Исследование функций активации и гиперпараметров в архитектуре сетей долгой краткосрочной памяти для прогнозирования финансовых временных рядов

Кумсков Михаил Иванович (Kumskov Mikhail Ivanovich)[a, *], Арсланова Алина Рустямовна (Arslanova Alina Rustiamovna)[a,], Махова Анастасия Геннадьевна (Makhova Anastasiia Gennadevna)[a,***]**

[a]Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный университет имени М. В. Ломоносова», Россия, Москва, тер. Ленинские Горы, д. 1, 119991

*e-mail: mikhail.kumskov@math.msu.ru
**e-mail: alina.arslanova@math.msu.ru
***e-mail: anastasiia.makhova@math.msu.ru

Аннотация В данной работе рассматриваются варианты архитектуры LSTM сетей в задаче прогнозирования финансовых временных рядов на примере цен BTC-USD и ETH-USD на минутных, часовых и дневных тиках. Целью работы является исследование влияния функции активации и гиперпараметров LSTM-сети на качество прогноза. Рассмотрены как классические функции активации, такие как сигмоидальная, гиперболический тангенс, линейная, ReLU, так и неклассические функции активации cloglog, loglog. Гиперпараметры задают архитектурные решения нейросети и влияют на качество прогноза и скорость обучения.

В работе показано, что правильный выбор функции активации и гиперпараметров сетей долгой краткосрочной памяти, является важным фактором, влияющим на качество прогноза и что неклассические функции активации cloglog и loglog, стоит использовать в архитектурах сетей долгой краткосрочной памяти для задач прогнозирования временных рядов наравне с классическими функциями активации.

Ключевые слова: рекуррентные нейронные сети, RNN, Long short-term memory, LSTM, функция активации, units, lookback, TensorFlow, гиперпараметры, финансовые временные ряды.

ВВЕДЕНИЕ

Прогнозирование цен уже давно является центральным направлением деятельности в области математических финансов и эконометрики. Однако финансовые временные ряды, как известно, сложны для анализа из-за их нестационарности, нелинейности и шума, возникающего в том числе из-за иррационального человеческого поведения на финансовых рынках. Обычно для задачи прогнозирования использовались методы, основанные на авторегрессии скользящего среднего (ARIMA), GARCH-модели, а также другие модели стохастической волатильности [1]. Использование этих моделей предполагает выдвижение гипотез о данных, их базовом распределении и различных процессов, влияющих на него, и по итогу часто методы плохо обобщаются для новых данных, хотя и дают ценную информацию о временных рядах.

В последнее время разработки в области машинного обучения и нейронных сетей привели к появлению нелинейных моделей временных рядов, которые все чаще адаптируются для финансовых задач. Например, машины опорных векторов (SVM), ограниченные машины Больцмана (RBM), случайные леса и деревья с градиентным бустингом (GBM) — это лишь некоторые из них моделей в финансовых задачах [3].

Нейронные сети обширно используются в прогнозировании временных рядов, таких как погода, потребление энергии, финансовые ряды. Временные ряды в финансах отличаются следующими особенностями: данные неструктурированного характера, высокий уровень неопределенности, скрытые отношения между признаками [3].

В этой работе была взята за основу архитектура нейронных сетей долгой краткосрочной памяти - long short-term memory networks, LSTM [2]. Главное отличие архитектуры сетей долгой краткосрочной памяти - они позволяют использовать информацию не только от текущих входных данных, но и от предыдущих входных данных [2]. Также одно из преимуществ нейронных сетей перед статистическими методами - они могут учитывать внезапные изменения [3].

В целом работа нейронной сети зависит от различных факторов, таких как количество скрытых слоев, алгоритм обучения, функция активации каждого слоя. Однако основной упор в исследованиях нейронных сетей делается на алгоритмы и архитектуры обучения, игнорируя важность

функций активации. С другой стороны, выбор функций активации может сильно влиять на сложность и производительность нейронных сетей и, по итогу играет важную роль в сходимости алгоритмов обучения. В этой работе рассматривается шесть функций активации с 1 и 3 скрытыми слоями.

Гиперпараметры задают архитектурные решения нейросети и влияют на качество прогноза и скорость обучения. В работе рассмотрены гиперпараметры units и lookback. Units — это количество нейронов в слое сети. Каждая ячейка LSTM формирует вектор скрытого состояния, длина которого и есть units. Lookback — параметр, определяющий количество предыдущих временных шагов, которые используются в качестве входных переменных для прогнозирования следующего периода.

В результате вычислительных экспериментов было выявлено, что представленные неклассические функции $cloglog$ и $loglog$ демонстрируют свою эффективность при использовании в LSTM архитектуре нейронной сети с настроенными гиперпараметрами $units$ и $lookback$ для всех видов тиков: для прогноза цены BTC-USD у $cloglog$ лучшие результаты по метрикам RMSE, MAE, MAPE, а функция активации loglog сопоставима по работе с sigmoid функцией, для ряда ETH-USD прогноз с функцией $cloglog$ сопоставим с результатами функции активации tanh. Кроме того, эмпирическим путем для каждого вида тика определены функции активации и значения $units$ и $lookback$, дающие наилучший прогноз.

В главе 1 дана постановка задачи, описаны функции активации, использующиеся в вычислительном эксперименте. В главе 2 описана схема архитектуры сети долгой краткосрочной памяти. В главе 3 приведены основные компоненты нейронной сети: функция потерь, алгоритм оптимизации и метрики качества. В главе 4 сделан обзор на программное обеспечение и использующиеся библиотеки. В главе 5 обсуждается проведенный вычислительный эксперимент и его результаты. В главе 6 приведен анализ переменных units и lookback. Глава 7 – заключение.

1. ПОСТАНОВКА ЗАДАЧИ

1.1 Существующие модели по прогнозированию финансовых временных рядов

На данный момент существует множество работ, посвященных прогнозированию цен на финансовые показатели с использованием методов машинного обучения и анализа данных.

Некоторые работы используют только одну модель, например, модели ARIMA (Autoregressive Integrated Moving Average), LSTM, GRU (Gated Recurrent Units) или Random Forest [1]. Некоторые работы используют ансамбли моделей, например, комбинацию LSTM и SVR, LSTM и ARIMA, или нейросетевые ансамбли [4].

Например, в работе "Long Short-Term Memory Neural Network for Financial Time Series"[4] авторы использовали ансамбли моделей независимой и параллельной архитектур LSTM. Было показано, что LSTM особенно подходят для данных временных рядов из-за их способности включать прошлую информацию, в то время как ансамбли нейронных сетей уменьшают вариативность результатов и улучшают обобщение. Портфель, полученный на основе ансамбля LSTM, обеспечивает лучшую среднюю дневную доходность и более высокую совокупную доходность с течением времени. Более того, портфель LSTM также демонстрирует меньшую волатильность, что приводит к более высокому соотношению риска и доходности.

В другой работе "Comparison of new activation functions in neural network for forecasting financial time series"[5] авторы так же сравнивали различные функции активации именно для рекуррентных нейронных сетей для прогнозирования финансовых временных рядов, но не классического вида:

1. $f(x) = 1 - \exp(-\exp(x))$
2. $f(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{\infty}^{x} e^{-\frac{t^2}{2}} dt$
3. $f(x) = \exp(-\exp(-x))$

Эти функции активации использовались для метода сопряженных градиентов, метода Флетчера – Ривса (eng. Fletcher-Reeves - CGF), и метода Левенберга — Марквардта (eng. Levenberg Marquardt - LM). Первый результат, который можно сделать из этого исследования, заключается в том, что при использовании алгоритма CGF функции активации, предложенные в этом исследовании, получили наилучшую среднюю производительность для меньших сетей, $q = 2, 4, 6$.

Второй вывод заключается в том, что при использовании алгоритма LM [3] модели с функцией активации классической сигмоиды достигли лучших результатов благодаря использованию именно этого алгоритма. Но предложенные функции активации также показали хорошую производительность. Однако авторы подчеркнули, что алгоритм LM работает дольше алгоритма CGF. Более того, хорошая производительность моделей с новыми функциями активации не оказывает большого влияния на изменение алгоритма обучения, как другие функции, описанные в литературе. Как итог, авторы рекомендуют использовать новые функции активации именно для финансовых временных рядов и сетей с небольшой структурой. Авторы также по итогу рекомендуют использовать эти функции в моделях нейронных сетей, использующих алгоритм CGF, так как LM слишком дорог в вычислительном отношении, и требует больше места в памяти.

Таким образом, существующие работы показывают, что использование архитектуры LSTM и поиск оптимальной функции активации может существенно улучшить прогнозирование финансовых показателей, что является важным инструментом на финансовых рынках.

1.2. Роль функции активации

Функция активации в нейронных сетях – это математическая функция, которая принимает входные данные и определяет выходной сигнал. Она 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 Основа искусственных нейронных сетей — это нейроны и функция активации, которая имеет несколько входов и один выход. Нейрон можно рассматривать как композицию других взвешенных нейронов, к нейронам применяется функция активации, а сеть образуется, когда нейроны объединяются в слои [13]. Каждый нейрон по сути выполняет нелинейное преобразование входного сигнала в выходной формуле:

$$\mathrm{y_k} = \phi_k\left(\sum_{\mathrm{i}} w_{ij} * x_i + b_k\right)$$

где w_{ij}- весовые коэффициенты, x_i- входной сигнал, ϕ - функция активации.

По сути функция активации определяет силу выходного сигнала.

1.3. Постановка задачи

Пусть у нас есть временной финансовый ряд y_i с временным интервалом между точками $i = 1, \dots, n$ размером в день, час или минуту. Назовем этот постоянный шаг по оси времени *тиком*. Пусть $\phi_\theta(y)$ - это функция активации с параметрами θ, которые определяют форму функции активации (например, точка нуля, наклон).

Наша задача – найти оптимальную функцию активации $\phi_\theta^*(y)$ для прогнозирования временного финансового ряда на архитектуре LSTM, минимизируя ошибку прогноза.

Математическая постановка задачи:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{n} \left(y_i - \phi_\theta(y_i)\right)^2$$

где θ^*- оптимальные параметры функции активации, минимизирущие ошибку прогноза на временном финансовом ряде.

Множество используемых функций активации:

$$\phi_\theta(y) = \begin{cases} cloglog(y,\theta) \\ loglog(y,\theta) \\ tanh(y,\theta) \\ sigmoid(y,\theta) \\ linear(y,\theta) \\ ReLU(y,\theta) \end{cases}$$

где θ - параметры функции активации.

Цель – найти оптимальную функцию активации ϕ из множества, чтобы минимизировать ошибку прогноза временного финансового ряда.

1.4. Функции активации

В вычислительном эксперименте рассматриваются шесть функций активаций. Четыре из них являются классическими: $tanh, sigmoid, linear, ReL$U. Остальные не такие популярные: $cloglog, loglog$. Вид функций активации, первые производные и их области определения приведены в таблице 1. Графики функций отображены на рис. 3 и 4.

Для того, чтобы мы могли использовать функцию в активационном слое, она должна удовлетворять следующим свойствам [6]:

1. *Дифференцируемость* поскольку при обратном распространении ошибки необходимо вычислять градиенты, использующие производную функции активации.
2. *Монотонное возрастание* обеспечивает стабильный процесс обучения нейронных сетей. Когда функция активации монотонно возрастает, градиенты ошибки эффективнее передаются от последнего слоя нейронной сети к первому. Это помогает избегать проблемы "затухания градиента", когда градиент ошибки становится очень малым и обновление весов прекращается, что может привести к замедлению и остановке обучения. Кроме того, монотонно возрастающая функция активации помогает сети лучше учиться сложным зависимостям в данных, так как она позволяет строить более сложные нелинейные отображения между входными и выходными данными.

Проверим, что *cloglog* и *loglog* являются функциями активации:

Дифференцируемость.
Не трудно заметить, что эти функции бесконечно дифференцируемые, достаточно посмотреть на их производные.

Монотонность.
Докажем для *cloglog*.

$$\text{Пусть } x_1 > x_2$$
$$-\exp(x_1) < -\exp(x_2)$$
$$\exp(-exp(x_1)) < \exp(-\exp(x_2))$$
$$1 - \exp(-exp(x_1)) > 1 - \exp(-\exp(x_2))$$
$$\phi(x_1) > \phi(x_2)$$

Аналогично для *loglog*.

Ограниченность.
Заметим так же, что обе функции являются ограниченными при $x \to \pm\infty$:
Докажем для *cloglog*.

$$\lim_{x\to\infty} \phi(x) = \lim_{x\to\infty} (1 - \exp(-\exp(x))) = 1 - 0 = 1$$
$$\lim_{x\to-\infty} \phi(x) = \lim_{x\to-\infty} (1 - \exp(-\exp(x))) = 1 - 1 = 0$$

2. LSTM

Любая рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети. (рис. 1) LSTM (Long short-term memory) – тип рекуррентной нейросети, модули, которой содержат четыре слоя, вместо одного (рис.2), 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. [7]

Для описания работы архитектуры нейронной сети LSTM для прогнозирования введем несколько основных компонент и операций. Для удобства обозначим временной ряд цен как $\{x^{(1)}, x^{(2)}, \ldots, x^{(T)}\}$, где $x^{(t)}$ - это цена биткоина 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[8], 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”.

2.1 Forget gate

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”.

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

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).

2.2 Input gate

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 $\widetilde{C_t}$, которые можно добавить в состояние ячейки.

$$\begin{cases} \mathrm{i_t} = \sigma\left(\mathrm{W_i} * [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_i}\right) \\ \widetilde{C_t} = tanh(\mathrm{W_C} * [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_C}) \end{cases}$$

2.3 Обновляем состояние ячейки

Заменим старое состояние ячейки C_{t-1} на новое состояние C_t.
Умножим старое состояние на f_t, удаляя то, что мы решили забыть. Затем прибавим $i_t * \widetilde{C_t}$. Это новые значения-кандидаты, умноженные на $t-$ на сколько мы хотим обновить каждое из значений состояния.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

2.4 Output gate

Решим, какую информацию мы хотим получать на выходе. Сначала применим 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.

$$\begin{cases} \mathrm{o_t} = \sigma\left(\mathrm{W_o} * [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_o}\right) \\ \mathrm{h_t} = \mathrm{o_t} * \tanh(\mathrm{C_t}) \end{cases}$$

2.5 Общая формула

Итого, простейший LSTM-модуль можно представить в виде системы уравнений:

$$\begin{cases} f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \\ \mathrm{i_t} = \sigma\left(\mathrm{W_i} * [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_i}\right) \\ \widetilde{C_t} = tanh(\mathrm{W_C} * [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_C}) \\ C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \\ \mathrm{o_t} = \sigma\left(\mathrm{W_o} * [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_o}\right) \\ \mathrm{h_t} = \mathrm{o_t} * \tanh(\mathrm{C_t}) \end{cases}$$

3 МОДЕЛЬ

3.1 *Оценка моделей*

3.1.1 MAE

MAE (Mean Absolute Error) - это метрика, используемая в машинном обучении для оценки точности прогнозов модели. Она показывает среднее абсолютное отклонение между прогнозируемыми значениями модели и их фактическими значениями.

Формула для расчета MAE выглядит следующим образом:

$$MAE = \frac{1}{n} * \sum_{i=1} n \left| y_i - y^i \right|$$

где y_i - фактическое значение, y^i - прогнозное значение, n - количество примеров в тестовом наборе данных.

MAE показывает, насколько сильно модель ошибается в среднем по всем примерам. Она измеряется в тех же единицах, что и целевая переменная, поэтому ее легче интерпретировать, чем другие метрики, такие как RMSE (Root Mean Squared Error).

Чем меньше значение MAE, тем более точными являются прогнозы модели.

3.1.2 RMSE

RMSE (Root Mean Square Error) - это метрика, используемая для оценки точности модели регрессии. Она измеряет среднеквадратичное отклонение прогнозов модели от фактических значений.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - y^i)^2}{n}}$$

где n - количество примеров в выборке, y_i - фактическое значение для i-го примера, y^i - прогнозное значение для i-го примера.

RMSE показывает, насколько сильно прогнозы модели отклоняются от фактических значений в среднем, при этом учитывая квадратичные отклонения. Чем меньше значение RMSE, тем более точной считается модель.

RMSE также может быть интерпретирована как среднее отклонение прогнозов модели от фактических значений, но взвешенное по квадратам отклонений.

3.1.3 MAPE

MAPE (Mean Absolute Percentage Error) - средняя абсолютная процентная ошибка.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - y^i}{y_i} \right| \times 100\%$$

Где n - количество наблюдений, y_i - истинное значение, y^i - прогнозное значение.

MAPE измеряет среднее относительное отклонение прогнозов от фактических значений в процентах. Чем ближе значение MAPE к нулю, тем лучше качество прогноза.

3.2 Функция потерь

Функция потерь (loss function) - это функция, которая измеряет разницу между предсказанными значениями модели и истинными значениями (целевой переменной) на обучающем наборе данных. Она является ключевым компонентом при обучении нейронных сетей и других моделей машинного обучения, так как используется для оценки качества предсказаний и регулирования параметров модели. В данной задаче использовалась функция: mean squared error.

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

Функция потерь используется на каждой итерации обучения для оценки ошибки предсказаний модели на текущем батче или на всем обучающем наборе данных. Затем этот показатель ошибки используется оптимизатором для обновления параметров модели в направлении уменьшения потерь.

3.3 Метод оптимизации

В текущей работе был выбран метод оптимизации Adam (Adaptive Moment Estimate) – это метод оптимизации, который обновляет веса модели на основе градиентов и квадратов градиентов. Adam использует две переменные первого и второго момента градиента (m и v соответственно), чтобы вычислить адаптивные скорости обучения для каждого параметра.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_t \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

где t - номер текущей итерации, g_t - градиент по параметрам на текущей итерации, β_1 и β_2 - коэффициенты забывания для первого и второго м омента (обычно близкие к 1, например, $\beta_1 = 0.9$ и $\beta_2 = 0.999$), m_t и v_t – оценки первого и второго момента градиента на текущей итерации.

Затем происходит коррекция смещения:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

И, наконец, обновление параметров:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

где: θ_t - параметры модели на итерации t, α - скорость обучения (learning rate), ϵ - малое число, добавленное для численной стабильности (обычно очень маленькое, например, 10−8).

Эти формулы позволяют Adam настраивать скорость обучения для каждого параметра в зависимости от его градиента и истории градиентов, что обычно приводит к более быстрому и стабильному обучению модели.

3.4 Выводы

По итогу практический смысл функции потерь и оптимизации состоит в следующем:

1. Функция потерь определяет, насколько хорошо модель предсказывает целевую переменную на обучающем наборе данных. Функция потерь служит целевой метрикой, которую модель пытается минимизировать в процессе обучения. Чем меньше значение функции потерь, тем лучше модель справляется с задачей.
2. Алгоритм оптимизации определяет способ обновления параметров модели в процессе обучения с целью минимизации функции потерь. Оптимизатор решает задачу оптимизации путем настройки параметров модели с помощью методов градиентного спуска или его модификаций. Он направляет процесс обучения модели в направлении уменьшения функции потерь и помогает модели сойтись к оптимальным параметрам.

4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

4.1 Описание программного обеспечения

Для реализации вычислительного эксперимента выбран язык Python [14] – высокоуровневый интерпретируемый язык программирования с динамической строгой типизацией и автоматическим управлением памятью. Язык является полностью объектно-ориентированным, то есть отвечает инкапсуляции, наследованию и полиморфизму. Выбран этот язык программирования, 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[15].

4.2 Используемые библиотеки

1. TensorFlow[16] - 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. Keras [17] - фреймворк, который предоставляет простой и понятный API, позволяющий быстро создавать и обучать модели. Keras также поддерживает множество различных архитектур нейронных сетей
3. Scikit-learn [18] - библиотека машинного обучения, которая использовалась для оценки моделей. Так же она предоставляет множество функций для работы с данными, обучения моделей и оценки их качества.
4. Matplotlib [19] - библиотека для визуализации данных. Использована для визуализации исторических данных о цене и результатов прогнозирования, а также для анализа их свойств.
5. yfinance [20] - 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.
6. datetime [21] - это библиотека для работы с датами и временем в Python.

5. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

5.1 Задачи для вычислительного эксперимента

Ниже описан список задач, которые нужно выполнить в ходе вычислительного эксперимента:

1. Выбрать финансовый инструмент. Мы хотим проверить результаты при разном масштабе по оси времени, поэтому нужен ликвидный инструмент с большим объемом исторических данных по нескольким тикам.
2. Найти и обработать датасет, разделить обработанные данные на обучающую и тестовую выборки.
3. Нормализовать данные. Это делается для ускорения работы нейросети. К тому же нормализация уменьшает риск переобучения.
4. Построить модель с LSTM-слоем.
5. Сделать прогнозы на архитектуре LSTM с разными функциями активации.
6. Провести обратную нормализацию, чтобы вернуться к исходному масштабу.
7. Сравнить модели, используя метрики: $MAE, RMSE, MAPE$
8. Проанализировать результаты применения модели к тестовой выборке.

5.2 Описание вычислительного эксперимента

5.2.1 Загрузка данных

Для вычислительного эксперимента мы выбрали курс биткоина к доллару (‘BTC-USD’) с начала 2018 года до 1 февраля 2024 года. В качестве тиков взяты один день, час и минута.

5.2.2 Нормализация данных

Нормализуем данные временного ряда. Для каждого элемента x в выборке:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

где x_{scaled} - нормализованное значение x, принимающее значение от 0 до 1, x_{min} - минимальное значение в выборке, x_{max}- максимальное значение в выборке.

5.2.3 Настраиваемые гиперпараметры

Для начала сделаем несколько важных определений:

1. $Lookback$ в LSTM (Long Short-Term Memory) - количество временных шагов, на которые модель может "возвращаться" при обучении. Этот параметр позволяет учитывать зависимости в данных на различных промежутках времени и улучшать способность модели предсказывать будущие значения. В данном случае, модели LSTM обучались и тестировались с различными значениями $lookback$, чтобы определить оптимальное количество временных шагов для данной задачи.

2. Число эпох. Одна эпоха — это полное прохождение всех обучающих данных через нейронную сеть один раз. Недостаток эпох приводит к недообучению (слабая обобщающая способность модели как на тренировочных, так и на тестовых данных), переизбыток — к переобучению (модель слишком сильно “подстроилась” под обучающие данные, при этом показывая плохую эффективность на тестовых).
3. Размер батча — количество строк данных, подающихся на вход модели за одну итерацию. Этот параметр определяет количество входных данных (подмножества обучающих данных), по которым будет рассчитываться градиент, необходимый для обратного распространения ошибки.

Проведено обучение 12 моделей, включающее в себя 6 различных функций активации на моделях с LSTM слоем.

Выбранные гиперпараметры для моделей:

- $Lookback$ в работе составлял 13 шагов (дней или часов).
- *Число эпох*. Решено было провести обучение на 100 эпохах. На вопрос о единственно верном количестве эпох ответить заранее невозможно, этот параметр подбирается экспериментально.
- *Размер батча*. В работе $batchsize = 32$.

Создаем пустую модель с использованием класса Sequential из библиотеки keras. Затем в модель добавляется слой LSTM с 32 нейронами и входной формой данных вида $(lookback, 1)$, где $lookback$ означает количество временных шагов, а 1 - количество признаков для каждого временного шага, у нас одномерный временной ряд, поэтому значение такое.

Далее к модели добавляется полносвязный слой Dense с одним выходным нейроном, который будет предсказывать значения целевой переменной.

После этого модель компилируется с помощью функции compile. Здесь определяется функция потерь (loss function), которая будет использоваться для оценки ошибки модели, а также алгоритм оптимизации, который будет использоваться для обновления весов модели в процессе обучения. В данном случае обучение моделей происходит путем минимизации функции потерь ’mean squared error’ с помощью алгоритма оптимизации ’adam’.

Затем модель обучается при помощи метода fit. В данном случае, модель обучается на данных с $batchsize = 32$, то есть на каждом шаге градиентного спуска используется 32 образца данных. Количество эпох (epochs) для обучения установлено равным 100.

5.2.4 Обратное восстановление значений

Для обратного преобразования (восстановления исходных значений из нормализованных) используются следующие формулы:

$$x_{original} = x_{scaled} \cdot (x_{max} - x_{min}) + x_{min}$$

где $x_{original}$ - исходное значение x, x_{scaled} - нормализованное значение x, x_{min} - минимальное значение в выборке, x_{max}- максимальное значение в выборке.

Таким образом, предсказание не будет локализоваться внутри отрезка, в котором лежали тренировочные данные. Для тренировочных данных используется 75% наблюдений и 25% для прогноза.

5.3 Результаты вычислительного эксперимента

Будем говорить, что функция активации сработала хорошо, если ее результаты по метрикам ($RMSE, MAE, MAPE$) будут лучше, чем у остальных рассматриваемых, аналогично, сработала плохо, если метрики хуже.

Хорошо сработавшие функции активации по результатам эксперимента для BTC-USD (рис.5):

1. $step = 1m$ (для минутного шага): Функции активации $tanh$ и $cloglog$ показали наилучшие результаты по всем метрикам ($RMSE, MAE, MAPE$). Хуже всего сработало с функцией активации $sigmoid$.
2. $step = 1h$ (для часового шага): Функции активации $tanh$ и $cloglog$ также продемонстрировали лучшие результаты по всем метрикам. Хуже всего сработало с функцией активации $sigmoid$.
3. $step = 1d$ (для дневного шага): Функция активации $cloglog$ показала лучшие результаты по всем метрикам. Хуже всего сработало с функцией активации $loglog$.

Хорошо сработавшие функции активации по результатам эксперимента для ETH-USD (рис. 6):

1. $step = 1m$ (для минутного шага): Функции активации $tanh$ и $cloglog$ показали наилучшие результаты по всем метрикам ($RMSE, MAE, MAPE$).
2. $step = 1h$ (для часового шага): Функции активации $tanh$ продемонстрировали лучшие результаты по всем метрикам. $cloglog$ - показала худшие результаты
3. $step = 1d$ (для дневного шага): Функция активации $cloglog$ показала лучшие результаты по всем метрикам, кроме метрики $MAPE$. Хуже всего сработало с функцией активации $sigmoid$.

5.4 Оценка сходимости

Теперь рассмотрим, как ведет себя loss-функция на тестовом и обучающем наборе. В качестве примера взяты прогнозы для дневных тиков пары "BTC-USD". Так как качественно графики одинаковы, оставим на рис.7 график только для одной из функций активации:

1. На графиках получилась убывающая тенденция. Это означает, что модель успешно уменьшает ошибку на обучающем наборе данных в процессе обучения. Значит даже неклассические функции активации сработали хорошо.
2. Стабилизация функции потерь: после начальной фазы обучения, функция потерь стабилизируется и перестает значительно изменяться. Везде это примерно после 30 эпохи. Значит можно было взять 30 эпох вместо 100. Это указывает на то, что модель достигла оптимальных параметров и больше не улучшается.
3. Малая разница между обучающей и тестовой функцией потерь. Оранжевый график для тестового набора и синий для тренировочной части сходятся: обученные модели имеет малую разницу между значениями функции потерь на обучающем и тестовом (или валидационном) наборах данных. Это указывает на то, что модель хорошо обобщает данные и избегает переобучения.
4. Отсутствие больших колебаний: график сходимости обученных моделей больших колебаний или "шума". Это свидетельствует о стабильности обучения.

5.5 Выводы

В результате получено, что предложенные функции $cloglog$ и $loglog$ хорошо показали себя в работе LSTM-алгоритма. Лучшие результаты были получены для функций активаций $tanh$ и $cloglog$. Для прогноза цены BTC-USD на всех видах тиков у $cloglog$ лучшие результаты: наименьшее $RMSE, MAE, MAPE$. По полученным результатам на всех видах тиках неклассическая функция активации $loglog$ сопоставима по работе с $sigmoid$ функцией.

Как можно заметить, у $cloglog$ и $loglog$ оказались так же лучшие скорости сходимости: примерно с $7 - 10$ эпохи для дневных и часовых тиков оранжевые и синие графики сходятся. Это так же позволяет ускорить программу, уменьшив число эпох для обучения

6. АНАЛИЗ ГИПЕРПАРАМЕТРОВ UNITS И LOOKBACK

6.1 Гиперпараметры LSTM в TensorFlow

Что обозначают входные параметры LSTM в TensorFlow? Этот вопрос важен, так как от подготовки данных и определения параметров зависит точность и скорость работы модели.

Приведем датасет к размерности $[batch, timesteps, feature]$ [9], где $feature$ - количество столбцов-признаков, на основе которых строим прогноз, $timesteps$ - количество предыдущих значений в последовательности (длина окна, далее обозначим как $lookback$), $batch$ - количество получившихся последовательностей (samples) размера $[timesteps, feature]$

Важным аргументом конструктора для всех слоев Keras RNN, таких как tf.keras.layers.LSTM, является аргумент $return\_sequences$ [10]. Этот параметр может настроить слой одним из двух способов:

- Если $return\_sequences = False$ (по умолчанию), слой возвращает только выходные данные последнего временного шага, давая модели время, чтобы обработать свое внутреннее состояние (warmup), прежде чем делать один прогноз.
- Если $return\_sequences = True$, слой возвращает результат для каждого входа. Этот способ используется, когда в модели есть несколько RNN слоев, или же в обучении модели на нескольких временных шагах одновременно.

Еще один немаловажный параметр - $units$. Это размерность выходного пространства. То есть наша модель берет на вход тензор $[batch, timesteps, feature]$ и на выход дает тензор $[batch, timesteps, units]$, если $return\_sequences = True$, и $[batch, units]$, если $return\_sequences = False$, соответственно [11]. Также $units$ отвечает за размер скрытого слоя LSTM, размера матрицы весов. То есть, чем больше $units$, тем точнее будет модель, но при этом скорость работы будет увеличиваться из-за количества вычислений. [12]

6.2 UNITS

Как влияет $units$ на прогноз? Как упоминалось выше, $units$ отвечает за размер матрицы весов, поэтому, чем больше $units$, тем лучше прогноз.

Будем менять параметр $units = [1,2,4,16,32,64,128]$ при фиксированных $lookback = 13$, на поминутном(1m), почасовом(1h), подневном(1d) датасете, используя для прогноза цену закрытия (Close). То есть спрогнозируем значение для следующего момента времени, основываясь на последних 13 наблюдениях.

6.2.1 Временной ряд с минутными тиками

При маленьких значениях параметра $units = 1, 2, 4$, функция активации $ReLU$ делает прогноз значительно хуже, чем остальные функции. Это видно по значениям метрик качества, что в десятки раз выше, чем у моделей с другими функциями активации.

$ReLU$ стоит использовать с большим числом $units$. По значениям метрик регрессии $RMSE, MAE, MAPE$ в зависимости от параметра $units$ видно[1]: функции $ReLU$ и $linear$ дают наилучший прогноз. Новые функции $cloglog$ и $loglog$ сравнимы по точности с классическими функциями $tanh$ и $sigmoid$ при маленьких $units$, по мере увеличения параметра точность ухудшается быстрее, чем у других функций активации (выбросы на 32 и 64). Видим, что на минутных тиках не стоит выбирать большое количество $units$. Значение 16 будет оптимальным почти для всех функций.

6.2.2 Временной ряд с часовыми тиками

На больших числах $units$ точность всех функций возрастает. Самый точный прогноз дает линейная функция активации при $units = 16, 32, 64$. Новая функция $loglog$ хорошо прогнозирует при $units = 128$. При маленьких значениях – $cloglog, tanh$.

Оптимальный выбор параметра $units$ для часовых тиков: $16, 32$.

6.2.3 Временной ряд с дневными тиками

Прогнозы на дневных тиках всех функций хуже, чем на минутных и часовых. Это видно из значений $MAPE$, выражающихся в процентах[22].

Для всех функций в совокупности наилучшее значение $units = 128$ (кроме $cloglog$) или 32. Наилучший результат дает $cloglog, 16$.

6.2.4 Время работы

С увеличением $units$, возрастает время обучения(рис. 9). Нейросеть с линейной функцией активации обучается быстрее остальных, вне зависимости от тиков.

Скорость обучения изменяется и от числа наблюдений, поэтому обучение на дневных тиках происходит быстрее, чем на минутных и часовых, так как минутных и часовых наблюдений около 10тыс., а дневных 2,5тыс.

[1] Так как графиков много, оставим их в [22]. Для примера оставим один на рис.8.

6.3 LOOKBACK

Узнаем, как влияет параметр $lookback$ на прогноз. Будем менять параметр $lookback = [1,2,5,10,30,60]$ при фиксированном $units = 16$ на поминутном(1m), почасовом(1h), подневном(1d) датасете, используя для прогноза цену закрытия (Close).

6.3.1 Временной ряд с минутными тиками

Результаты работы почти всех функций сопоставимы (кроме выбросов в 10). При маленьких значениях $lookback$ наилучший прогноз у $sigmoid$. При больших значениях переменной лучше остальных себя показывают $ReLU$ и линейная функции, а новые функции $cloglog$ и $loglog$ на $5 - 10$ минутах дают наибольшую ошибку.

На минутных тиках стоит выбирать $lookback = 1$ и функции $linear, loglog, ReLU$ или $lookback = 30$ и функции $tanh, ReLU$.

6.3.2 Временной ряд с часовыми тиками

Точность прогноза на часовых тиках при значении $lookback = 2$ почти у всех функций(кроме $cloglog$) точности сопоставимы. Наилучшая точность у функции $linear$ при $lookback = 1, 30, 60$

6.3.3 Временной ряд с дневными тиками

Дневные тики стоит прогнозировать при значении параметра $lookback = 1$ с использованием любой функции активации кроме $ReLU$.

6.3.4 Время работы

С увеличением $lookback$ возрастает время обучения нейросети, так как увеличивается объем обрабатываемых данных. Как и в предыдущем случае, обучение на дневных тиках происходит в разы быстрее из-за меньшего числа наблюдений.

Все рассматриваемые функции активации затрачивают сопоставимое время на обучение.

6.4 ВЫВОДЫ

Мы убедились, что функции $cloglog$ и $loglog$ могут быть использованы для решения задач регрессии. В некоторых случаях они дают более точный результат по сравнению с классическими функциями. $cloglog$ почти всюду точнее $loglog$.

Вывод о выборе лучших параметров $units$ и $lookback$ приведен в таблице 2.

Итого, наилучшие параметры LSTM-нейросети для задачи регрессии прогнозирования финансового временного ряда (вне зависимости от тика) это:

- $units = 16$
- $lookback = 1$
- функция активации: $linear$

Эти параметры стоит использовать для первого запуска нейросети, и, при необходимости, меняя параметры согласно таблице 2, можно донастроить точность.

7. ЗАКЛЮЧЕНИЕ

В результате вычислительных экспериментов было выявлено, что представленные неклассические функции $cloglog$ и $loglog$ демонстрируют свою эффективность при использовании в LSTM архитектуре нейронной сети с настроенными гиперпараметрами $units$ и $lookback$ для всех видов тиков: для прогноза цены BTC-USD у $cloglog$ лучшие результаты по метрикам RMSE, MAE, MAPE, а функция активации loglog сопоставима по работе с sigmoid функцией, для ряда ETH-USD прогноз с функцией $cloglog$ сопоставим с результатами функции активации tanh.

Кроме того, эмпирическим путем для каждого вида тика определены функции активации и значения $units$ и $lookback$, дающие наилучший прогноз, и начальные значения параметров для первого запуска процесса обучения, меняя которые согласно таблице 2, можно улучшить точность прогноза в зависимости от решаемой задачи.

Эти результаты важны для создания новых архитектур LSTM-сетей для прогнозирования финансовых временных рядов и для проведения дальнейших исследований в этой области. Например, для исследования параметров функции активации, определяющих ее форму: наклон, точку нуля

БЛАГОДАРНОСТИ

Авторы выражают благодарность Научно-образовательной школе МГУ «Мозг, когнитивные системы, искусственный интеллект» за поддержку.

КОНФЛИКТ ИНТЕРЕСОВ

Процесс написания и содержание статьи не дает оснований для постановки вопроса о конфликте интересов.

СООТВЕТСТВИЕ ЭТИЧЕСКИМ СТАНДАРТАМ

Данная статья является полностью оригинальным произведением ее авторов, прежде не опубликована и до получения решения редколлегии PRIA о непринятии ее к публикации не будет направляться в другие издания

СПИСОК ЛИТЕРАТУРЫ

[1] Артамонов Н. В., Введение в анализ временных рядов: учебное пособие для вузов/ Н. В.Артамонов, Е. А. Ивин, А. Н. Курбацкий, Д. Фантаццини: ФГБУН ВолНЦ РАН, 2021– 134

[2] Hyndman, R.J., Athanasopoulos, G. (2018) Forecasting: principles andpractice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2. https://otexts.com/fpp2/index.html (04.05.2024).

[3] Нильсен, Эйлин.Практический анализ временных рядов: прогнозирование со статистикой и машинное обучение/Нильсен, Эйлин: Пер. с англ. под ред. докт. физ.-мат. наук Д.А. Клюшина: ООО “Диалектика”, 2021. — 544 с.

[4] Carmina Fjellstr…om, Long Short-Term Memory Neural Network for Financial Time Series https://arxiv.org/abs/2201.08218 (20.01.2022)

[5] Gecynalda S da S. Gomes, Teresa B. Ludermir, Leyla M., M. R. Lima, Comparison of new activation functions in neural network for forecasting financial time series, https://www.researchgate.net/publication/225165990_Comparison_of_new_activation_functions_in_neural_network_for_forecasting_financial_time_series (04.2011)

[6] Sreekanth Tadakaluru Activation Functions in Neural Networks Explained (https://www.mygreatlearning.com/blog/activation-functions/) (05.03.2024)

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

[8] Christopher Olah Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (14.05.2023)

[9] Mohhamad Fneish Keras_LSTM_Diagram https://github.com/MohammadFneish7/Keras_LSTM_Diagram (20.05.2023)

[10] Time series forecasting https://www.tensorflow.org/tutorials/structured_data/time_series?hl=ru#recurrent_neural_network (20.05.2023)

[11] Shiva Verma Input and Output shape in LSTM (Keras) https://www.kaggle.com/code/shivajbd/input-and-output-shape-in-lstm-keras (20.05.2023)

[12] Jiyang Wang What is "units"in LSTM layer of Keras? https://zhuanlan.zhihu.com/p/58854907 (20.05.2023)

[13] Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с.

[14] Python documentation https://www.python.org/ (14.11.2024)

[15] Google Colaboratory https://colab.google/ (14.11.2024)

[16] TensorFlow documentation https://www.tensorflow.org/ (14.11.2024)

[17] Keras documentation https://keras.io/ (02.05.2024)

[18] Scikit-learn documentation. https://scikit-learn.org/ (02.05.2024)
[19] Matplotlib documentation. https://matplotlib.org/stable/ (02.05.2024)
[20] Yfinance documentation. https://pypi.org/project/yfinance/ (02.05.2024)
[21] Datetime documentation. https://docs.python.org/3/library/datetime.html (02.05.2024)
[22] Дополнительные материалы к статье https://github.com/MakhovaAnastasia/Paper-activation-functions-units-lookback

ТАБЛИЦЫ

Таблица 1. Функции активации, их первые производные и области значений.

Функция активации	$\phi(y) =$	$\phi'(y) =$	Область значений
cloglog	$1 - \exp(-\exp(y))$	$\exp(y * (1 - \phi(y)))$	$(0; 1)$
loglog	$\exp(-\exp(-y))$	$\exp(-y * (1 - \phi(y)))$	$(0; 1)$
tanh	$\frac{\exp(2y) - 1}{\exp(2y) + 1}$	$1 - \phi^2(y)$	$(-1; 1)$
sigmoid	$\frac{1}{1 + \exp(-y)}$	$\phi(y) * (1 - \phi(y))$	$(0; 1)$
linear	y	1	$(-\infty; +\infty)$
ReLU	$\max(0; y)$	$\mathbb{1}_{y \geq 0}$	$[0; +\infty)$

Таблица 2. Сводная таблица для выбора параметров *units* и *lookback* LSTM-нейросети. Подходящие функции активации выписаны в порядке убывания точности.

тик	*units*	*lookback*
1m	16(*tanh*, *ReLU*, *linear*)	1(*linear*, *loglog*, *ReLU*), 30 (*tanh*, *ReLU*)
1h	16,32 (*linear tanh*, *cloglog*)	1, 30, 60 (*linear*)
1d	16*(cloglog*, *linear*)	1 (все, кроме *ReLU*)

ПОДПИСИ К РИСУНКАМ

Рис. 1. Структура LSTM, развернутая во времени, напоминает цепочку.

Рис. 2. Архитектура LSTM. Обозначения: c_* —состояние ячейки, x_*— вход, h_*—выход , × — поточечное умножение, + — сложение векторов, σ — сигмоидальный слой, $tanh$ — $tanh$- слой.

Рис. 3. Графики функций активации: $cloglog, loglog,\ tanh, sigmoid$

Рис 4. $cloglog, loglog,\ tanh, sigmoid$ на одном графике

Рис. 5. Прогноз LSTM в зависимости от функции активации, $step = 1m/1h/1d$ минутные/часовые/дневные тики. Чем темнее - тем хуже для пары BTC-USD

Рис. 6 Прогноз LSTM в зависимости от функции активации. $step = 1m/1h/1d$ минутные/часовые/дневные тики. Чем темнее - тем хуже для пары ETH-USD

Рис. 7 Линия в точку – сходимость на тренировочных данных, пунктирная – сходимость на тестовых, для функции активации $tanh$, тики дневные. Графики для других тиков и функций можно найти в [22].

Рис. 8 Минутные тики. График метрик регрессии на *тренировочном* датасете в зависимости от параметра $units$. Здесь и далее часть значений для $ReLU$ не отображается, чтобы не нарушать масштаб. Графики для других тиков и датасетов можно найти в [22]

Рис. 9 Минутные тики. Время обучения нейросети. Графики для других тиков можно найти в [22]

РИСУНКИ

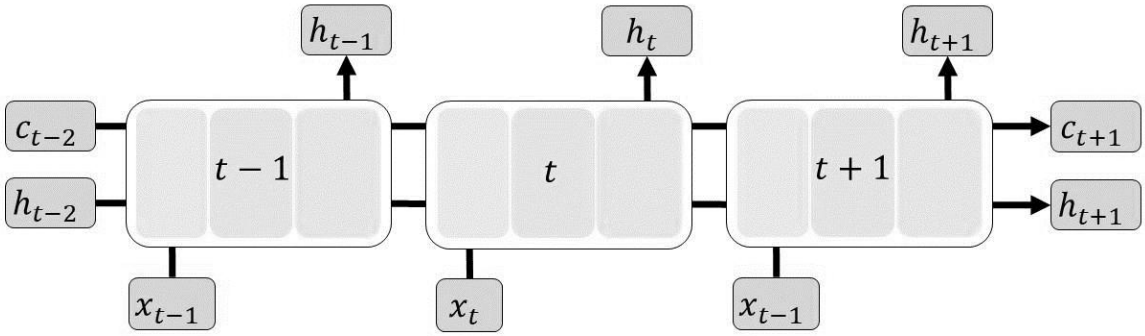

Рис. 1.

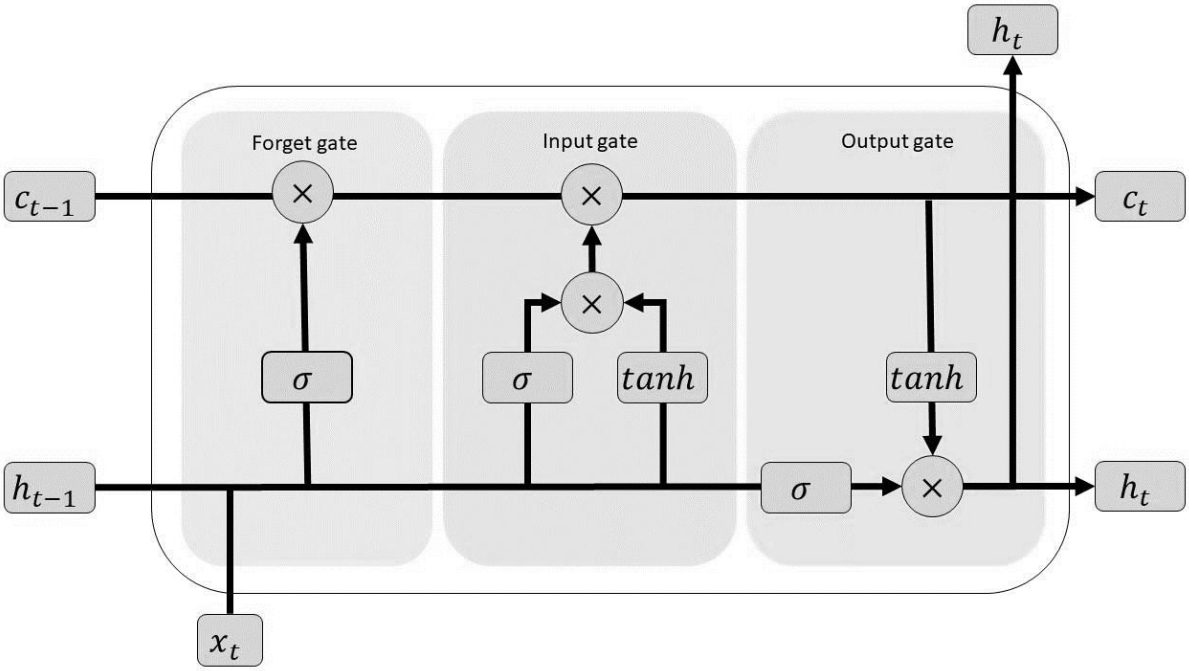

Рис. 2.

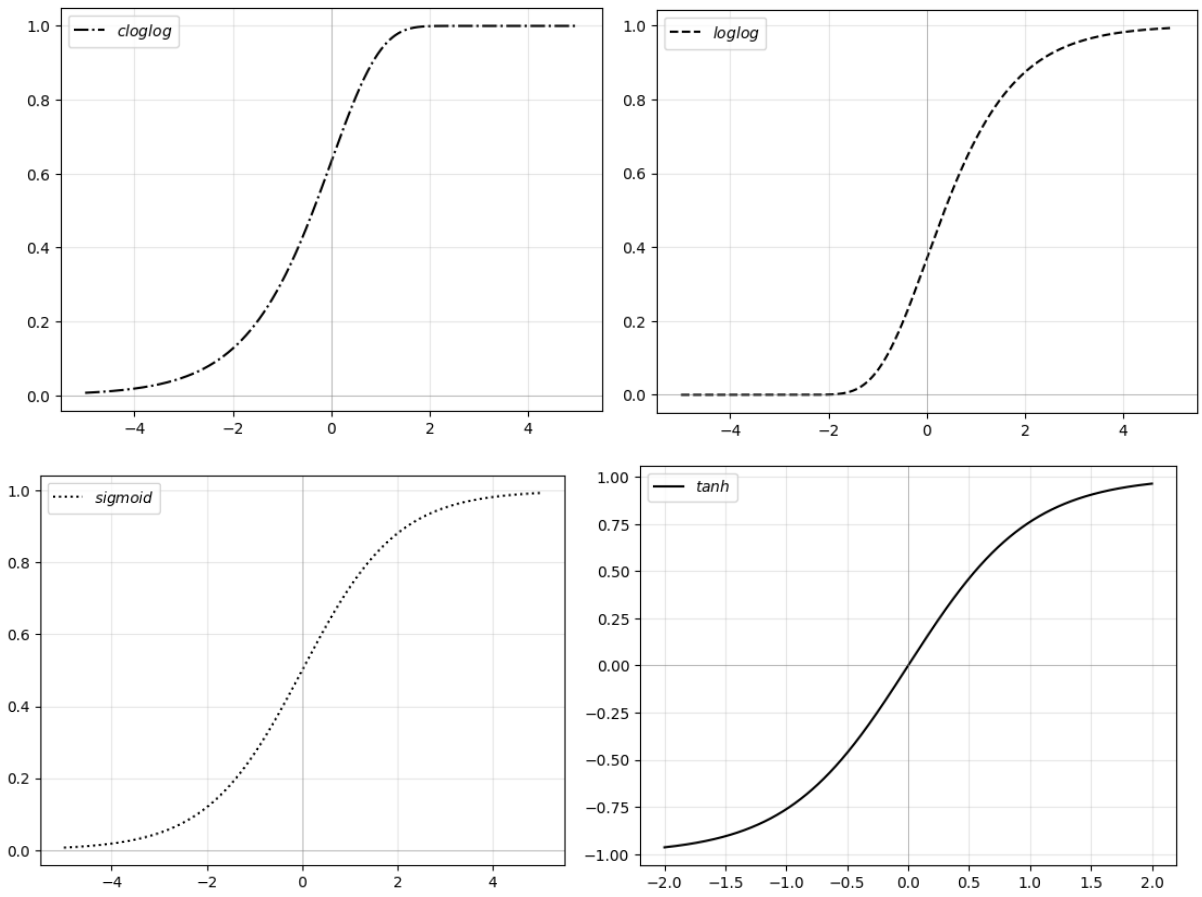

Рис. 3.

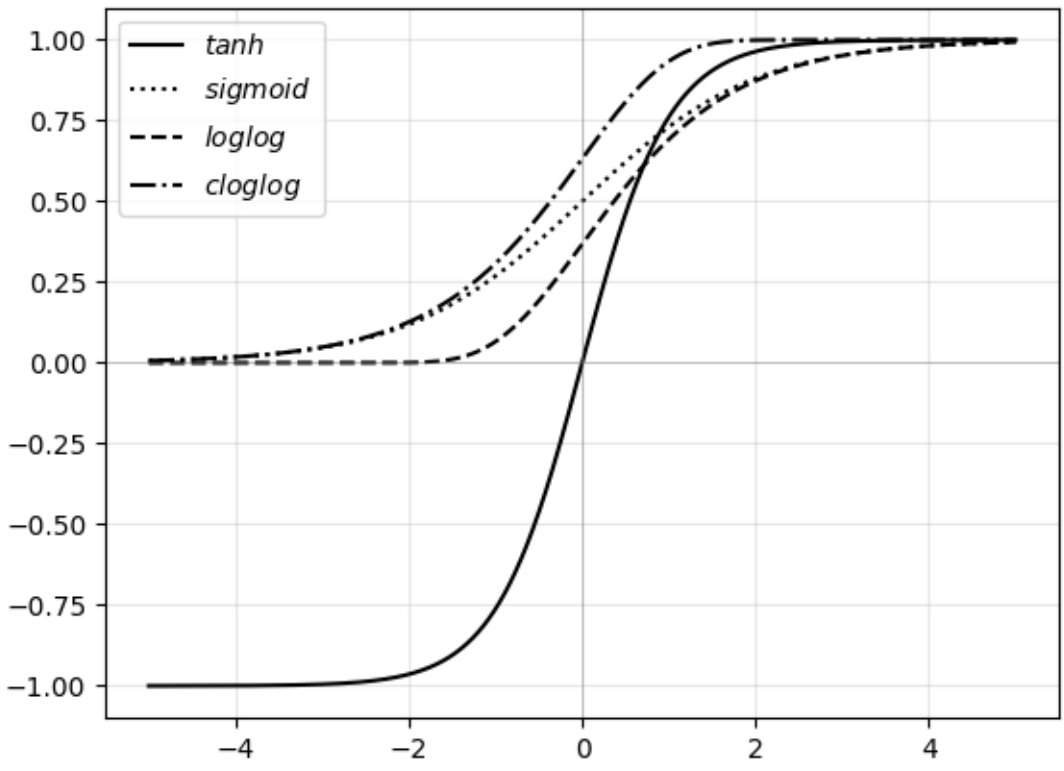

Рис. 4

step=1m	tanh	sigmoid	loglog	cloglog
rmse test	63,75	67,29	66,90	64,47
mae test	38,91	40,75	39,76	37,29
mape test	0,06	0,06	0,06	0,05
step=1h	tanh	sigmoid	loglog	cloglog
rmse test	167,52	182,83	181,03	166,67
mae test	125,33	137,87	135,95	119,73
mape test	0,57	0,61	0,60	0,54
step=1d	tanh	sigmoid	loglog	cloglog
rmse test	417,16	568,93	568,25	385,96
mae test	303,76	445,54	490,34	263,83
mape test	4,52	6,57	8,01	3,78

Рис. 5

step=1m	tanh	sigmoid	loglog	cloglog
rmse test	3,27	3,43	3,44	3,31
mae test	1,93	2,09	2,04	1,92
mape test	0,29	0,30	0,29	0,28
step=1h	tanh	sigmoid	loglog	cloglog
rmse test	9,53	9,58	9,87	18,00
mae test	6,14	5,99	6,40	16,07
mape test	0,06	0,06	0,06	0,07
step=1d	tanh	sigmoid	loglog	cloglog
rmse test	37,52	40,71	39,64	34,59
mae test	24,04	29,58	28,33	21,19
mape test	0,06	0,06	0,06	0,07

Рис. 6

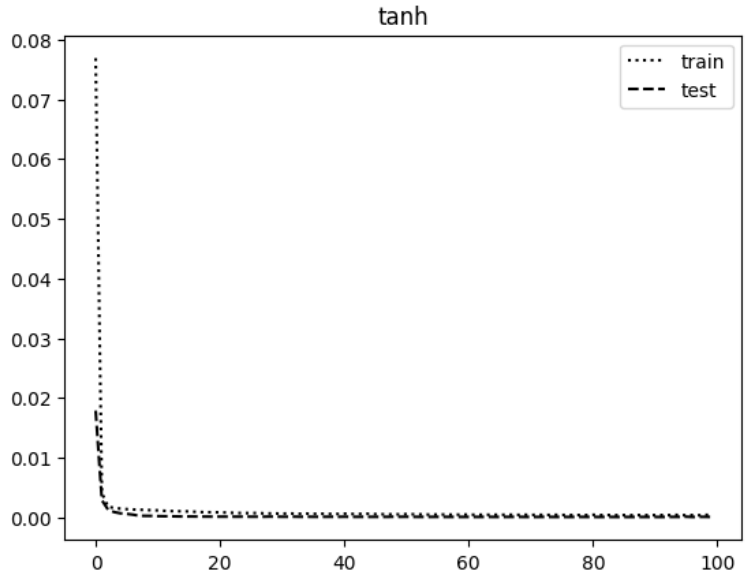

Рис.7

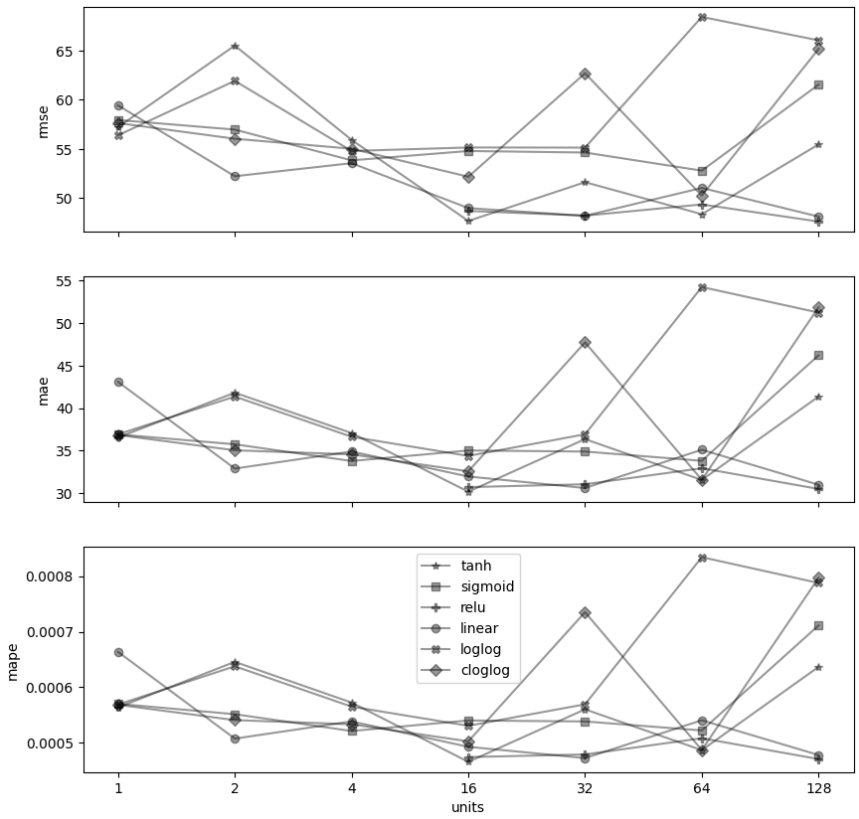
rmse
mae
mape
tanh
sigmoid
relu
linear
loglog
cloglog
units

Рис. 8

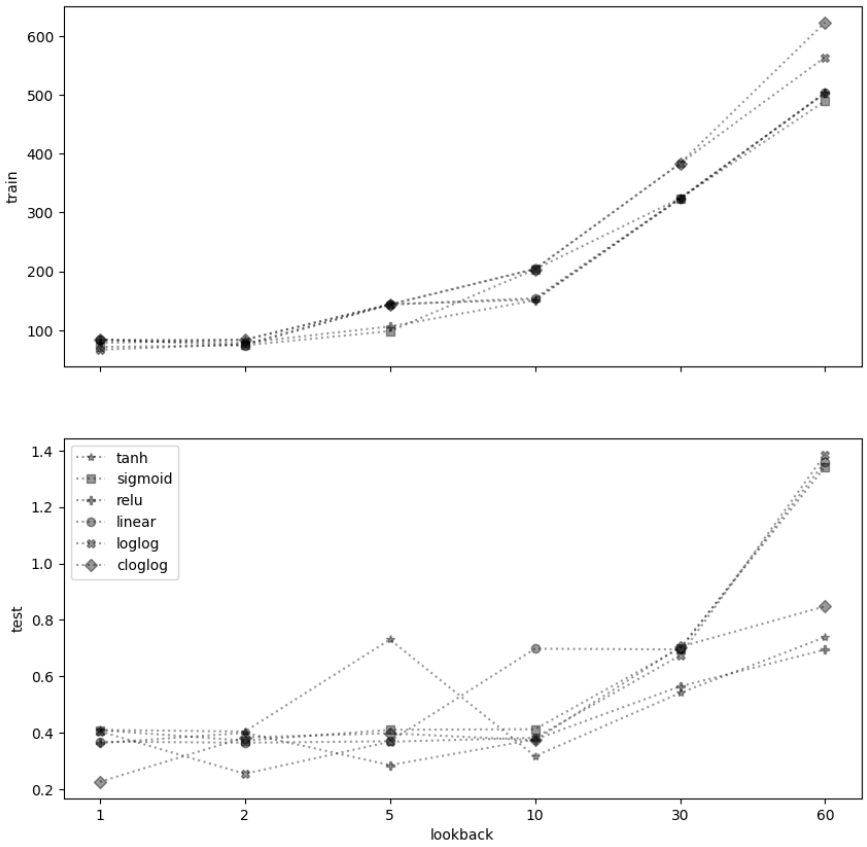
train
test
tanh
sigmoid
relu
linear
loglog
cloglog
lookback

Рис. 9

КРАТКИЕ БИОГРАФИИ И ФОТОГРАФИИ ВСЕХ АВТОРОВ