ее вариации со смещенным центром нуля и измененным наклоном можно рассматривать как функции принадлежности к нечеткому множеству. А так как нейронная сеть это композиция нейронов производящих вычисления с помощью функций активации, можно сказать, что нейросеть производит операции над нечеткими множествами и является моделью нечеткого вывода.

2.3 Дефазификация

Приведем нечеткие результаты работы нейронной сети к четким прогнозам.

Гауссианы. Возьмем за прогноз y_i вершину спрогнозированного купола c

Probabilistic Fuzzy Sets Зная треугольные функции принадлежности на каждом интервале и, спрогнозировав распределение степеней принадлежности к множествам, можем однозначно определить четкий прогноз.

Лингвистическая переменная Этот случай аналогичен.

3 LSTM

3.1 Почему LSTM

Исследования в области прогнозирования временных рядов ведутся в течение многих лет, но с открытием применения глубокого обучения для решения данной задачи интерес к теме невероятно возрос. За последние десятилетия вышло огромное количество работ о прогнозировании финансовых временных рядов с использованием глубокого обучения (Deep Learning (DL)), рекуррентных нейронных сетей (Recurrent Neural Networks (RNN)), в частности, LSTM (Long Short-Term Memory) нейронных сетей с долгой краткосрочной памятью.

Минус RNN в невозможности смотреть далеко в прошлое, поэтому, работая в 1991 году над решением проблемы затухающего(исчезающего) градиента (the vanishing gradient problem) уже в 1997 году Зеппом Хохрайтером и Юргеном Шмидхубером была представлена новая архитектура рекуррентной нейронной сети - LSTM - которая смогла эффективно решать следующие задачи[37]:

1. Распознавание долгосрочных закономерностей в зашумленных входных последовательностях
2. Определение в зашумленных входных потоках порядка событий, находящихся во времени далеко друг от друга.
3. Извлечение информации, передаваемой расстоянием между событиями
4. Точная генерация периодических событий, закономерностей.
5. Надежное и длительное хранение действительных чисел

LSTM является самой цитируемой нейронной сетью 20 века [36], а современные алгоритмы LSTM разрабатываются и по сей день и используются для решения широкого спектра задач[37]: распознавание речи, машинный перевод, распознавание видео, распознавание рукописного ввода, *прогнозирование временных рядов*.

3.2 Архитектура

Любая рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети (рис. 9).

LSTM (Long short-term memory) – тип рекуррентной нейросети, модули, которой содержат четыре слоя, вместо одного (рис.10), что помогает эффективно работать с длинными временными периодами: распознавать долгосрочные закономерности, определять события, находящиеся во времени далеко друг от друга, извлекать информацию, передаваемую расстоянием между событиями. [37] Для описания ра-

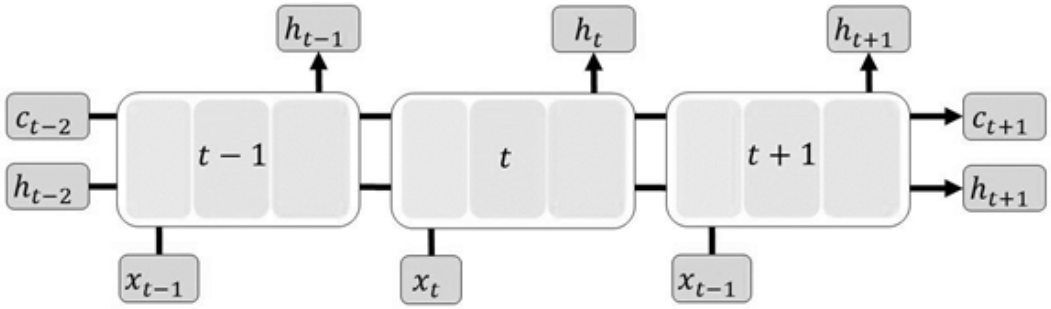

Рис. 9: Структура LSTM, развернутая во времени, напоминает цепочку

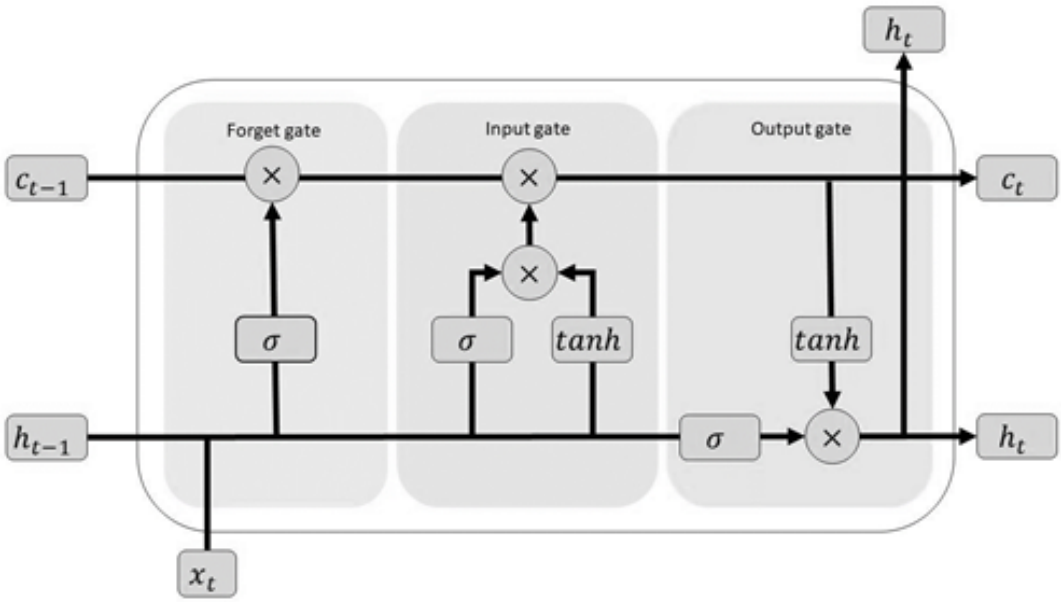

Рис. 10: Архитектура LSTM. Каждая линия переносит целый вектор от выхода одного узла ко входу другого. Сливающиеся линии означают объединение, а разветвляющиеся стрелки говорят о том, что данные копируются и копии уходят в разные компоненты сети. Обозначения: c_* —состояние ячейки, x_*—вход, h_*—выход , $\times$ — поточечное умножение, $+$— сложение векторов, σ — сигмоидальный слой, $tanh$ — tanh-слой.

боты архитектуры нейронной сети LSTM для прогнозирования введем несколько основных компонент и операций. Для удобства обозначим временной ряд цен как

$\{x^{(1)}, x^{(2)}, ..., x^{(T)}\}$, где $x^{(t)}$ - это цена актива в момент времени t. Для каждого момента времени t, входной вектор $x^{(t)}$ содержит информацию о цене.

Нейронная сеть LSTM состоит из нескольких ячеек LSTM[7], каждая из которых имеет вход $x^{(t)}$, предыдущее скрытое состояние $c^{(t-1)}$ и предыдущий выход $h^{(t-1)}$

Ключевой компонент LSTM – это состояние ячейки (cell state) c_*. LSTM может удалять информацию из состояния ячейки; этот процесс регулируется структурами, называемыми фильтрами (gates). В LSTM три таких фильтра, позволяющих определять какая информация должна быть сохранена или забыта. Фильтры состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения. Сигмоидальный слой возвращает числа от нуля до единицы, которые обозначают, какую долю каждого блока информации следует пропустить дальше по сети. Ноль в данном случае означает "не пропускать ничего", единица – "пропустить все". [39]

3.2.1 Слой фильтра забывания

Определим, какую информацию можно выбросить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый "слоем фильтра забывания" (forget gate layer). Он смотрит на h_{t-1} и x_t и возвращает число от 0 до 1 для каждого числа из состояния ячейки C_{t-1}. 1 означает "полностью сохранить", а 0 – "полностью выбросить".

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{1}$$

где W, b - матрица и вектор коэффициентов размерности $(n, m + d)$ и $(n, 1)$ соответственно, где n – количество нейронов в текущем слое, m - размер скрытого состояния (количество нейронов в предыдущем скрытом состоянии), d - размер входного вектора (количество признаков во входном векторе).

3.2.2 Слой входного фильтра

Теперь решим какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием "слой входного фильтра" (input layer gate) определяет, какие значения следует обновить. Затем tanh-слой строит вектор новых значений-кандидатов $\tilde{c}_t$, которые можно добавить в состояние ячейки.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{c}_t = tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{3}$$

3.2.3 Обновляем состояние ячейки

Заменим старое состояние ячейки c_{t-1} на новое состояние c_t. Для этого умножим старое состояние на f_t, забывая то, что мы решили забыть. Затем прибавим $i_t * \tilde{c}_t$. Это новые значения-кандидаты, умноженные на t – на сколько мы хотим обновить каждое из значений состояния.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{4}$$

3.2.4 Слой выходного фильтра

Решим, какую информацию мы хотим получать на выходе. Сначала мы применяем сигмоидальный слой, который решает, какую информацию из состояния ячейки мы будем выводить. Затем значения состояния ячейки проходят через tanh-слой, чтобы получить на выходе значения из диапазона от-1 до 1, и перемножаются с выходными значениями сигмоидального слоя, что позволяет выводить только требуемую информацию.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * tanh(c_t) \tag{6}$$

3.2.5 Общая формула

Итого, простейший LSTM-модуль можно представить в виде системы уравнений (1-6)

$$\begin{cases} f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \\ i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \\ \tilde{c}_t = tanh(W_c * [h_{t-1}, x_t] + b_c) \\ c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \\ o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \\ h_t = o_t * tanh(c_t) \end{cases} \tag{7}$$

4 Модель

4.1 Функция потерь

Функция потерь помогает оптимизировать параметры нейронных сетей. Наша цель - минимизировать потери для нейронной сети путем оптимизации ее параметров (весов). Потери рассчитываются с использованием функции потерь путем сопоставления целевого (фактического) значения и прогнозируемого значения нейронной сетью. Затем мы используем метод градиентного спуска для обновления весов нейронной сети таким образом, чтобы потери были сведены к минимуму. Так мы обучаем нейронную сеть.[15]

Нейронные сети пытаются минимизировать потери[19]

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(X_i, y_i; \theta) \to \min_{\theta}$$

где θ - параметры(веса) нейронной сети
$l(X_i, y_i; \theta)$ - функция, которая показывает как хорошо нейросеть с параметрами θ предсказывает y_i для X_i
N - число элементов в датасете

Функция потерь является мерой расхождения между истинным значением оцениваемого параметра (y) и оценкой параметра$(\hat{y})$.

Основной результат, который даёт функция потерь — оценка того, насколько хорошо классификатор работает на обучающей выборке. Но мы помним, что главная цель машинного обучения — заставить алгоритм правильно работать на тестовой выборке. Поэтому хорошо справляющийся с обучающими данных метод может совершенно не работать на новых объектах. Это явление называется «переобучение».[14]

В обучении с учителем (supervised learning) есть 2 основных типа функций потерь, которые соответствуют 2 основным типам нейронных сетей: регрессионные и классификационные функции потерь[16]

1. Регрессионная функция потерь: Mean Squared Error, Mean Absolute Error.

2. Классификационная функция потерь: Binary Cross-Entropy, Categorical Cross-Entropy

4.1.1 Кросс-энтропия

Кросс-энтропия (или логарифмическая функция потерь – log loss): Кросс-энтропия измеряет расхождение между двумя вероятностными распределениями. Если кросс-энтропия велика, это означает, что разница между двумя распределениями велика, а если кросс-энтропия мала, то распределения похожи друг на друга.[17]

Кросс-энтропия определяется как:

$$L(P, Q) = -\sum_{x} P(x) \log(Q(x))$$

где P – распределение истинных ответов, а Q – распределение вероятностей прогнозов модели.

Давайте упростим это для нашей модели:

1. N – количество наблюдений
2. K – количество возможных меток класса (в нашем случае - разновидности тренда: K = 2 или 3)
3. I – двоичный индикатор (0 или 1) того, является ли метка класса C правильной классификацией для наблюдения y
4. p – прогнозируемая вероятность модели

4.1.2 Бинарная классификация

В случае бинарной классификации (K=2), формула бинарной кросс-энтропии(Binary Cross-Entropy или BCE) имеет вид:

$$BCE = -(I \log(p) + (1 - I) \log(1 - p))$$

При двоичной классификации каждая предсказанная вероятность сравнивается с фактическим значением класса (0 или 1), и вычисляется оценка, которая штрафует вероятность на основе расстояния от ожидаемого значения.[17]

Если мы используем BCE, нужен один выходной узел, чтобы классифицировать данные на два класса. Выходное значение должно быть передано через сигмоидальную функцию активации(28) и диапазон выхода $[0, 1]$.[14]

4.1.3 Мульти-классовая классификация

В случае мульти-классовой классификации (K>2) мы берем сумму значений логарифмических функций потерь для каждого прогноза наблюдаемых классов. Получаем формулу категориальной кросс-энтропии (Categorical Cross-Entropy или CCE)[17]

$$CCE = -\sum_{c=1}^{K}(I_{y,c}\log(p_{y,c}))$$

Если мы используем CCE, должно быть столько же выходных узлов, сколько классов. И окончательный выходной слой должен быть пропущен через SoftMax(27) функцию активации, чтобы каждый узел выводил значение вероятности в диапазоне $[0, 1]$.[14]

4.2 Метод оптимизации

Мы говорили, что функция потерь показывает, насколько хорошо или плохо выбранные веса справляются с поставленной задачей. В двумерном пространстве значения функции можно представить в виде цветного «ландшафта», где наиболее холодный (синий) — наименьшая величина потерь, к которой мы стремимся; а тёплый регион (красный) — наибольшая. На рисунке 11 черная линия - траектория движения алгоритма оптимизации, которая стремится к минимуму функций потерь.
приведем ниже самые популярные алгоритмы оптимизации.

4.2.1 Stochastic Gradient Descent (SGD)

SGD или Стохастический градиентный спуск обновляет параметры модели (θ) в отрицательном направлении градиента g взяв подмножество или мини-пакет (mini-batch) данных размером (m):

$$g = \frac{1}{m}\Delta_\theta \sum_i L(f(X_i;\theta), y_i) \tag{8}$$

$$\theta = \theta - (\epsilon_k \times g) \tag{9}$$

Нейронная сеть представлена $f(X_i;\theta)$, градиент функции потерь L вычисляется относительно параметров модели θ

Скорость обучения (Learing rate) ϵ_k определяет размер шага, который алгоритм выполняет вдоль градиента (в отрицательном направлении в случае минимизации и в положительном направлении в случае максимизации). [20]

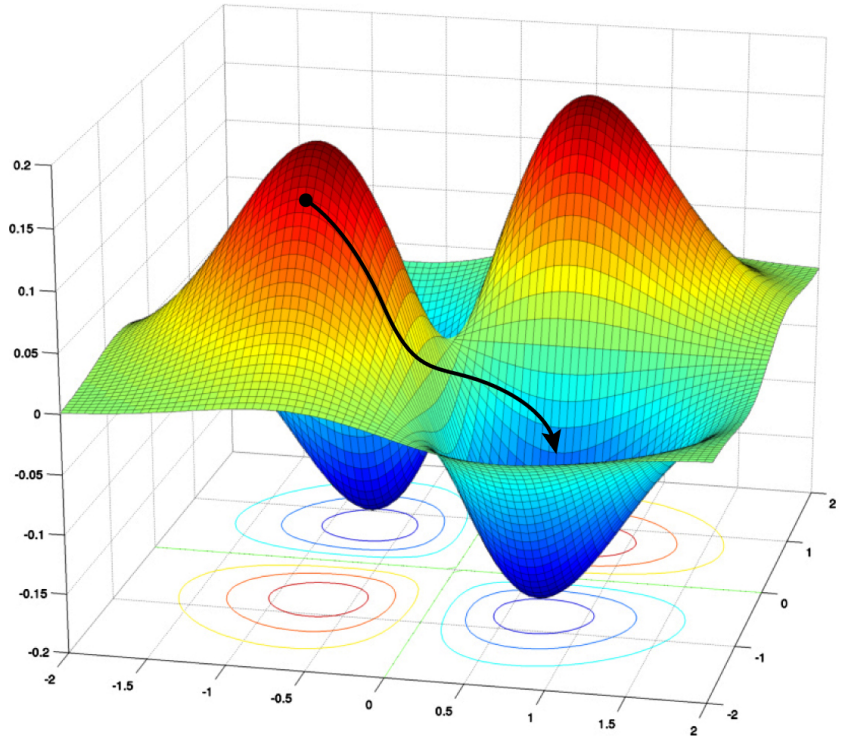

Рис. 11: Loss landscape или Ландшафт функции потерь для 2х весов

ϵ_k является функцией k-ой итерации алгоритма оптимизации и является чрезвычайно важным гиперпараметром. Слишком высокая скорость обучения (например, $> 0,1$) может привести к обновлениям параметров, которые пропускают оптимальное решение, а слишком низкая скорость обучения (например, $< 1e-5$) приведет к длительному времени обучения или же к сходимости в ближайшем локальном минимуме, который вряд ли окажется искомым ответом. [23]

Мы хотим, чтобы ϵ_k удовлетворяло условиям Роббинса-Монро (10). Первое равенство гарантирует, что алгоритм найдет оптимальное решение внезависимости от начальной точки, а второе неравенство регулирует колебания:

$$\sum_k \epsilon_k = \infty \qquad \sum_k \epsilon_k^2 < \infty \tag{10}$$

4.2.2 Momentum

Momentum(Импульс) - Импульс накапливает экспоненциально затухающую скользящую среднюю прошлых градиентов и продолжает двигаться в их направлении. Идея методов с накоплением импульса проста: «Если мы некоторое время движемся в определённом направлении, то, вероятно, нам следует туда двигаться

некоторое время и в будущем»[21]

$$v = \alpha v - \epsilon \Delta_\theta (\frac{1}{m} \sum_i L(f(X_i; \theta), y_i)) \tag{11}$$

$$\theta = \theta + v \tag{12}$$

Широко используемые значения параметра α: 0.5 и 0.9

4.2.3 Nesterov's accelerated gradient method

$$v = \alpha v - \epsilon \Delta_\theta (\frac{1}{m} \sum_i L(f(X_i; \theta + (\alpha \times v), y_i)) \tag{13}$$

$$\theta = \theta + v \tag{14}$$

Разница между Нестеровым и стандартным импульсом заключается в том, когда оценивается градиент. С импульсом Нестерова градиент оценивается после применения текущей скорости, поэтому в импульс Нестерова добавляется поправочный коэффициент к градиенту.

4.2.4 AdaGrad

$$g = \frac{1}{m} \Delta_\theta \sum_i L(f(X_i; \theta), y_i) \tag{15}$$

$$s = s + g^T g \tag{16}$$

$$\theta = \theta - (\epsilon_k \times \frac{g}{\sqrt{s + eps}}) \tag{17}$$

AdaGrad(adaptive gradient) решает проблему того, что скорость обучения не зависит от градиента. То есть мы можем застревать в каких-то местах, например, когда градиент почти не меняется (рисунок 12, левый график) или проскакивать слишком быстро, колеблясь, но не достигая оптимального решения, когда градиент большой (рисунок 12, справа).

4.2.5 RMSProp

RMSProp(Root Mean Square Propagation или же Среднеквадратичное распространение корня) - модифицирует AdaGrad, изменяя накопление градиента на экспонен-

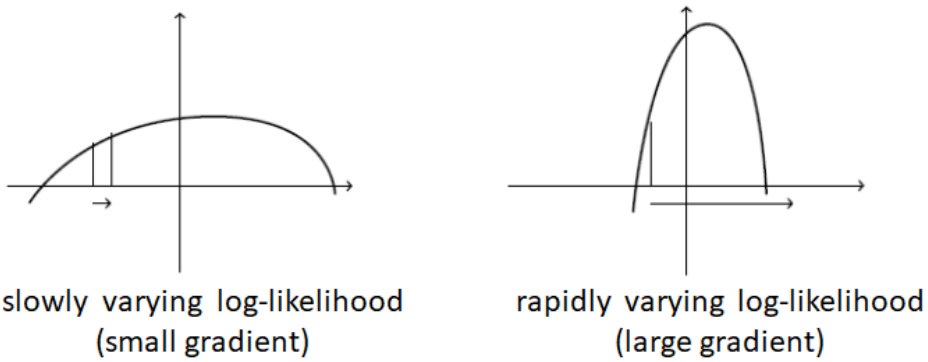

Рис. 12: примеры графиков с маленьким(слева) и большим(справа) градиентом

циально взвешенную скользящую среднюю, то есть отбрасывает историю из отдаленного прошлого. Из-за этого RMSProp является эффективным алгоритмом оптимизации и часто используется в глубоком обучении.

$$g = \frac{1}{m}\Delta_\theta \sum_i L(f(X_i;\theta), y_i) \tag{18}$$

$$s = decay\_rate \times s + (1 - decay\_rate)g^T g \tag{19}$$

$$\theta = \theta - (\epsilon_k \times \frac{g}{\sqrt{s + eps}}) \tag{20}$$

4.2.6 Adam

Adam(adaptive moments) - один из самых популярных оптимизаторов благодаря своей скорости сходимости. Является вариантом комбинации RMSProp (4.2.5) и Momentum (4.2.2). Итерации выглядят как RMSProp, за исключением того, что вместо необработанного стохастического градиента используется плавная версия градиента, также полное обновление Адама включает механизм коррекции смещения.

$$g = \frac{1}{m}\Delta_\theta \sum_i L(f(X_i;\theta), y_i) \tag{21}$$

$$m = \beta_1 m + (1 - \beta_1)g \tag{22}$$

$$s = \beta_2 s + (1 - \beta_2)g^T g \tag{23}$$

$$\theta = \theta - (\epsilon_k \times \frac{g}{\sqrt{s + eps}}) \tag{24}$$

Рекомендованные значения параметров [22]

$$\beta_1 = 0.9 \qquad \beta_2 = 0.999 \qquad eps = 1e - 8 \tag{25}$$

Вывод

Во многих случаях градиентный спуск может привести к оверфиттингу (переобучению), упасть в слишком глубокий минимум функции ошибки, который плохо будет обобщаться на новые данные. Собственно, алгоритмам градиентного спуска неоткуда знать о том, насколько хорошо обобщается то, что они делают, они просто оптимизируют ошибку на тренировочном множестве. Поэтому мы сами должны следить за качеством обобщения, которое в простейшем случае соответствует ошибке на валидационном множестве.

Когда данные достаточно разрежены, то однозначно стоит сделать выбор в сторону адаптивных моментов. Так как Adam является улучшенной версией AdaGrad, RMSProp, то в силу скорости сходимости стоит выбрать его. Но существуют задачи, когда Adam не справляется, тогда можно обратиться и к другим алгоритмам. [23]

Мой выбор пал на Adam и SGD - наиболее известные и используемые методы оптимизации.

4.3 Оценка моделей

Метрика оценки качества - третий важный элемент в построении нейрорсети. С помощью нее можем оценить на тестовом множестве качество предсказаний обученной нейросети, отследить переобучение. Метрики, как и функции потерь, для задач классификации и регрессии различаются. Рассмотрим метрики оценки качества для задач классификации.

Перед переходом к самим метрикам необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — confusion matrix (матрица ошибок). Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Таблица 1: матрица ошибок(confusion matrix) для двух классов

Поясним обозначения в таблице.

True Positive (TP):

1. Прогнозируемое значение равно фактическому значению ($\hat{y} = y$).

2. Фактическое значение было положительным ($y = 1$), и модель предсказала положительное значение ($\hat{y} = 1$).

True Negative (TN):

1. Прогнозируемое значение равно фактическому значению ($\hat{y} = y$).

2. Фактическое значение было отрицательным ($y = 0$), и модель предсказала отрицательное значение ($\hat{y} = 0$).

False Negative (FN) - ошибка 1-ого рода:

1. Прогнозируемое значение было предсказано неверно ($\hat{y} \neq y$).

2. Фактическое значение было положительным ($y = 1$), но модель предсказала отрицательное значение ($\hat{y} = 0$).

False Positive (FP) - ошибка 2-ого рода:

1. Прогнозируемое значение было предсказано неверно ($\hat{y} \neq y$).

2. Фактическое значение было отрицательным ($y = 0$), но модель предсказала положительное значение ($\hat{y} = 1$).

Случай нескольких классов отображен на рисунке 13.

4.3.1 Accuracy

Accuracy (точность) - одна из самых простых метрик - доля правильных ответов алгоритма[25]:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{26}$$

В задачах с неравными классами accuracy бесполезна, так как не выполняет главную задачу метрики - адекватную оценку точности работы нейросети. Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых наш классификатор определил верно ($TN = 90, FP = 10$), и 10 спам-писем, 5 из которых классификатор также определил верно ($TP = 5, FN = 5$). Тогда accuracy:

$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4\%$$

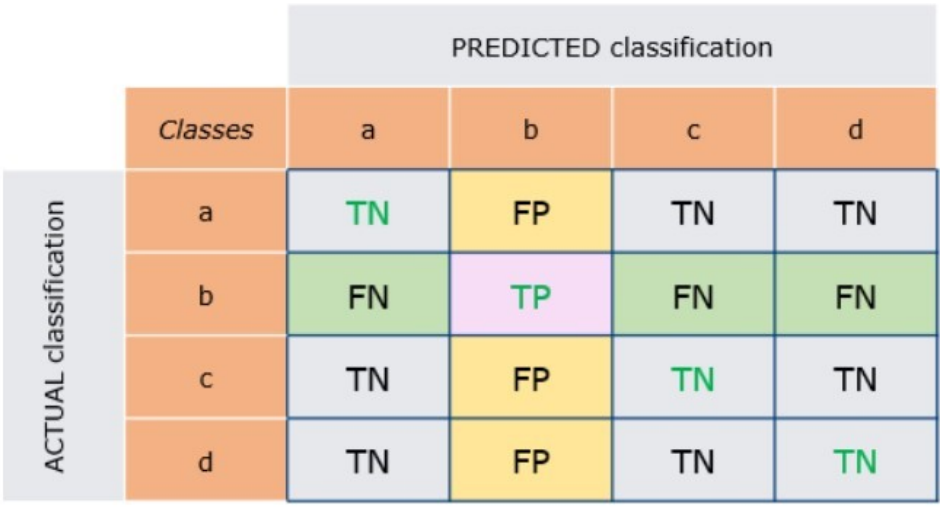

Рис. 13: confusion matrix для мультиклассовой классификации относительно класса b.

Однако если мы просто будем предсказывать все письма как не-спам, то получим более высокую accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9\%$$

Для случая мультиклассовой классификации рассматривается accuracy относительно 1 из классов. [24]

4.3.2 Precision

Precision (точность) можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными. То есть мы можем оценить возможность алгоритма отличать данный класс от других.

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

Именно введение precision не позволяет нам записывать все объекты в один класс, как в случае с accuracy, так как в мы получаем рост уровня False Positive.

4.3.3 Recall

Recall (полнота) показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. Recall демонстрирует способность

алгоритма обнаруживать данный класс.

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

Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок.

4.3.4 F-measure

Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-мера (в общем случае F_β) — среднее гармоническое precision и recall :

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

β в данном случае определяет вес точности в метрике. Часто используется вариант

$$\beta = 1 \qquad\qquad F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

В случае когда количество классов $K > 2$ есть 2 разновидности F-меры. Макро F-мера $\in [0; 1]$. В ней нет связи с размером классов, то есть большие и маленькие равнозначны. Чем больше Macro F1 мера, тем лучше алгоритм предсказывает все классы.[24]

$$MacroF_1 = \frac{\sum_{k=1}^{K} F_{1_k}}{K}$$
$$MicroF_1 = \frac{\sum_{k=1}^{K} TP_k}{N}$$

индекс k обозначает класс относительно которого считается метрика,
N - общее число наблюдений.

Заметим, что Micro F1 мера равна accuracy, то есть преимущества и недостатки у нее такие же.

4.4 Функции активации

Ниже приведены классические функции активации:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} x_j} \in [0; 1] \tag{27}$$

$$sigmoid = \frac{1}{1 + e^{-x}} \in [0; 1] \tag{28}$$

$$ReLU = \max(0; x) \in [0; +\infty] \tag{29}$$

$$LeakyReLU = \max(\alpha x; x) \in [-\infty; +\infty] \tag{30}$$

$$tanh = \max(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}) \in [-1; 1] \tag{31}$$

Вывод

В этой части приведен обзор на основные моменты, которые используются при создании нейросети: функции потерь, алгоритмы оптимизации, метрики оценки качества и функции активации. Выбор каждого из методов зависит от задачи, типа данных и отображается на правильности, скорости работы алгоритма.

В случае прогнозирования финансовых временных рядов стоит выбрать:

1. функция потерь: MAE
2. алгоритм оптимизации: Adam4.2.6
3. метрика: MAE, RMSE, MAPE

5 Программное обеспечение

Для реализации вычислительного эксперимента выбран язык Python [27] так как у него один из самых обширных набор библиотек, которые упрощают работу с данными и позволяют выполнять сложные операции. Все вычисления проводились в Google Colab[28].

Используемые библиотеки:

1. TensorFlow[33] - фреймворк, разработанный компанией Google. TensorFlow позволяет создавать сложные модели, которые могут быть легко масштабированы для работы на нескольких устройствах. Он также предоставляет множество инструментов для оптимизации и отладки моделей.

2. Scikit-learn [32] - библиотека машинного обучения, которая использовалась для оценки моделей. Так же она предоставляет множество функций для работы с данными, обучения моделей и оценки их качества.

3. Matplotlib [34] - библиотека для визуализации данных. Использована для отображения исторических данных о цене и результатов прогнозирования, а также для анализа их свойств.

4. yfinance [29] - это библиотека для загрузки и работы с финансовыми данными с помощью Yahoo Finance API (Application Programming Interface). Библиотека позволяет загружать исторические данные о ценах акций, индексов, валют, а также другую информацию, такую как объемы торгов и дивиденды.

5. datetime [**?**] - это библиотека для работы с датами и временем в Python.

6. numpy [30] и pandas [31] - библиотеки для работы с данными.

6 Вычислительный эксперимент

6.1 Задачи для вычислительного эксперимента

В рамках вычислительного эксперимента необходимо решить следующие задачи:

1. Исследовать влияние фаззификации входных данных на точность прогнозирования временного ряда.
2. Исследовать влияние параметрических функций активации внутри LSTM на качество прогнозирования временного ряда.
3. Провести сравнение результатов предложенной модели с традиционной моделью LSTM.

6.2 Описание вычислительного эксперимента

6.2.1 Сбор данных

Данные для экспериментов были получены с финансовой платформы Yahoo Finance. Для анализа и прогнозирования использовались данные котировок криптовалюты BTC-USD (биткоин к доллару США) с разными интервалами: 1 минута, 1 час и 1 день. Период, охваченный экспериментами:

- 1-минутный интервал: последняя неделя.
- 1-часовой интервал: с 01.01.2023 по 15.02.2024.
- 1-дневный интервал: с 01.01.2018 по 15.02.2024.

Полученные данные были предварительно обработаны с помощью нормализации (и фаззификации), а затем разделены на тренировочные (75%) и тестовые (25%) выборки.

Диапазон значений исходного временного ряда составляет от $81,000$ до $88,000$ долларов. Это важно учитывать при интерпретации метрик качества прогноза.

6.2.2 Проведение эксперимента

Все модели запускались, в зависимости от тика, на следующих параметрах:

Тик	units	lookback
1m	16	30
1h		
1d		

Для обучения возьмем число эпох = 100, размер батча = 32.

Сравним результаты работы следующих архитектур:

LSTM – базовая модель LSTM без применения фаззификации и параметрической активации. Размерность входных данных: $(lookback, 1)$. Состоит из слоя $LSTM(units)$ и полносвязного слоя $Dense(1)$.

Fuzzy Gauss – LSTM с добавленной фаззификацией данных методом гауссиан. Локальную волатильность ("ширину" купола) будем считать по $window = 10$ последним измерениям. Размерность входных данных: $(lookback, 2)$. Модель состоит из слоя $LSTM(units)$ и полносвязного слоя $Dense(2)$.

PFS – LSTM с добавленной фаззификацией данных с помощью Probabilistic Fuzzy Sets. Число интервалов $m = 14$, как в оригинальной статье. Размерность входных данных: $(lookback, m)$. Состоит из слоя $LSTM(units)$ и полносвязного слоя $Dense(m)$.

Fuzzy Activation – LSTM с параметрическими функциями активации. Есть несколько вариантов:

1. Каждая из трех сигмоид в LSTM имеют одинаковую параметризацию. (+2 параметра)
2. Все три сигмоиды в LSTM имеют разные параметры. (+6 параметров)
3. Параметризуем все 5 функций активации в LSTM. Заменяем $tanh$ на параметрическую сигмоиду.(+10 параметров)

Каждое увеличение количества параметров модели увеличивает время обучения. Размерность входных данных: $(lookback, 1)$. Состоит из слоя $LSTM(units)$ и полносвязного слоя $Dense(1)$.

Fuzzy LSTM – итоговая модель с введением нечеткости на этапе предобработки данных и внутри модели. Для эксперимента выбрана комбинация лучших по качеству прогноза версий моделей выше: Fuzzy Gauss + Fuzzy Activation 1 с одной параметрической сигмоидой.

6.3 Результаты вычислительного эксперимента

Результаты эксперимента на минутных тиках представлены в таблице 6.3. Качество прогнозирования оценивалось на основе метрик RMSE, MAE, MAPE.

Модель	Тренировочная			Тестовая		
	RMSE ($)	MAE ($)	MAPE (%)	RMSE ($)	MAE ($)	MAPE (%)
LSTM	59.0921	46.9797	0.05465	60.6736	43.1379	0.05224
Fuzzy Gauss	31.6476	22.0033	0.02542	38.1486	23.6430	0.02817
PFS	42.2479	28.1578	0.03246	49.3677	35.7689	0.04172
Fuzzy Activation 1	42.1211	29.6191	0.03453	52.5275	35.3794	0.04288
Fuzzy LSTM	**31.6104**	**16.6874**	**0.01943**	**36.7655**	**18.0011**	**0.02178**

Таблица 2: Сравнение результатов прогнозирования моделей на минутных тиках. Лучшие результаты выделены жирным шрифтом.

Видим, что все предложенные методы фаззификации по отдельности улучшают прогноз. Фаззификация входных данных через гауссианы повышает точность прогноза в 2 раза. А предложенная модель Fuzzy LSTM улучшила прогноз в 2.5 раза. Рисунок 14 показывает прогнозы модели на основе нечеткой логики на тестовой выборке в сравнении с реальными значениями.

Рис. 14: Сравнение прогноза предложенной модели на основе нечеткой логики и тестовой выборки.

Из-за того, что в моделях увеличилось количество параметров, время обучения нейросети, соответственно, тоже возросло:

Модель	Число дополнительных параметров	Время обучения (сек)		
		1m	1h	1d
LSTM	0	249.614		
Fuzzy Gauss	$1 \cdot lookback$	249.472		
PFS	$(m-1) \cdot lookback$	262.7114		
Fuzzy Activation 1	2	420.686		
Fuzzy LSTM	$2 + 1 \cdot lookback$	452.313		

Таблица 3: Сравнение времени обучения моделей.

По графикам значений loss-функции от эпохи (Рис. 15), видно, что все модели с первых же эпох достигают асимптоты, поэтому число эпох можно уменьшить до $10 - 20$, что уменьшит время обучения.

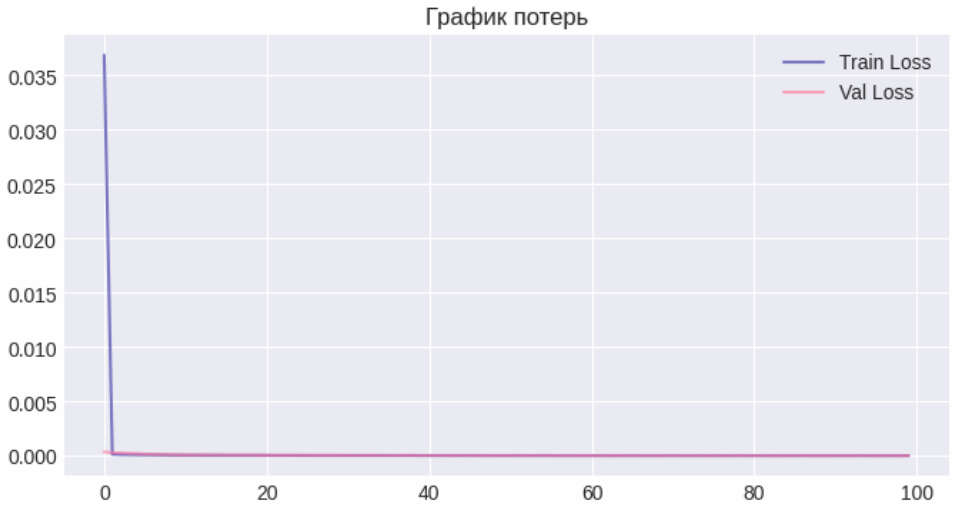

Рис. 15: График потерь для Fuzzy LSTM. Остальные модели ведут себя похожим образом.

6.4 Выводы

Предложенная архитектура LSTM на основе нечеткой логики в 2.5 раза увеличивает точность прогноза финансового временного ряда на минутных тиках.

7 Заключение

В данной работе был разработан вариант архитектуры нейронной сети с памятью на основе нечеткой логики для прогнозирования временных рядов. Введение нечеткости на этапе предобработки данных и внутри модели увеличило точность прогноза в 2.5 раза по сравнению с базовой LSTM. Такой подход позволил встроить нечеткую логику в процесс обучения модели и повысить интерпретируемость логики работы нейросети.

Полученные результаты открывают новые возможности для применения нечеткой логики и глубокого обучения в задачах прогнозирования финансовых временных рядов.

Перспективными направлениями дальнейших исследований является:

- Применение альтернативных параметрических функций активации, включая параметризацию $tanh$;
- Использование лингвистических переменных для классификации состояний рынка и тренда;
- Внедрение методов обучения с подкреплением (reinforcement learning) для выработки динамических стратегий прогноза или управления портфелем на основе предсказанных значений.

Список литературы

[1] Ширяев А.Н. Основы стохастической финансовой математики: В 2 т. Т.1: Факты,модели. -М. : МЦНМО, 2016.

[2] Andersen T. G. et al. Handbook of Financial Time Series 2009 Springer Berlin, Heidelberg

[3] Афанасьева Т. В. Прогнозирование временных рядов: нечеткие модели /Т. В. Афанасьева, А. М. Наместников,И. Г. Перфильева, А. А. Романов, Н. Г. Ярушкина; под науч. ред. Н.Г. Ярушкиной. – Ульяновск : УлГТУ, 2014. – 145 с.

[4] Kamlesh Bisht, Sanjay Kumar, Fuzzy time series forecasting method based on hesitant fuzzy sets, Expert Systems with Applications,Volume 64, 2016

[5] Mandal J. K., Bhattacharyya D., Auluck N. Advanced Computing and Communication Technologies //Resource. – 2022.

[6] Кумсков М.И, Арсланова А.Р., Махова А.Г Исследование функций активации и гиперпараметров в архитектуре сетей долгой краткосрочной памяти для прогнозирования финансовых временных рядов

[7] Bingham G., Miikkulainen R. Discovering parametric activation functions //Neural Networks. – 2022. – Т. 148. – С. 48-65.

[8] Beke A., Kumbasar T. Learning with type-2 fuzzy activation functions to improve the performance of deep neural networks //Engineering applications of artificial intelligence. – 2019. – Т. 85. – С. 372-384.

[9] Max Rokatansky Анализ временных рядов https://habr.com/ru/companies/otus/articles/732080/ (20.05.2023)

[10] Brian Christopher Time Series Analysis (TSA) in Python - Linear Models to GARCH https://www.blackarbs.com/blog/time-series-analysis-in-python-linear-models-to-garch/11/1/2016#AR (20.05.2023)

[11] Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019 https://arxiv.org/pdf/1911.13288.pdf (27.11.2022)

[12] Ф.М. Гафаров, Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018.

[13] Н. Паклин Нечеткая логика — математические основы https://loginom.ru/blog/fuzzy-logic

[14] Fei-Fei Li, Justin Johnson, Serena Yeung Loss Functions and Optimization (перевод: https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-3-funkciya-poter-i-optimizaciya/?utm_source=yandex.ru&utm_medium=organic&utm_campaign=yandex.ru&utm_referrer=yandex.ru) (14.05.2023)

[15] Shiva Verma Understanding different Loss Functions for Neural Networks https://shiva-verma.medium.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718 (14.05.2023)

[16] Vishal Yathish Loss Functions and Their Use In Neural Networks https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9 (14.05.2023)

[17] Javaid Nabi Estimators, Loss Functions, Optimizers —Core of ML Algorithms (перевод: https://id-lab.ru/posts/developers/funkcii/) (14.05.2023)

[18] Fei-Fei Li, Justin Johnson, Serena Yeung Training Neural Networks II https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-7-obuchenie-nejrosetej-chast-2/?utm_source=yandex.ru&utm_medium=organic&utm_campaign=yandex.ru&utm_referrer=yandex.ru (14.05.2023)

[19] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein Visualizing the Loss Landscape of Neural Nets 2018 https://arxiv.org/abs/1712.09913 (14.05.2023)

[20] Vadim Smolyakov Neural Network Optimization Algorithms https://towardsdatascience.com/neural-network-optimization-algorithms-1a44c282f61d (14.05.2023)

[21] Павел Садовников Методы оптимизации нейронных сетей https://habr.com/ru/articles/318970/ (14.05.2023)

[22] Kingma D., Ba J. Adam: A Method for Stochastic Optimization // arXiv, 2014. http://arxiv.org/abs/1412.6980 (14.05.2023)

[23] Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»).

[24] Margherita Grandini, Enrico Bagli, Giorgio Visani Metrics for Multi-Class Classification: an Overview, arXiv, 2020 https://arxiv.org/abs/2008.05756 (14.05.2023)

[25] Egor Labintcev Метрики в задачах машинного обучения https://habr.com/ru/companies/ods/articles/328372/ (14.05.2023)

[26] Ran Aroussi Welcome to the yfinance wiki https://github.com/ranaroussi/yfinance/wiki (20.05.2023)

[27] Python https://www.python.org/ (20.05.2023)

[28] Google colab https://colab.research.google.com/ (20.05.2023)

[29] yfinance https://pypi.org/project/yfinance/ (20.05.2023)

[30] NumPy https://numpy.org/ (20.05.2023)

[31] Pandas https://pandas.pydata.org/ (20.05.2023)

[32] sklearn.preprocessing.minmax_scale https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html (20.05.2023)

[33] TensorFlow https://www.tensorflow.org/?hl=ru (20.05.2023)

[34] Matplotlib: Visualization with Python https://matplotlib.org/ (20.05.2023)

[35] Anas Ait Aomar Over-smoothing issue in graph neural network https://towardsdatascience.com/over-smoothing-issue-in-graph-neural-network-bddc8fbc2472 04.03.2024

[36] The most cited neural networks all build on work done in my labs.Jürgen Schmidhuber (2021, slightly updated 2022) https://people.idsia.ch/~juergen/most-cited-neural-nets.html (27.11.2022)

[37] Jürgen Schmidhuber's page on Recurrent Neural Networks (updated 2017) https://people.idsia.ch/~juergen/rnn.html (27.11.2022)

[38] Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019 https://arxiv.org/pdf/1911.13288.pdf (27.11.2022)

[39] LSTM – сети долгой краткосрочной памяти https://habr.com/ru/companies/wunderfund/articles/331310/ (перевод Christopher Olah Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs/) (14.05.2023)

Программная реализация:

Базовая LSTM https://colab.research.google.com/drive/1_yJcy-8DzC1IZmTTfm05F7o68wEC2wwh?usp=sharing

Нечеткая предобработка данных https://colab.research.google.com/drive/1VzeMbfygAKDebMshnmGNSqEKLzSt0CSf?usp=sharing

LSTM с параметрическими функциями активации и финальная модель FuzzyLSTM https://colab.research.google.com/drive/1H8pq2R5ZYYMPnWXRyZ7Y33UmS928aU6V?usp=sharing