УДК 519.24
DOI 10.19110/1994-5655-2021-6-14-19

А.О. МАРГАСОВ

О НЕЙРОННЫХ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЯХ И ИХ ВЕРОЯТНОСТНОМ РАСШИРЕНИИ

Кафедра математической статистики, Московский государственный университет им. М.В. Ломоносова, г. Москва

margasovarsenij@gmail.com

Аннотация

В данной статье описывается переход от нейросетевой архитектуры к обыкновенным дифференциальным уравнениям и задаче Коши. Рассматривается сравнение двух нейросетевых архитектур: классическая RNN и ODE-RNN, в которой используются нейронные обыкновенные дифференциальные уравнения. В работе предлагается новая архитектура p-ODE-RNN, позволяющая добиться качества, сопоставимого с ODE-RNN, но при этом обучается значительно быстрее. Кроме того, рассматривается вывод предложенной архитектуры с точки зрения теории случайных процессов.

Ключевые слова:
обыкновенные дифференциальные уравнения, нейронные сети, случайные процессы, вероятностные распределения

A.O. MARGASOV

NEURAL ORDINARY DIFFERENTIAL EQUATIONS AND THEIR PROBABILISTIC EXTENSION

Department of Mathematical Statistics, Lomonosov Moscow State University, Moscow

Abstract

This paper describes the transition from neural network architecture to ordinary differential equations and initial value problem. Two neural network architectures are compared: classical RNN and ODE-RNN, which uses neural ordinary differential equations. The paper proposes a new architecture of p-ODE-RNN, which allows you to achieve a quality comparable to ODE-RNN, but is trained much faster. Furthermore, the derivation of the proposed architecture in terms of random process theory is discussed.

Keywords:
ordinary differential equations, neural networks, stochastic processes, probability distributions

Введение

Ежедневно в современном мире появляется огромное количество новых данных, которые генерируются в различных областях человеческой деятельности: от сельского хозяйства до сервисов по заказу такси. Такие объёмы информации позволяют применять новые подходы к построению моделей, обработке и аналитике на основе полученных данных. Помимо классических алгоритмов машинного обучения, которые не всегда могут предложить необходимое качество решения вышеописанных задач, стоит отметить парадигму глубокого обучения (deep learning), поскольку модели, построенные в её рамках, позволяют решать многие востребованные и ранее недоступные задачи.

Стандартная методология решения новых задач в различных областях математики предполагает сведение новой задачи к старой и последующее рассмотрение в рамках ранее полученного алгоритма решения известной задачи. В связи с этим особый интерес представляют различные подходы рассмотрения методов глубокого обучения в рамках ранее хорошо изученных областей математики. Рассматриваемый в работе новый класс таких моделей (нейронные обыкновенные дифференциальные уравнения (НОДУ)) позволяет использовать все возможности и гибкость парадигмы глубокого обучения [1], опираясь на достаточно изученную теорию обыкновенных дифференциальных уравнений.

Наиболее известной нейросетевой архитектурой, естественно трансформирующейся к нейронному дифференциальному уравнению, является

ResNet [2]. Эти преобразования и формализация НОДУ описаны в статье [3], которая стала началом развития указанной тематики в сообществе исследователей и аналитиков данных. В этой же работе используется метод, позволяющий уменьшить затраты оперативной памяти для обучения НОДУ. Дальнейшее развитие идеи о применении гибкости дифференциальных уравнений в рамках НОДУ реализовано в модели ODE-RNN, которая позволяет работать с данными, полученными через разные промежутки времени, и описано в работе [5].

Далее, для векторных величин будет использоваться жирный шрифт, для скалярных – обычный, если не оговорено иного. Например: $\boldsymbol{h}_t$ – вектор, а t – скаляр.

1. Нейронные обыкновенные дифференциальные уравнения (Neural ODE)

1.1. Мотивация

Большинство моделей глубокого обучения представляют собой композиции сложных последовательных функциональных преобразований скрытого состояния $\boldsymbol{h}_t$ (для нейронной сети – это скрытый слой с номером t), каждая из которых позволяет приблизить любую функцию из $\mathcal{L}^p$ (при выполнении простых условий на функцию активации и число скрытых слоёв) [1].

Определение 1.1
Нейронная сеть – вычислительный граф, в котором: каждая вершина с выходящей из нее дугой – это или тензор, или матрица, или вектор, или скаляр; каждая вершина с входящим в неё ребром – функция, зависящая от вершин, дуги которых входят в неё; каждая дуга указывает функциональную зависимость.

Получается, что можно представить следующий скрытый слой в сети через некоторую функцию $F(\cdot)$:

$$\boldsymbol{h}_{t+1} = F(\boldsymbol{h}_t).$$

Архитектура остаточной нейронной сети (ResNet), предложенная в [2], позволяет представить скрытое состояние в таком виде:

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + f(\boldsymbol{h}_t, t, \boldsymbol{\theta}), \tag{1}$$

где $\boldsymbol{\theta}$ – параметры нейросети f. Можно заметить, что если переписать (1) в виде:

$$\frac{\boldsymbol{h}_{t+1} - \boldsymbol{h}_t}{(t+1) - t} = f(\boldsymbol{h}_t, t, \boldsymbol{\theta}),$$

то полученное выражение является дискретизацией Эйлера для обыкновенного дифференциального уравнения:

$$\frac{d\boldsymbol{h}(t)}{dt} = f(\boldsymbol{h}(t), t, \boldsymbol{\theta}), \quad t \in [t_0, t_1]. \tag{2}$$

Полученное выражение является нейронным обыкновенным дифференциальным уравнением (НОДУ) [3]. Таким образом, увеличивая число слоёв в сети и уменьшая шаг дискретизации, можно получить функцию $\boldsymbol{h}(t)$, которая определяется (2) без привязки к конкретной архитектуре нейронной сети.

1.2. Задача Коши для НОДУ

Легко заметить, что $\boldsymbol{h}(t_0) = \boldsymbol{h}_0$ – вход нейронной сети, а $\boldsymbol{h}(t_1) = \boldsymbol{h}_1$ – её выход на последнем слое. Имея НОДУ (2) и начальное условие $\boldsymbol{h}(t_0) = \boldsymbol{h}_0$, получаем задачу Коши:

$$\begin{cases} \dfrac{d\boldsymbol{h}(t)}{dt} = f(\boldsymbol{h}(t), t, \boldsymbol{\theta}), \quad t \in [t_0, t_1], \\ \boldsymbol{h}(t_0) = \boldsymbol{h}_0. \end{cases}$$

Обучение полученной модели заключается в минимизации дифференцируемой функции потерь $L(\cdot)$:

$$L\big(\boldsymbol{h}(t_1)\big) = L\left(\boldsymbol{h}(t_0) + \int_{t_0}^{t_1} f(\boldsymbol{h}(t), t, \boldsymbol{\theta})\, dt\right) \approx$$
$$\approx L\big(\text{ODESolve}(t_0, t_1, \boldsymbol{h}(t_0), \boldsymbol{\theta}, f)\big),$$

где ODESolve$(\cdot)$ – численный метод решения обыкновенного дифференциального уравнения.

2. Сравнение рекуррентной нейронной сети и прозводной от нее архитектуры ODE-RNN

2.1. RNN

Рассмотрим сначала стандартную модель рекуррентной нейронной сети (RNN), которая на вход получает набор данных $\{(\boldsymbol{x}_{t_i}, t_i)\}$, и каждое её скрытое состояние $\boldsymbol{h}_{t_i}$ описывается формулой (рис. 1):

$$\boldsymbol{h}_i = \boldsymbol{g}(\boldsymbol{h}_{i-1}, \boldsymbol{x}_{t_i}).$$

Используя в виде функции активации гиперболический тангенс, положим $\boldsymbol{g}$:

$$\boldsymbol{g}(\boldsymbol{h}_{i-1}, \boldsymbol{x}_{t_i}) = \boldsymbol{th}(\boldsymbol{W}_{ih}\boldsymbol{x}_{t_i} + \boldsymbol{b}_{ih} + \\ + \boldsymbol{W}_{hh}\boldsymbol{h}_{i-1} + \boldsymbol{b}_{hh}), \tag{3}$$

где $\boldsymbol{W}_{ih}, \boldsymbol{W}_{hh}$ – соответствующие матрицы весов, а $\boldsymbol{b}_{ih}, \boldsymbol{b}_{hh}$ – соответствующие векторы смещения.

Можно заметить, что в промежутке времени $[t_{i-1}, t_i]$ скрытое состояние сети остается константным. Данная архитектура эффективна для наблюдений, которые получены через равные промежутки времени.

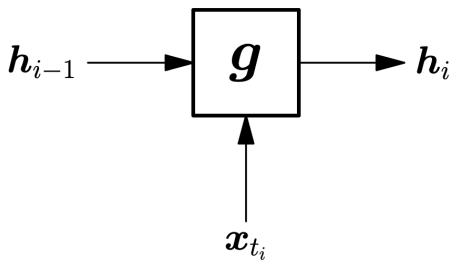

Рис. 1. Блок рекуррентной нейронной сети (RNNCell).
Fig. 1. RNNCell.

Нередко в реальных задачах промежутки времени между наблюдениями отличаются, и именно

это различие несет полезную информацию для обучения модели. Например: медицинские данные, промежутки времени получения которых могут отличаться значительно и при этом могут нести много полезной информации. Стандартная архитектура RNN не учитывает эту особенность.

В данной модели также используется НОДУ: $\boldsymbol{g}$ – нейронная сеть, которая параметризует динамику развития скрытой переменной во времени. Легко заметить, что полученная архитектура позволяет учесть интервалы получения данных, а также – находить скрытое состояние между промежутками получения данных, поскольку траектория состояния не является константной (вследствие того, что $\boldsymbol{h}'_i$ – решение НОДУ), как в случае с классической RNN. Этот подход позволяет увеличить обобщающую способность модели, но при этом для каждого шага обучения требуется намного больше как временных, так и вычислительных затрат, поскольку на каждой итерации обучения решается НОДУ.

2.2. ODE-RNN

Рассмотрим расширение [5] предыдущей архитектуры при помощи нейронных обыкновенных дифференциальных уравнений. Сравним различия архитектур на примере алгоритмов обучения каждой из сетей.

Алгоритм 1 Обучение ODE-RNN. Единственное отличие, выделенное рамкой, от стандартной рекуррентной сети – предобработка скрытого состояния.

Вход: Наблюдения и промежутки времени, в которые они получены $\{(\boldsymbol{x}_{t_i}, t_i)\}_{i=1..N}$

$\boldsymbol{h}_0 = \boldsymbol{0}$

для $i = 1, 2, \ldots, N$:

$\boxed{\boldsymbol{h}'_i = \text{ODESolve}((t_{i-1}, t_i), \boldsymbol{h}_{i-1}, \boldsymbol{\theta}, \boldsymbol{g})}$ ▷ ODESolve – численный метод решения ОДУ.

$\boldsymbol{h}_i = \text{RNNCell}(\boxed{\boldsymbol{h}'_i}, \boldsymbol{x}_i)$ ▷ RNNCell – рекуррентный блок, аналогичный рис. 1.

для $i = 1..N$:

$\boldsymbol{o}_i = \text{OutputNN}(\boldsymbol{h}_i)$ ▷ OutputNN – полносвязная нейросеть

Выход: $\{\boldsymbol{o}_i\}_{i=1..N}; \boldsymbol{h}_N$

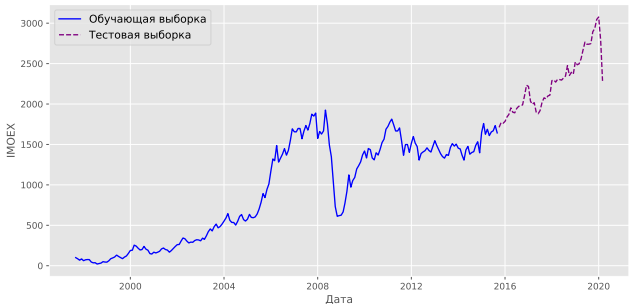

Рис. 2. Цена закрытия индекса Московской биржи с сентября 1997 г. по март 2020 г.
Fig. 2. The closing price of the Moscow Exchange index from September 1997 to March 2020.

2.3. Сравнение RNN и ODE-RNN

Далее ниже во всех экспериментах будут использоваться два набора данных: помесячные данные по ценам закрытия индекса Московской биржи (IMOEX, https://moex.com/ru/index/IMOEX/archive/) и среднемесячная температура в г. Москве (Kaggle, https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data (рис. 2–4).

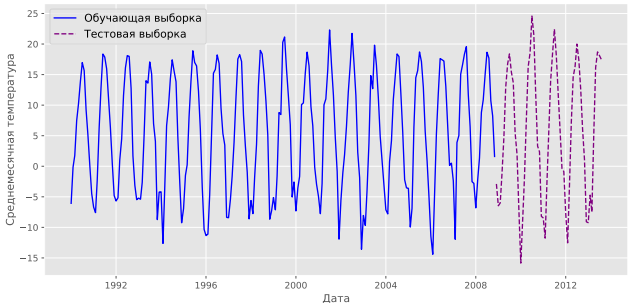

Рис. 3. Среднемесячная температура (°C) в г. Москве с января 1990 г. по август 2013 г.
Fig. 3. The average monthly temperature (°C) in Moscow from January 1990 to August 2013.

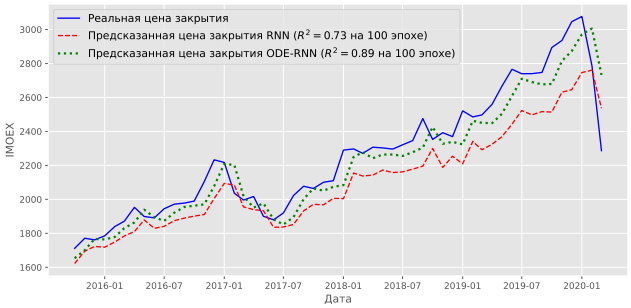

Рис. 4. Цена закрытия индекса Московской биржи (тестовая выборка).
Fig. 4. The closing price of the Moscow Exchange index (test data).

Для обучающей выборки использовалось $80\,\%$ наблюдений, а для тестовой – оставшиеся $20\,\%$. Для каждой модели применялась одна и та же базовая архитектура RNN. Все эксперименты были реализованы с использованием фреймворка глубокого обучения PyTorch и библиотеки (PyTorch Implementation of Differentiable ODE Solvers, https://github.com/rtqichen/torchdiffeq) численного решения НОДУ. В качестве функции $\boldsymbol{g}$ из алгоритма 1 использовалась полносвязная нейронная сеть с одним слоем.

Определение 2.1
Коэффициентом детерминации, или R^2, будем называть $R^2 = 1 - SS_{res}/SS_{tot}$, где $\boldsymbol{y}$ – вектор реальных значений, $\hat{\boldsymbol{y}}$ – вектор, предсказанных значений, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, $SS_{tot} = \sum_{i=1}^{n}(y_i - \bar{y})^2$, $SS_{res} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

Сравним модели RNN и ODE-RNN по графикам, представленным ниже (рис. 5). Можно заметить, что ODE-RNN лучше приближает реальные данные цен закрытия индекса IMOEX.

Аналогично, ODE-RNN лучше приближает реальные данные температуры в г. Москве, чем RNN.

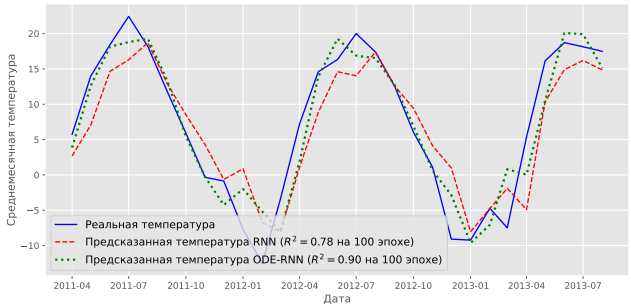

Рис. 5. Среднемесячная температура (°C) в г. Москве (тестовая выборка). Для наглядности показан период с апреля 2011 г. по август 2013 г.
Fig. 5. The average monthly temperature (°C) in Moscow (test sample). For clarity, the period from April 2011 to August 2013 is shown.

Определение 2.2
Эпоха (epoch) – одна полная (весь обучающий набор данных) итерация обучения модели.

Таблица 1. Среднее время, затраченное на обучение на 1 эпоху (цены закрытия индекса IMOEX)
Table 1. Average time spent on training for 1 epoch (closing price of IMOEX index)

Модель	RNN	ODE-RNN
Время	0.12 сек.	1.86 сек.

Таблица 2. Среднее время, затраченное на обучение на 1 эпоху (среднемесячная температура в г. Москве)
Table 2. Average time spent on training for 1 epoch (average monthly temperature (°C) in Moscow)

Модель	RNN	ODE-RNN
Время	0.18 сек.	2.64 сек.

Рассмотрим значение R^2 на каждой эпохе по рис. 6. Стоит отметить, что кривая R^2 для модели ODE-RNN круто растёт до 20 эпохи, а далее выходит на плато.

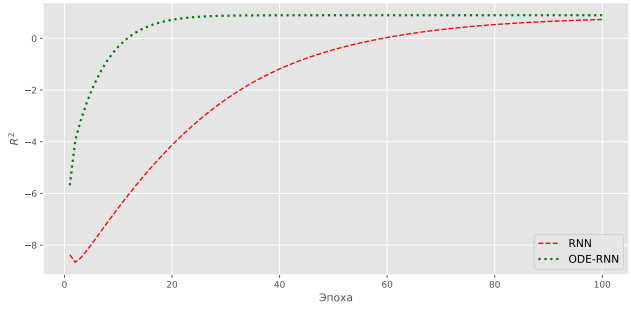

Рис. 6. R^2 моделей RNN и ODE-RNN (тестовая выборка, цены закрытия индекса IMOEX).
Fig. 6. R^2 of RNN and ODE-RNN (test data, closing price of IMOEX index).

На рис. 6 можно заметить, что разность между значением R^2 для ODE-RNN и RNN постепенно увеличивается до 20 эпохи, а далее ODE-RNN обучается уже не так быстро, и разрыв сокращается. Таким образом, к 100 эпохе R^2 у ODE-RNN больше на **0.16**, а время обучения больше на **174** сек. Стоит также отметить, что значения $R^2 = 0.72$ ODE-RNN достигает к 20 эпохе, а RNN – к 98, и затрачивает при этом на **25.44** сек. больше.

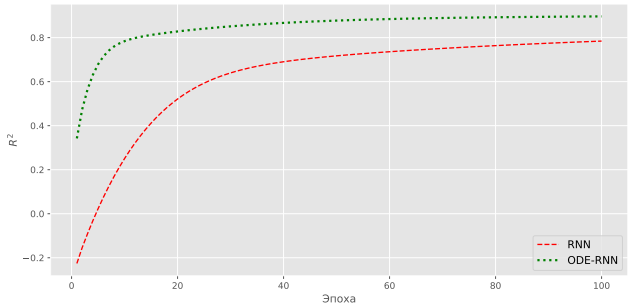

Рис. 7. R^2 моделей RNN и ODE-RNN (тестовая выборка, среднемесячная температура (°C) в г. Москве).
Fig. 7. R^2 of RNN and ODE-RNN (test data, average monthly temperature (°C) in Moscow).

На рис. 7 можно заметить, что разность между значением R^2 для ODE-RNN и RNN постепенно сокращается при довольно большой первоначальной разнице. Таким образом, к 100 эпохе R^2 у ODE-RNN больше на **0.12**, а время обучения – на **246** сек. Стоит также отметить, что значения $R^2 = 0.71$ ODE-RNN достигает к 6 эпохе, а RNN – к 48, и затрачивает при этом на **7.2** сек. больше.

3. p-ODE-RNN

3.1. Теоретическая часть

Для того, чтобы уменьшить вычислительные затраты каждой итерации обучения, предложим новую архитектуру **p-ODE-RNN**, для которой скрытое состояние $\boldsymbol{h}_i$ будет являться случайной величиной, принимающей значения RNNCell(ODESolve$((t_{i-1}, t_i), \boldsymbol{h}_{i-1}, \boldsymbol{\theta}, \boldsymbol{g}), \boldsymbol{x}_i)$ с вероятностью p и RNNCell$(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i)$ с вероятностью $1-p$.

Используя полученное вероятностное расширение скрытого состояния $\boldsymbol{h}_i$, получаем схему Бернулли, в которой «успехом» будет считаться предобработка $\boldsymbol{h}_{i-1}$.

Связь с геометрическим распределением. Пусть ν – случайная величина равная номеру первого «успеха» (обращения к численному методу решения НОДУ для предобработки скрытого состояния $\boldsymbol{h}_{i-1}$). Тогда $\nu \sim Geom_1(p)$ и, следовательно, $\mathbb{P}(\nu = k) = (1-p)^{k-1}p, \quad k = 1, 2, \ldots$

Связь с биномиальным распределением. Пусть α – количество «успехов» в n испытаниях Бернулли, описанных выше, с вероятностью успеха p, тогда $\alpha \sim Bin(n, p)$ и $\mathbb{P}(\alpha = k) = C_n^k p^k (1-p)^{n-k}$.

Связь с пуассоновским распределением. Возникает вопрос: как выбрать значение параметра p для получения наилучшей обобщающей способности модели при минимальных временных и вычислительных затратах?

Предложение 3.1
Рассмотрим $p = p(t)$ как функцию, зависящую от времени, причём выберём её так, чтобы $\lim\limits_{t\to\infty} p(t) = 0$, поскольку, исходя из рис. 6, 7, на первых эпохах ODE-RNN обучается быстрее чем RNN. Пусть $p(t) = 1/t$.

Используя это предложение, получаем удовлетворение всех условий теоремы Пуассона 1.

Теорема 1 (Пуассона)
Пусть $t \to \infty$ и $p(t) \to 0$ так, что $tp(t) \to \lambda > 0$. Тогда для любого $k \geqslant 0$ вероятность получить k успехов в t испытаниях схемы Бернулли с вероятностью успеха $p(t)$ стремится к величине $e^{-\lambda}\lambda^k/k!$.

Таким образом, получаем, что

$$\lim_{t\to\infty} tp(t) = \lim_{t\to\infty} t\frac{1}{t} = 1 = \lambda.$$

Скрытое состояние $\boldsymbol{h}_i$ как случайный процесс. Нетрудно заметить, что теперь можно рассматривать $\{\boldsymbol{h}_i(\omega), i = 0, 1 \dots N\}$ как случайный процесс. Таким образом, данный факт позволяет использовать весь аппарат теории случайных процессов в рамках скрытого состояния в архитектуре p-ODE-RNN.

Определение 3.1
Случайный процесс $\{\boldsymbol{h}_t(\omega), t = 0, 1 \dots N\}$ называется интегрируемым [6], если $E|\boldsymbol{h}_t| < \infty$ для всех $t \in [0, N]$.

Теорема 2
Случайный процесс $\{\boldsymbol{h}_i(\omega), i = 0, 1 \dots N\}$ является интегрируемым.

Доказательство. Пусть $\boldsymbol{a}$ – вектор такой же размерности, как и $\boldsymbol{h}_i$, состоящий только из единиц. Распишем первый абсолютный момент через формулу (3):

$$\mathbb{E}|\boldsymbol{h}_i| = p|\boldsymbol{th}(\boldsymbol{W}_{ih}\boldsymbol{x}_i + \boldsymbol{b}_{ih} +$$
$$+\boldsymbol{W}_{hh}\text{ODESolve}((t_{i-1}, t_i), \boldsymbol{h}_{i-1}, \boldsymbol{\theta}, f), \boldsymbol{x}_i) + \boldsymbol{b}_{hh})| +$$
$$+(1-p)|\boldsymbol{th}(\boldsymbol{W}_{ih}\boldsymbol{x}_i + \boldsymbol{b}_{ih} + \boldsymbol{W}_{hh}\boldsymbol{h}_{i-1} + \boldsymbol{b}_{hh})| \leqslant$$
$$\leqslant p\boldsymbol{a} + (1-p)\boldsymbol{a} \leqslant 1 \leqslant \infty.$$

В последнем неравенстве было использовано соотношение $|\text{th}(x)| \leqslant 1$. □

3.2. Практическая часть

В этом разделе будет использоваться параметр $p = p(t) = 1/t$.

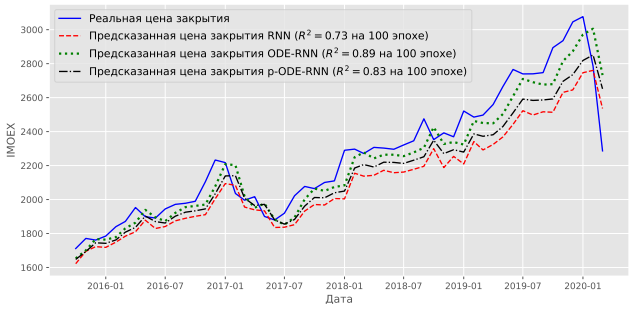

Рис. 8. Цена закрытия индекса Московской биржи (тестовая выборка).
Fig. 8. Closing price of IMOEX index (test data).

На рис. 8 представлено сравнение качества различных алгоритмов. Можно заметить, что кривая, сгенерированная p-ODE-RNN, находится между ODE-RNN и RNN.

Аналогично, можно заметить на рис. 9, что кривая, сгенерированная p-ODE-RNN, в случае другого датасета имеет схожее поведение.

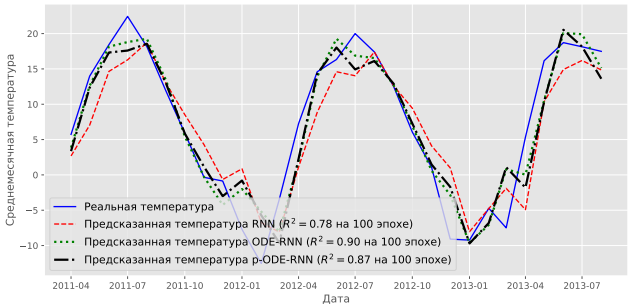

Рис. 9. Среднемесячная температура (°C) в г. Москве (тестовая выборка). Для наглядности показан период с апреля 2011 г. по август 2013 г.
Fig. 9. The average monthly temperature (°C) in Moscow (test data). For clarity, the period from April 2011 to August 2013 is shown.

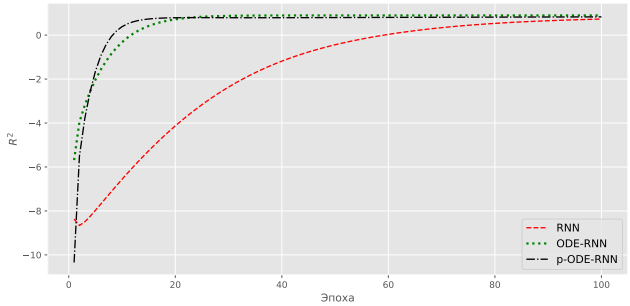

Рис. 10. R^2 моделей RNN, ODE-RNN, p-ODE-RNN (цены закрытия индекса IMOEX, тестовая выборка).
Fig. 10. R^2 of RNN, ODE-RNN, p-ODE-RNN (closing price of IMOEX index, test data).

На рис. 10 p-ODE-RNN быстрее всех моделей достигает уровня $R^2 = 0.77$ и далее обучается со скоростью, сопоставимой с ODE-RNN.

На рис. 11 кривая R^2 для p-ODE-RNN находится между RNN и ODE-RNN на протяжении всего обучения.

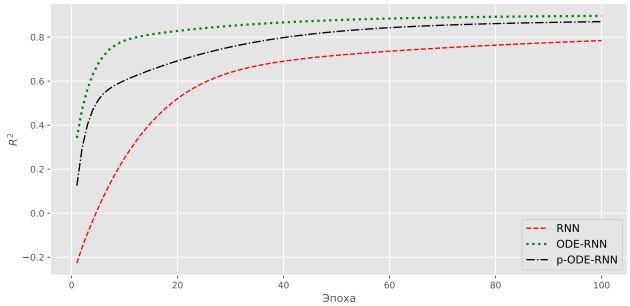

Рис. 11. R^2 моделей RNN, ODE-RNN, p-ODE-RNN (среднемесячная температура (°C) в г. Москве, тестовая выборка).
Fig. 11. R^2 of RNN, ODE-RNN, p-ODE-RNN (average monthly temperature (°C) in Moscow, test data).

Определение 3.2
NFE (number of function evaluations) – количество вычислений функции f (нейронной сети) при численном решении НОДУ.

Количество вычислений нейронной сети при численном решении НОДУ сопоставимо для каждого набора данных и при этом для p-ODE-RNN оно значительно меньше, поскольку $p(t)$ выбрано одним и

тем же и равно $1/t$. Этот факт сильно снижает время обучения p-ODE-RNN.

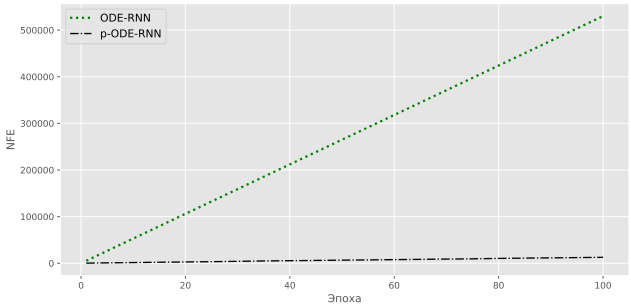

Рис. 12. Кумулятивное значение NFE для ODE-RNN и p-ODE-RNN (цены закрытия индекса IMOEX).
Fig. 12. Cumulative NFE value of ODE-RNN and p-ODE-RNN (closing price of IMOEX index).

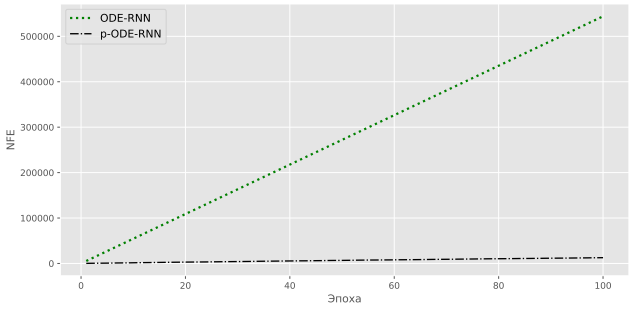

Рис. 13. Кумулятивное значение NFE для ODE-RNN и p-ODE-RNN (среднемесячная температура (°C) в г. Москве).
Fig. 13. Cumulative NFE value of ODE-RNN and p-ODE-RNN (average monthly temperature (°C) in Moscow).

Количество вычислений нейронной сети при численном решении НОДУ сопоставимо для каждого набора данных, и при этом для p-ODE-RNN оно значительно меньше, поскольку $p(t)$ выбрано одним и тем же и равно $1/t$. Этот факт сильно снижает время обучения p-ODE-RNN.

Таблица 3. Среднее время, затраченное на обучение на 1 эпоху (цены закрытия индекса IMOEX)
Table 3. Average time spent on training for 1 epoch (closing price of IMOEX index)

Модель	RNN	ODE-RNN	p-ODE-RNN
Время	0.12 сек.	1.86 сек.	0.20 сек.

Таким образом, на данных цен закрытия индекса IMOEX к 100 эпохе R^2 у p-ODE-RNN меньше на **0.06** (рис. 8), чем у ODE-RNN, а время обучения меньше на **166** сек. Получается, что p-ODE-RNN позволяет получать сопоставимую с ODE-RNN обобщающую способность, при этом затрачивая значительно меньшее время.

Таблица 4. Среднее время, затраченное на обучение на 1 эпоху (среднемесячная температура (°C) в г. Москве)
Table 4. Average time spent on training for 1 epoch (average monthly temperature (°C) in Moscow)

Модель	RNN	ODE-RNN	p-ODE-RNN
Время	0.18 сек.	2.64 сек.	0.28 сек.

Если же использовать набор данных о среднемесячной температуре в г. Москве, то к 100 эпохе R^2 у p-ODE-RNN меньше на **0.03** (рис. 9), чем у ODE-RNN, а время обучения меньше на **236** сек. Аналогично предыдущему набору данных в этом случае обобщающая способность, получаемая при помощи p-ODE-RNN, сопоставима с ODE-RNN при значительно меньших временных затратах.

Из вышеописанных сравнений следует, что нейросетевая архитектура p-ODE-RNN позволяет получать схожее с ODE-RNN и лучше, чем у RNN, качество, затрачивая при этом значительно меньшие временные ресурсы, нежели ODE-RNN.

Литература–References

1. *Hornik K.* Approximation Capabilities of Multilayer Feedforward Networks // Neural Networks. 1991. Vol. 4. P. 251–257.
2. Deep Residual Learning for Image Recognition / *K. He, X. Zhang, S. Ren, J. Sun* // The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 770–778.
3. Neural Ordinary Differential Equations / *R.T.Q. Chen, Yu. Rubanova, J. Bettencourt, D. Duvenaud* // Advances in Neural Information Processing Systems. 2018. Vol. 31. P. 6571–6583.
4. Математическая теория оптимальных процессов / *Л.С. Понтрягин, В.Г. Болтянский, Р.В. Гамкрелидзе, Е.Ф. Мищенко.* М.: Наука, 1983. 392 с.
   Matematicheskaja teorija optimal'nyh processov [Mathematical theory of optimal processes] / *L.S. Pontryagin, V.G. Boltyansky, R.V. Gamkrelidze, E.F. Mishchenko.* Moscow: Nauka, 1983. 392 p.
5. *Rubanova Yu., Chen R.T.Q., Duvenaud D.* Latent ODEs for Irregularly-Sampled Time Series // Advances in Neural Information Processing Systems. 2019. Vol. 32. P. 5320–5330.
6. *Круглов В.М.* Случайные процессы. М.: Юрайт, 2016. 280 с.
   Kruglov V.M. Sluchajnye processy [Random processes]. Moscow: Yurait, 2016. 280 p.

Статья поступила в редакцию 08.11.2021.