УДК 517.9:004.8

ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ КАК МАТЕМАТИЧЕСКАЯ ОСНОВА НЕЙРОННЫХ СЕТЕЙ НЕПРЕРЫВНОГО ВРЕМЕНИ

Богуш Р.Д.; Телипко Д.А.; Доценко Е.В., студенты

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь.

Лущакова И.Н. – канд. физ.-мат. наук, доцент

Аннотация. В работе рассматривается класс рекуррентных нейронных сетей с непрерывным временем – нейронные сети с текучей временной константой, в которых динамика скрытых состояний описывается системой обыкновенных дифференциальных уравнений (ОДУ). В отличие от традиционных подходов, константа времени таких сетей не является фиксированной, а изменяется в зависимости от входных данных, что обеспечивает повышенную выразительность и стабильность модели. Описываются математические основы модели, метод численного решения, сферы применения и дальнейшие перспективы ее использования. В данной работе представлены результаты экспериментального сравнения двух моделей (ДКП и ТКВ), выполненного с использованием языка программирования Python. Полученные результаты демонстрируют превосходство архитектуры ТКВ над ДКП.

Ключевые слова. Обыкновенные дифференциальные уравнения (ОДУ), нейронные сети непрерывного времени, рекуррентные нейронные сети (РНС), нейронные обыкновенные дифференциальные уравнения (нейронные ОДУ), текучая константа времени (ТКВ), нейронные сети с текучей временной константой, долгая краткосрочная память (ДКП).

1. Введение. Нейронные сети с непрерывным временем, основанные на обыкновенных дифференциальных уравнениях, представляют собой перспективное направление в области глубокого обучения. Они позволяют моделировать временные зависимости в данных, которые имеют нерегулярную природу – например, медицинские показатели, промышленные датчики или финансовые временные ряды.

Классические рекуррентные нейронные сети (РНС), например, ДКП работают в дискретном времени и не способны естественным образом описывать непрерывную динамику систем. Для решения проблемы с описанием непрерывных событий были предложены нейронные ОДУ [1]. Данный способ устранил этот недостаток, однако обладал ограниченной выразительностью. В работе [2] предложена модель ТКВ, которая расширяет семейство нейронных ОДУ за счёт введения входозависимой константы времени.

2. Математическая модель ТКВ. Классические РНС имеют дискретную среду и описываются следующим уравнением:

$$h_{n+1} = h_n + f(h_n, \Theta), \tag{1}$$

где h_n – скрытое состояние сети РНС; h_{n+1} – скрытое состояние сети РНС на следующем шаге; $f(h_n, \Theta)$ – нейросеть, вычисляющая поправку скрытого состояния на каждом шаге; Θ – обучаемые веса нейросети f.

Уравнение (1) представляет собой базовое уравнение РНС. В данное уравнение не входит переменная времени, следовательно, такая нейронная сеть не может качественно реагировать на непрерывные события.

Для решения описанной проблемы, от базового уравнения РНС (1) переходят к дифференциальному уравнению.

Для этого уравнение (1) запишем в следующем виде:

$$h_{n+1} - h_n = f(h_n, \Theta)$$

До этого между шагами проходила одна единица времени, после она была заменена на Δt – реальный промежуток времени между шагами. Тогда формула примет вид:

$$\frac{h_{n+1} - h_n}{\Delta t} = \frac{f(h_n, \Theta)}{\Delta t}$$

При $\Delta t \to 0$ левая часть $\frac{h_{n+1}-h_n}{\Delta t}$ становится равной производной от h по переменной t, а правая часть превращается в совершенно новую функцию. Тогда новое уравнение примет вид дифференциального уравнения:

$$\frac{dh}{dt} = f(h(t), I(t), \Theta)$$

Однако нейроны имеют свойство затухания их состояния, поэтому для сохранения этого свойства к уравнению добавим затухание, зависящее от константы времени τ *:*

$$\frac{dh}{dt} = -\frac{h(t)}{\tau} + f(h(t), I(t), \Theta), \quad (2)$$

где $h(t)$ – скрытое состояние сети в момент времени *t;* $\frac{dh}{dt}$ – скорость изменения скрытого состояния в момент времени *t;* τ – константа времени, определяющая скорость затухания; $I(t)$ – входной сигнал в момент времени *t; t* – непрерывное время*;* Θ – обучаемые веса нейросети *f;* $f(h(t), I(t), \Theta)$ – нейронная сеть, вычисляющая поправку скрытого состояния на основе текущего скрытого состояния*,* входного сигнала и обучаемых весов.

Уравнение (2) представляет собой нейронное ОДУ (Neural ODE).

В уравнении учитывается константа времени, однако она является фиксированной, что ограничивает её выразительную способность. Реальные биологические нейронные сети демонстрируют широкий спектр временных масштабов и способны менять скорость реакции нейронов. Поэтому имеет смысл рассмотреть усовершенствованную версию нейронных ОДУ – так называемую модель ТКВ.

Если в ОДУ (2) заменить нелинейный член $f(h(t), I(t), \Theta)$ на специальную структуру вида $S(t) = f(h(t), I(t), \Theta) \cdot (A - h(t))$, где *A* – это константный вектор, который представляет точку равновесия, к которой стремится система, а *h(t)* тоже представляет собой вектор с запоминаемой информацией в момент времени *t*. Тогда подставим $S(t)$ в уравнение (2) вместо $f(h(t), I(t), \Theta)$ и раскроем скобки:

$$\frac{dh}{dt} = -\frac{h(t)}{\tau} + f(h(t), I(t), \Theta) \cdot (A - h(t)) \quad (3)$$

$$\frac{dh}{dt} = -\frac{h(t)}{\tau} + f(h(t), I(t), \Theta) \cdot A - f(h(t), I(t), \Theta) \cdot h(t)$$

$$\frac{dh}{dt} = -h(t) \cdot \left(\frac{1}{\tau} + f(h(t), I(t), \Theta)\right) + f(h(t), I(t), \Theta) \cdot A\,, \quad (4)$$

В уравнении (2) коэффициент затухания $-\frac{1}{\tau}$, где τ – это независимая константа, теперь же коэффициент затухания зависит ещё и от $f(h(t), I(t), \Theta)$. Обозначим этот коэффициент затухания τ_{sys}

$$\frac{1}{\tau_{sys}} = \left(\frac{1}{\tau} + f(h(t), I(t), \Theta)\right) = \left(\frac{1+\tau \cdot f(h(t), I(t), \Theta)}{\tau}\right),$$

$$\tau_{sys} = \left(\frac{\tau}{1 + \tau \cdot f(h(t), I(t), \Theta)}\right)$$

Тогда уравнение (4) примет вид:

$$\frac{dh}{dt} = -\frac{h(t)}{\tau_{sys}} + f(h(t), I(t), \Theta) \cdot A \quad (5)$$

Уравнение (5) представляет собой идею ТКВ. Ключевым отличием данной формулировки является то, что эффективная константа времени системы τ_{sys} теперь зависит от входных данных.

Это свойство и дало название модели – «жидкая» (liquid). Каждый нейрон сети адаптирует свою динамику под текущие входные данные, что позволяет сети выделять специализированные динамические режимы для различных входных признаков.

3. Биологическая мотивация LTC. Идея модели LTC изначально была основана на наблюдениях за биологически мотивированной динамикой несинаптических нейронов малых организмов, в

частности нематоды *C. Elegans*, у которой всего 302 нейрона, но она способна ориентироваться в пространстве, находить пищу и избегать опасных веществ. Несмотря на минимальное количество нейронов, червь демонстрирует температурную память и даже элементы социального поведения. Строение нейронной системы нематоды представлено на рисунке 1.

Внутри этих организмов не происходит дискретных вычислений, вместо этого нейроны в реальном времени могут изменять свои свойства и реагировать на различные входные данные.

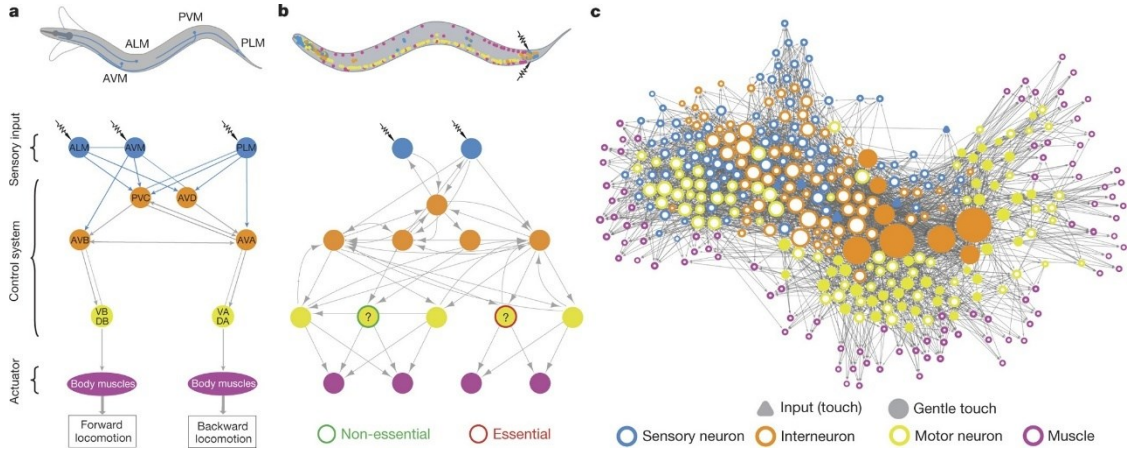

Рисунок 1. Нейронная система нематоды *C. Elegans*

Таким образом, для описания такого поведения математически необходимо представить его с помощью дифференциальных уравнений. Потенциал нейрона *v(t)* описывается уравнением:

$$\frac{dv}{dt} = -g_l \cdot v(t) + S(t)\ , \tag{6}$$

где $v(t)$ – потенциал мембраны нейрона; $\frac{dv}{dt}$ – скорость изменения потенциала; g_l – проводимость утечки; $S(t)$ – суммарный синаптический ток.

$$\boldsymbol{S(t) = f(v(t), I(t), \Theta) \cdot (A - v(t))}\ , \tag{7}$$

где $f(v(t), I(t), \Theta)$ – сила синаптической связи; $(A - v(t))$ – разница потенциалов между точкой равновесия и текущим состоянием; $I(t)$ – входной сигнал.

Раскрывая скобки, получаем следующее уравнение:

$$\frac{dv}{dt} = -v(t) \cdot \Big(g_l + f\big(v(t), I(t)\big)\Big) + f\big(v(t), I(t)\big) \cdot A \tag{8}$$

Нетрудно заметить, что уравнения (3) и (8) одинаковы. В них g_l и $\frac{1}{\tau}$ являются идентичными величинами.

4. Численное решение ОДУ: метод слитного решателя. При решении ОДУ учитываются два ключевых фактора:

- устойчивость – свойство численного метода не усиливать ошибки при интегрировании.
- эффективность – соотношение между точностью результата и вычислительными затратами на его получение.

ОДУ, описывающее динамику ТКВ, является “жёстким”, то есть решение его явными численными методами (например, методом Рунге-Кутты) является неудовлетворительным из-за резкого увеличения числа вычислений (при малом шаге интегрирования) или из-за резкого возрастания погрешности.

Поэтому необходимо использовать более комплексный подход. Например в статье [2] предложен специальный слитный решатель (Fused ODE Solver), объединяющий явный и неявный методы Эйлера. Идея состоит в том, чтобы нелинейные части $(-\frac{h(t)}{\tau_{sys}})$ вычислять в точке $h(t_i)$ (как в явном методе), а линейные – в точке $h(t_{i+1})$ (как в неявном). Аналитическое решение полученного уравнения даёт формулу обновления:

$$h(t + \Delta t) = \frac{h(t) + \Delta t \cdot f(x(t), I(t), t, \Theta) \cdot A}{1 + \Delta t \cdot (\frac{1}{\tau} + f(x(t), I(t), t, \Theta))} \quad (9)$$

Данная формула сочетает вычислительную эффективность явного метода Эйлера с устойчивостью неявного. Вычислительная сложность алгоритма составляет O(L×T), где L – число шагов дискретизации, T – длина входной последовательности.

5. Свойства модели: стабильность и ограниченность. Одним из важных теоретических результатов является доказательство ограниченности как константы времени, так и скрытых состояний нейронов LTC, то есть доказательство того, что они остаются в конечном диапазоне при любых входных данных и не уходят в ноль или в бесконечность.

Теорема 1 [2] (ограниченность константы времени). При использовании ограниченной монотонно возрастающей сигмоидальной функции f эффективная константа времени τ_{sys} нейрона i ограничена диапазоном:

$$\frac{\tau_i}{1+\tau_i \cdot W_i} \leq \tau_{sys} \leq \tau_i \ , \quad (10)$$

где τ_i – базовая константа времени *i*-ого нейрона; τ_{sys} – изменяемая константа времени нейрона; W_i – это максимально возможное значение функции *f* для нейрона *i, i* – номер нейрона.

Из формулы $\tau_{sys} = \left(\frac{\tau_i}{1+\tau_i \cdot f(h(t), I(t), \Theta)}\right)$ следует:

если f$(h(t), I(t), \Theta)$=0 (нейрон не получает значимого входа), то знаменатель минимален, τ_{sys}=τ_i – это верхняя граница;

если f$(h(t), I(t), \Theta) = W_i$ (нейрон насыщен), то знаменатель максимален, $\tau_{sys} = \left(\frac{\tau_i}{1+\tau_i \cdot W_i}\right)$ – это нижняя граница.

Теорема 2 [2] (ограниченность скрытых состояний). Скрытое состояние любого нейрона *i* на конечном интервале *[0, T]* удовлетворяет неравенству:

$$\left(0, A_i^{min}\right) \leq h_i(t) \leq max(0, A_i^{max}) \ , \quad (11)$$

где A_i^{min} и A_i^{max} – минимальное и максимальное значения вектора равновесия *A* из формулы (3) для нейрона *i*.

Данное свойство, называемое стабильностью состояний, гарантирует, что выходы ТКВ никогда не «взрываются», даже если входные данные неограниченно возрастают. Это делает ТКВ пригодным для применения в системах реального времени.

6. Экспериментальные результаты. Далее представлены результаты проведённого нами эксперимента. В этом эксперименте были обучены две модели разных архитектур (ТКВ(LTC) и ДКП(LSTM) и выполнены сравнения результатов предсказаний двух нейросетей. В эксперименте используется набор данных дорожного трафика, взятый с сайта Kaggle.com [3].

Для начала были подключены основные библиотеки языка Python для работы с данными и моделями машинного обучения. Также был загружен основной файл с входными данными для обучения моделей и были обработаны признаки этих данных, в результате чего осталось только 3 обучаемых признака.

```
sns.set_theme(style="darkgrid")

# Загрузка файла с данными

df = pd.read_csv('Metro_Interstate_Traffic_Volume.csv')

# Обработка признаков

features = ['temp', 'clouds_all', 'traffic_volume']
data = df[features].values

scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
```

После этого данные были разбиты на тренировочную и тестовую выборку для предотвращения переобучения моделей.

```
def create_sequences(data, seq_length):  1 usage
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:(i + seq_length)])
        y.append(data[i + seq_length, 2])
    return np.array(X), np.array(y)

SEQ_LENGTH = 24
X, y = create_sequences(data_scaled, SEQ_LENGTH)

split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

При создании моделей мы указываем, что обе модели будут обучаться с использованием оптимизатора Adam, а в качестве функции потерь было выбрано среднеквадратичное отклонение(MSE). Модели импортируем из библиотек Tensorflow и ncps.

```
# Создание моделей
lstm_model = Sequential([
    Input(shape=(SEQ_LENGTH, X_train.shape[2])),
    LSTM( units: 32, activation='tanh'),
    Dense(1)
])
lstm_model.compile(optimizer='adam', loss='mse')

wiring = AutoNCP( units: 32,  output_size: 1)
ltc_model = Sequential([
    Input(shape=(SEQ_LENGTH, X_train.shape[2])),
    TimeDistributed(Dense( units: 16, activation='tanh')),
    LTC(wiring, return_sequences=False)
])
ltc_model.compile(optimizer='adam', loss='mse')
```

После этого мы обучаем наши модели на уже обработанном наборе данных и проводим прогнозирование значений

```
# Обучение моделей LSTM и LTC

EPOCHS = 10
BATCH_SIZE = 64


lstm_model.fit(X_train, y_train, epochs=EPOCHS, batch_size=BATCH_SIZE, validation_split=0.1, verbose=1)

print("\n--- Обучение LTC ---")
ltc_model.fit(X_train, y_train, epochs=EPOCHS, batch_size=BATCH_SIZE, validation_split=0.1, verbose=1)

# Прогнозирование значений
lstm_predictions = lstm_model.predict(X_test)
ltc_predictions = ltc_model.predict(X_test)

def get_final_signal(data, window=3):  1 usage
    return np.convolve(data, np.ones(window)/window, mode='same')

# Прогнозирование значений
lstm_predictions = lstm_model.predict(X_test)
ltc_predictions = ltc_model.predict(X_test)

def get_final_signal(data, window=3):  1 usage
    return np.convolve(data, np.ones(window)/window, mode='same')
```

В результате были получены следующие результаты, показанные на рисунке 2.

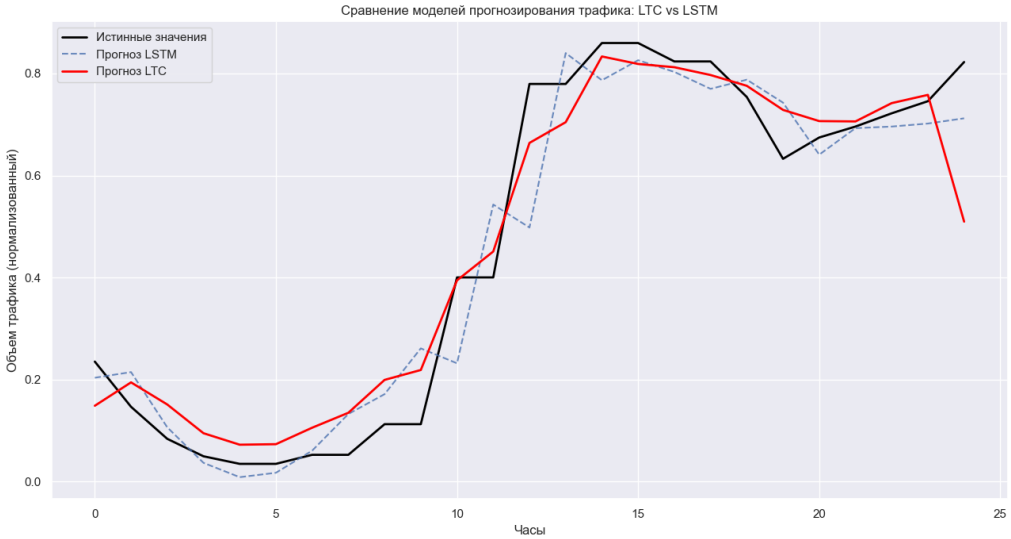

Рисунок 2. График точности предсказаний моделей LSTM и LTC

По данному графику можно увидеть, что LTC имеет более точные значения в местах взлетов и падений. Также видно, что график прогноза LTC имеет более “гладкую” форму, что обусловлено непрерывностью времени, которая заложена в нейронной сети LTC. По данным результатам можно сделать вывод, что модель LTC имеет более точные значения в случаях непрерывных поступлений данных.

7. Сферы применения и ограничения модели. ТКВ применяются в различных направлениях, где необходимо работать с данными в настоящем времени. Одно из наиболее распространенных применений заключается в управлении автономными транспортными средствами, например автомобилями и беспилотными летательными аппаратами. Это является хорошим примером использования ТКВ, так как они хорошо справляются с преобразованием шумов из входных данных, а также используют значительно меньше нейронов, что делает их более компактными, чем модели-конкуренты. Также одним из частых применений данного вида рекуррентных нейронных сетей является прогнозирование временных рядов, например в областях финансов или медицины.

Несмотря на преимущества, ТКВ имеет ряд ограничений. Во-первых, модель подвержена проблеме затухания градиентов при обучении на длинных последовательностях, что затрудняет

моделирование долгосрочных зависимостей. Во-вторых, производительность сильно зависит от выбора численного решателя ОДУ – использование простого явного метода Эйлера вместо рекомендованного слитного решателя заметно снижает качество. В-третьих, ТКВ требует значительно больше памяти по сравнению с нейронными ОДУ, что может стать ограничением при работе с большими моделями.

Дальнейшие направления исследований включают преодоление проблемы затухания градиентов для моделирования долгосрочных зависимостей, снижение вычислительных затрат, а также изучение причинно-следственных структур в динамике ТКВ.

Список использованных источников:

1. Chen T.Q., Rubanova Y., Bettencourt J., Duvenaud D.K. Neural Ordinary Differential Equations // Advances in Neural Information Processing Systems. — 2018. — С. 6571–6583.

2. Hasani R., Lechner M., Amini A., Rus D., Grosu R. Liquid Time-constant Networks // Proceedings of the AAAI Conference on Artificial Intelligence. — 2021.

3. https://www.kaggle.com/datasets/anshtanwar/metro-interstate-traffic-volume.

UDC 517.9:004.8

ORDINARY DIFFERENTIAL EQUATIONS AS THE MATHEMATICAL FOUNDSTION OF CONTINUOUS-TIME NEURAL NETWORKS

Bogush R.D.,Telipko D.A., Docenko E.V. students

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Lushakova I.N. – PhD in Physics and Mathematics, Associate Professor

Annotation. This paper examines a class of continuous-time recurrent neural networks known as Liquid Time-Constant Networks (LTC), in which the dynamics of hidden states are described by a system of ordinary differential equations (ODEs). Unlike traditional approaches, the time constant of such networks is not fixed but varies depending on the input data, which provides enhanced expressiveness and model stability. The mathematical foundations of the model, the numerical solution method, areas of application, and prospects of the model are described. We present the results of an experimental comparison of two models (LSTM and LTC) conducted using the Python programming language. The obtained results demonstrate the superiority of the LTC architecture over LSTM.

Keywords. Ordinary differential equations (ODE), continuous-time neural networks, recurrent neural networks (RNN), neural ordinary differential equations (neural ODEs), liquid time-constant (LTC), liquid time-constant neural networks, long short-term memory (LSTM).