УДК 681.51:004.94 Научная статья

https://doi.org/10.32603/2071-8985-2024-17-7-44-50

Применение нейронной сети в логическом управлении электромеханической системой

Д. В. Чернышев

Дальневосточный государственный университет путей сообщения, Хабаровск, Россия

fersy@mail.ru

Аннотация. В некоторых системах управления и автоматики логические преобразования на базе уравнений алгебры логики выполняются на основе программного обеспечения или посредством электрических схем, собранных на логических элементах. В публикации представлен вариант программной реализации логического управления как результат работы нейронной сети. В качестве примера управление определяется как преобразование двоичного множества одной размерности во множество другой размерности согласно заданной логической функции. Преобразование основано на использовании искусственной нейронной сети прямого распространения сигнала. При обучении нейронной сети применяется алгоритм обратного распространения ошибки. В качестве обучающих данных используются множество двоичных чисел как входные и выходные векторы сети и логическая функция, устанавливающая взаимосвязь между ними. Нейронная сеть и алгоритм обучения созданы на языке программирования Python с применением стандартных математических пакетов. После обучения нейронной сети с заданной степенью точности выполнялось переключение в рабочее состояние и на вход сети подавался входной вектор, состоящий из данных множества двоичных чисел. В другом варианте координаты входного вектора из множества двоичных чисел случайно изменялись как по уровню сигнала, так и по длительности в течение периода работы. Представлены результаты работы нейронной сети в реализации управления при отсутствии и наличии помех во входном векторе сети. Разработанный программный код может применяться в стандартных промышленных контроллерах как для предложенного преобразования, так и для других логических преобразований при изменении структуры нейронной сети и применении новых данных обучения.

Ключевые слова: система управления, математическое моделирование, язык программирования Python, нейронная сеть, алгоритм обучения, логические переменные

Для цитирования: Чернышев Д. В. Применение нейронной сети в логическом управлении электромеханической системой // Изв. СПбГЭТУ «ЛЭТИ». 2024. Т. 17, № 7. С. 44–50. doi: 10.32603/2071-8985-2024-17-7-44-50.

Original article

Application of a Neural Network in the Logical Control of an Electromechanical System

D. V. Chernyshev

Far Eastern State Transport University, Khabarovsk, Russia

fersy@mail.ru

Abstract. In some control and automation systems, logical transformations based on logic algebra equations are performed using software or through electrical circuits assembled on logical elements. This article presents a version of the software implementation of logical control as a result of the operation of a neural network. As an example, control is defined as the transformation of a binary set of one dimension into a set of another dimension, according to a given logical function. The conversion is based on the use of an artificial neural network of feedforward signal propagation. When training a neural network, a backpropagation algorithm is used.

The training data uses a set of binary numbers as input and output vectors of the network and a logical function that establishes the relationship between them. The neural network and learning algorithm were created in the Python programming language using standard mathematical packages. After training the neural network with a given degree of accuracy, it is switched to the operating state and an input vector was supplied to the network input. The input vector consisted of a given set of binary numbers. In another variant, the coordinates of the input vector from a set of binary numbers randomly changed both in signal magnitude and duration during the operation period. The results of the operation of a neural network in implementing control in the absence and presence of interference in the input vector of the network are presented. The developed program code can be used in standard industrial controllers, both for the proposed transformation and for other logical transformations when changing the structure of the neural network and applying new training data.

Keywords: control system, math modeling, Python programming language, neural network, learning algorithm, logical variables

For citation: Chernyshev D. V. Application of a Neural Network in the Logical Control of an Electromechanical System // LETI Transactions on Electrical Engineering & Computer Science. 2024. Vol. 17, no. 7. P. 44–50. doi: 10.32603/2071-8985-2024-17-7-44-50.

Введение. Во многих системах управления, основанных на знаниях, используются современные информационные технологии, – например, нечеткая логика или искусственные нейронные сети. Применение известных архитектур нейронных сетей и алгоритмов обучения зависит от решаемой задачи и методов практической реализации. Нейронные сети позволяют получать модели систем управления, реализовывать регуляторы сложными системами и выполнять преобразования данных.

Логику работы некоторых систем автоматики и управления электромеханическими объектами на основе логических функций реализуют как программными, так и аппаратными средствами, например составляют из стандартных логических элементов, реле, программируемых контроллеров. При этом современные методы искусственного интеллекта – такие, как нейронные сети, позволяют реализовывать логические функции посредством самообучения на заданных множествах. В публикациях [1]–[3] приведены примеры использования нейронной сети при работе с логическими переменными в решении технических и логических задач.

В решении таких задач широко применяется язык программирования Python и различные библиотеки. Известные современные программные продукты для работы с нейронными сетями TensorFlow и PyTorch – используются для различных приложений и во многом универсальны. Эти продукты используют теорию графов и математический аппарат тензорного вычисления. Каждое из этих приложений имеет определенные особенности как в создании моделей систем искусственного интеллекта, так и в работе с ними, например PyTorch использует динамические графы вычислений. Для удобства пользователя создаются дополнительные приложения. Приложение Keras работает с TensorFlow и определяет высокоуровневые нейронные сети [4], [5]. Эти модульные программные продукты предъявляют определенные требования к памяти и производительности устройств, на которых они применяются.

Предлагается разработать программную реализацию нейронной сети, как вариант для логического управления электромеханическими системами с возможностью самообучения. При этом используются архитектура нейронной сети и алгоритм обучения, необходимые для решения этой задачи, написанные на языке Python с применением встроенных математических пакетов.

Логические элементы систем управления. При реализации автоматизации на основе логических элементов управление определяют как функцию, которая зависит от входных логических переменных, и посредством этой логической функции определяют выходное значение управления. При реализации логической функции учитывают алгоритм работы и минимальное количество логических элементов, составляя ее на основе алгебры логики. На рис. 1 представлен логический элемент с входными значениями X, внутренними переменными Q и выходными управляющими воздействиями Y на выходе в общем виде.

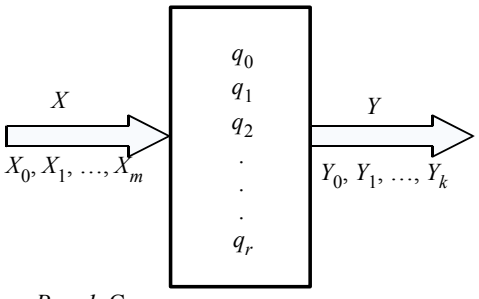

Рис. 1. Структура логического элемента
Fig. 1. Logic element structure

При этом все переменные логического элемента принимают значения 0 или 1. Размерность входного вектора – m, количество переменных логического элемента – r, размерность выходного вектора – k, и он применяется как управление. Размерности векторов входа и выхода определяются задачей управления.

На рис. 2 показан вариант включения логического элемента в систему управления.

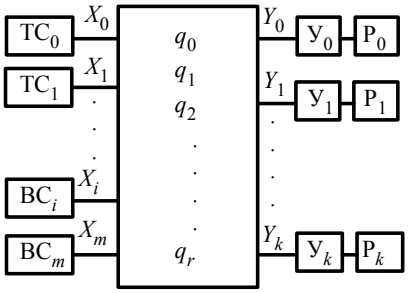

Рис. 2. Включение логического элемента в систему управления
Fig. 2. Incorporating a logic element into a control system

К входным переменным относятся технологические сигналы ТС, принимающие значения 0 или 1, и временные сигналы ВС со значениями 0 или 1. На выходе логического элемента подключены усилители логического сигнала У и силовые реле Р, используемые в управлении электромеханической системой. Вместо реле в схеме могут быть использованы силовые ключи.

В качестве примера использования логических элементов в преобразовании сигналов выполним преобразование с размерностью входного сигнала X, равной 3 ($m = 3$), по уравнениям логики преобразования:

$$\begin{cases} Y_0 = \overline{X_2} \wedge \overline{X_1} \wedge \overline{X_0}; & Y_1 = \overline{X_2} \wedge \overline{X_1} \wedge X_0; \\ Y_2 = \overline{X_2} \wedge X_1 \wedge \overline{X_0}; & Y_3 = \overline{X_2} \wedge X_1 \wedge X_0; \\ Y_4 = X_2 \wedge \overline{X_1} \wedge \overline{X_0}; & Y_5 = X_2 \wedge \overline{X_1} \wedge X_0; \\ Y_6 = X_2 \wedge X_1 \wedge \overline{X_0}; & Y_7 = X_2 \wedge X_1 \wedge X_0, \end{cases} \quad (1)$$

где X_i – значение i-го разряда входного двоичного вектора; Y_i – значение i-го разряда выходного двоичного вектора.

В системах автоматики и управления для преобразования (1) применяют схемы на логических элементах (рис. 3).

При включении на выход схемы силовых реле (рис. 3) можно управлять силовыми сигналами электромеханической системы в зависимости от комбинации входных логических сигналов. Технические решения на основе логических элементов

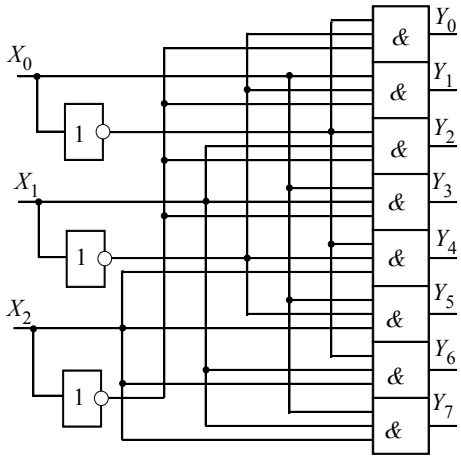

Рис. 3. Схема преобразования данных на логических элементах
Fig. 3. Data conversion circuit using logical elements

могут использоваться, как вариант, вместо релейных схем или с ними комбинируются в системах управления. Подобные логические схемы составляются на основе логических уравнений, которые посредством теорем алгебры логики преобразуют к наиболее простому виду с минимальным количеством элементов.

Структура и алгоритм обучения нейронной сети. Применение нейронных сетей в составлении логических функций основано на возможности получения гиперплоскости, разделяющей множество пространства решений на требуемые области. В частности, трехслойная сеть с нелинейными функциями активации формирует требуемое решение. Основы теории по применению нейронных сетей в аппроксимации функций изложены в [6]–[9].

Логический элемент при моделировании его работы нейронной сетью показан на рис. 4, где переменные X и Y определяются как векторные величины и задаются множествами входа и выхода соответственно.

Многослойные нейронные сети прямого распространения сигнала представляют один из вариантов нейронных сетей и могут использоваться в системах управления для преобразования информации и получения требуемой разделяющей гиперплоскости. В случае программной реализации логических функций нейронной сетью, в процессе самообучения можно реализовывать требуемые логические и функциональные зависимости. Алгоритмы самообучения таких нейронных сетей позволяют достигать требуемой точности [6]–[9].

Если изменения значений входных логических сигналов лежат в определенных диапазонах, то нейронная сеть выполняет преобразование. При изменении логики работы автоматической системы управления другие логические функции реализуются посредством обучения сети на новых данных. Структура нейронной сети для выполнения системы логических уравнений (1) представлена на рис. 4.

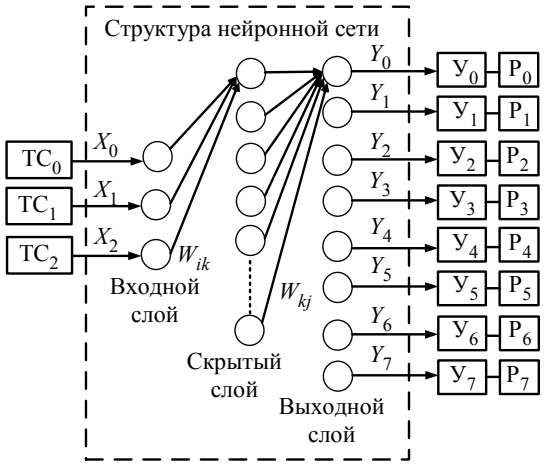

Рис. 4. Структура нейронной сети
Fig. 4. Neural network structure

На рис. 4 обозначены весовые коэффициенты нейронной сети от входного слоя к скрытому как W_{ik}, весовые коэффициенты нейронной сети от скрытого к выходному слою – как W_{kj}. В качестве активационной функции f в скрытом и выходном слоях использовалась логистическая.

Уравнением (1) определяется архитектура сети. Во входном слое 3 входных распределительных элемента, количество вычислительных элементов: в скрытом слое – 24, в выходном слое – 8. Параметры сети могут изменяться программно при решении других логических задач.

При обучении сети применялся алгоритм обратного распространения [6]–[9]. Нейронная сеть и алгоритм обучения, а также данные обучения и работы реализованы на языке программирования Python [10]–[12] с применением стандартных математических библиотек и пакета научных вычислений NumPy. Графический материал создан на основе пакета MatPlotLib.

Перед обучением сети составлялась обучающая выборка из входных и выходных данных, которая состоит из множества (X, T) где X – входной вектор сети в двоичном коде, а T – желаемое значение выходного вектора Y в двоичном коде. Обучающая выборка состоит из входного множества векторов с размерностью [3 × 1], элементы которых принимают значения 0 или 1, и выходного множества векторов с размерностью [8 × 1], элементы которых принимают значения 0 или 1. На основе пакета Random случайным образом выбиралась обучающая пара (X, T), которая подавалась на вход сети, и выполнялось обучение методом обратного распространения как изменение матрицы W весов нейронной сети, так чтобы суммарная квадратичная ошибка выхода сети была равной или меньше заданной. Суммарная квадратичная ошибка сети E определяется как

$$E = \frac{1}{2}\sum_{j=1}^{p}(y_j - t_j)^2, \qquad (2)$$

где j – номер нейрона выходного слоя; p – количество нейронов выходного слоя; y_j – значение выходного j-го нейрона; t_j – эталонное значение выходного j-го нейрона.

Для уменьшения ошибки изменяют весовые коэффициенты в направлении антиградиента E:

$$W_{ij}(h+1) = W_{ij}(h) - \Delta W_{ij}; \qquad (3)$$

$$\Delta W_{ij} = \alpha\frac{\partial E}{\partial W_{ij}}, \qquad (4)$$

где h – эпоха обучения; $W_{ij}(h)$ – весовой коэффициент от i-го нейрона к j-му нейрону; α – параметр скорости обучения; $\frac{\partial E}{\partial W_{ij}}$ – частная производная ошибки по весовому коэффициенту нейрона.

Частные производные ошибки по весовому коэффициенту от k-го нейрона скрытого слоя к j-му нейрону выходного вычислялись по формуле

$$\frac{\partial E}{\partial W_{kj}} = e_j f_j' y_k, \qquad (5)$$

где e_j – ошибка j-го нейрона выходного слоя

Частные производные ошибки по весовому коэффициенту от i-го нейрона входного слоя к k-му нейрону скрытого слоя

$$\frac{\partial E}{\partial W_{ik}} = e_k f_k' y_i, \qquad (6)$$

где e_k – ошибка k-го нейрона скрытого слоя; y_i – выходное значение i-го нейрона входного слоя.

По приведенным формулам (2)–(6) выполнялось изменение весов от слоя к слою до тех пор, пока суммарная квадратичная ошибка E не станет меньше или равна заданному значению. Количество примеров обучения составляет 3600. В процессе обучения, согласно (7), ошибка сети определялась как

$$e = \sum_{j=1}^{p} (y_j - t_j), \qquad (7)$$

где e – ошибка сети.

На рис. 5 показано, как изменялась ошибка сети в зависимости от эпох обучения.

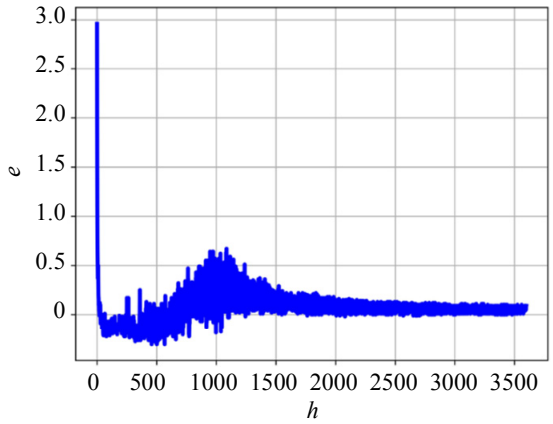

Рис. 5. Ошибка нейронной сети при обучении
Fig. 5. Error of the neural network during training

По завершении обучения нейронная сеть переводится в рабочее состояние.

Моделирование работы нейронной сети. Моделирование работы логической системы на основе нейронной сети выполнялось после обучения. Во время работы значения весовых коэффициентов сохранялись в памяти и использовались при необходимости выполнить преобразование для реализации изображенного на рис. 3 логического элемента.

При этом на вход сети подавался входной вектор X с периодом $T_p = 1$ изменения координат. На рис. 6 показаны эталонные координаты входного вектора X, подаваемого на вход нейронной сети, обозначенные как $x_i, i = 0...2$.

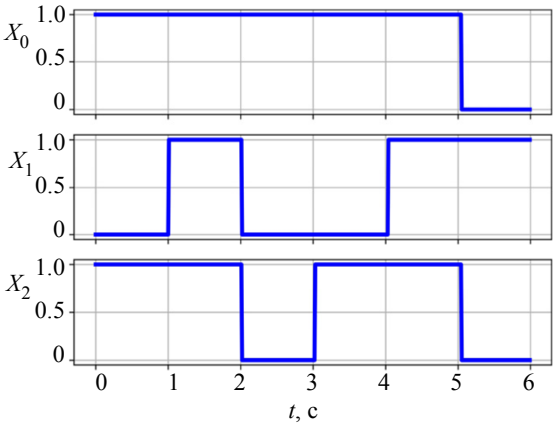

Рис. 6. Координаты входного вектора
Fig. 6. Input vector coordinates

На рис. 7 представлен результат работы сети с входным вектором X и координаты выходного вектора сети Y, обозначенные как y_i, $i = 0…7$.

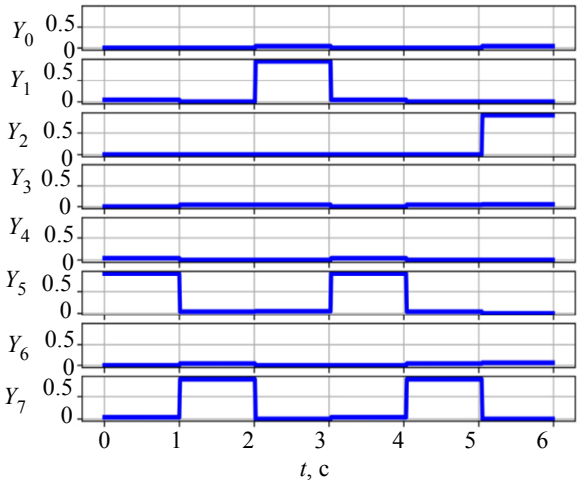

Рис. 7. Координаты выходного вектора сети
Fig. 7. Network output vector

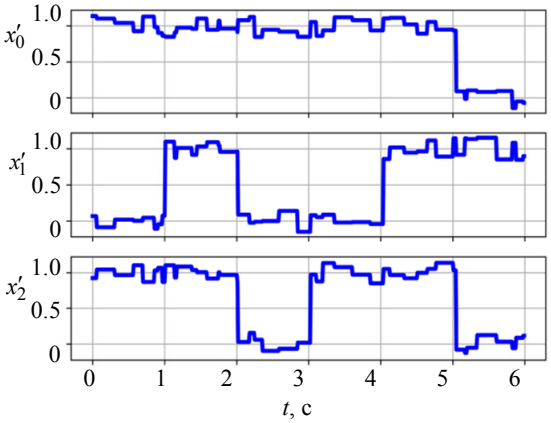

Рис. 8. Координаты входного вектора сети при влиянии помех
Fig. 8. Coordinates of the network input vector under the influence of noise

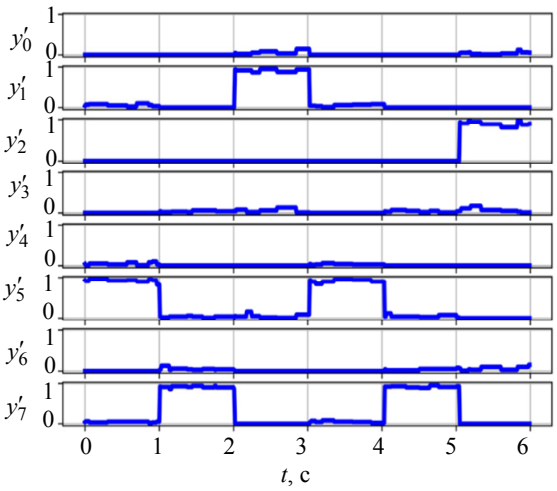

Рис. 9. Координаты выходного вектора сети при входном векторе с помехами
Fig. 9. Coordinates of the network output vector with an input vector with noise

На рис. 8 показаны координаты входного вектора X' с помехами, подаваемого на вход нейронной сети, обозначенные как $x'_i, i = 0...2$. Помехи во входном векторе X моделировались на основе пакета Random как случайные изменения по $x'_i = x_i \pm \text{random}(0...0.1)$ и по длительности $t_i = \text{random}(0...0.1)$ влияния.

На рис. 9 представлены выходные значения вектора сети Y, обозначенные как $y'_i, i = 0...7$, в результате работы с вектором X' на входе.

Представленные результаты моделирования определяют, что нейронная сеть реализует логические уравнения (1) с требуемой точностью. При наличии помех во входном векторе X выходной вектор Y лежит в требуемом интервале. Созданная программной реализацией нейронная сеть позволяет изменять архитектуру, параметры алгоритма обучения и с новыми обучающими данными применяться для решения других логических задач на стандартных промышленных контроллерах, которые используют язык программирования Python. Повышение производительности таких контроллеров задает перспективы применения техник искусственного интеллекта, и нейронных сетей в частности, в системах управления.

Список литературы

1. Титов В. С., Шевелев С. С. Арифметический вычислитель на элементах нейронной логики // Изв. вузов. Приборостроение. 2023. Т. 66, № 11. С. 950–959. doi: 10.17586/0021-3454-2023-66-11-950-959.

2. Лютикова Л. А. Разработка алгоритма на основе логических операций для обнаружения закономерностей в неполных данных // Изв. Кабардино-Балкарского науч. центра РАН. 2023. № 6 (116). С. 109–115. doi: 10.35330/1991-6639-2023-6-116-109-115.

3. Гуров С. И., Золотарёв Д. В., Самбурский А. И. Обучение с подкреплением в задаче синтеза мажоритарных схем // Современные информационные технологии и ИТ-образование. 2021. Т. 17, № 2. С. 295–307. doi: 10.25559/SITITO.17.202102.295-307.

4. Макмахан Б., Рао Д. Знакомство с PyTorch: глубокое обучение при обработке естественного языка. СПб.: Питер, 2020. 256 с.: ил.

5. Шакла Н. Машинное обучение и TensorFlow. СПб.: Питер, 2019. 336 с.: ил.

6. Уоссермен Ф. Нейрокомпъютерная техника: Теория и практика / пер. с англ. М.: Мир, 1992. 184 с.

7. Редько В. Г. Эволюция, нейронные сети, интеллект: Модели и концепции эволюционной кибернетики. М.: Ленанд, 2019. 220 с.

8. Горбань А. Н. Обучение нейронных сетей. М.: СП «ParaGraph», 1990. 160 с.

9. Галушкин А. И. Нейронные сети: основы теории. М.: РиС, 2015. 496 с.

10. Саммерфилд М. Программирование на Python 3: подробное руководство. М.: Символ-Плюс, 2011. 608 с.

11. Прохоренок Н. А., Дронов В. А. Python 3 и PyQt 5. Разработка приложений. СПб.: БХВ-Петербург, 2016. 832 с.: ил.

12. Васильев А. Н. Python на примерах. Практический курс по программированию. СПб.: Наука и Техника, 2016. 432 с.: ил.

Информация об авторе

Чернышев Денис Валентинович – кандидат технических наук, доцент, кафедра «Автоматика, телемеханика и связь», Дальневосточный государственный университет путей сообщения, ул. Серышева, 47, Хабаровск, 680021, Россия.
E-mail: fersy@mail.ru

References

1. Titov V. S., Shevelev S. S. Arifmeticheskij vychislitel' na jelementah nejronnoj logiki // Izv. vuzov. Priborostroenie. 2023. T. 66, № 11. S. 950–959. doi: 10.17586/0021-3454-2023-66-11-950-959. (In Russ.).

2. Ljutikova L. A. Razrabotka algoritma na osnove logicheskih operacij dlja obnaruzhenija zakonomernostej v nepolnyh dannyh // Izv. Kabardino-Balkarskogo nauch. centra RAN. 2023. № 6 (116). S. 109–115. doi: 10.35330/1991-6639-2023-6-116-109-115. (In Russ.).

3. Gurov S. I., Zolotarjov D. V., Samburskij A. I. Obuchenie s podkrepleniem v zadache sinteza mazhoritarnyh shem // Sovremennye informacionnye tehnologii i IT-obrazovanie. 2021. T. 17, № 2. S. 295–307. doi: 10.25559/SITITO.17.202102.295-307. (In Russ.).

4. Makmahan B., Rao D. Znakomstvo s PyTorch: glubokoe obuchenie pri obrabotke estestvennogo jazyka. SPb.: Piter, 2020. 256 s.: il. (In Russ.).

5. Shakla N. Mashinnoe obuchenie i TepsorFlow. SPb.: Piter, 2019. 336 s.: il. (In Russ.).

6. Uossermen F. Nejrokomp#juternaja tehnika: Teorija i praktika / per. s angl. M.: Mir, 1992. 184 c. (In Russ.).

7. Red'ko V. G. Jevoljucija, nejronnye seti, intellekt: Modeli i koncepcii jevoljucionnoj kibernetiki. M.: Lenand, 2019. 220 c. (In Russ.).

8. Gorban' A. N. Obuchenie nejronnyh setej. M.: SP «ParaGraph», 1990. 160 c. (In Russ.).

9. Galushkin A. I. Nejronnye seti: osnovy teorii. M.: RiS, 2015. 496 c. (In Russ.).

10. Sammerfild M. Programmirovanie na Python 3: podrobnoe rukovodstvo. M.: Simvol-Pljus, 2011. 608 c. (In Russ.).

11. Prohorenok N. A., Dronov V. A. Python 3 i PyQt 5. Razrabotka prilozhenij. SPb.: BHV-Peterburg, 2016. 832 s.: il. (In Russ.).

12. Vasil'ev A. N. Python na primerah. Praktiche-skij kurs po programmirovaniju. SPb.: Nauka i Tehnika, 2016. 432 s.: il. (In Russ.).

Information about the author

Denis V. Chernyshev – Cand. Sci. (Eng.), Associate Professor of the department of Automation, Telematics and Communication of Far Eastern State Transport University, Seryshev St., 47, Khabarovsk, 680021, Russia.
E-mail: fersy@mail.ru

Статья поступила в редакцию 12.03.2024; принята к публикации после рецензирования 09.06.2024; опубликована онлайн 30.09.2024.

Submitted 12.03.2024; accepted 09.06.2024; published online 30.09.2024.