ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ ҒЫЛЫМ ЖӘНЕ ЖОҒАРЫ БІЛІМ МИНИСТРЛІГІ
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН
MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE REPUBLIC OF KAZAKHSTAN

ХАЛЫҚАРАЛЫҚ УНИВЕРСИТЕТІ
INTERNATIONAL UNIVERSITY SINCE 2009
МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

2024 (19) 3

шілде - қыркүйек

ISSN 2708–2032 (print)
ISSN 2708–2040 (online)

Халықаралық ақпараттық және коммуникациялық технологиялар журналы
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Меншіктенуші: «Халықаралық ақпараттық технологиялар университеті» АҚ (Алматы қ.).
Қазақстан Республикасы Ақпарат және әлеуметтік даму министрлігінің Ақпарат комитетінде – **20.02.2020** жылы берілген.
№ KZ82VPY00020475 мерзімдік басылым тіркеуіне қойылу туралы куәлік.
Тақырыптық бағыты: ақпараттық технологиялар, әлеуметтік-экономикалық жүйелерді дамытудағы цифрлық технологиялар, ақпараттық қауіпсіздік және коммуникациялық технологияларға арналған.
Мерзімділігі: жылына 4 рет.
Тиражы: 100 дана
Редакцияның мекенжайы: 050040, Алматы қ-сы, Манас к-сі, 34/1, 709-кабинет, тел: +7 (727) 244-51-09.
E-mail: ijict@iitu.edu.kz Журнал сайты: https://journal.iitu.edu.kz

---

Международный журнал информационных и коммуникационных технологий
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)

Собственник: АО «Международный университет информационных технологий» (г. Алматы).

Свидетельство о постановке на учет периодического печатного издания в Министерство информации и общественного развития Республики Казахстан **№ KZ82VPY00020475,** выданное от **20.02.2020 г.**

Тематическая направленность: информационные технологии, информационная безопасность и коммуникационные технологии, цифровые технологии в развитии социо-экономических систем**.**

Периодичность: 4 раза в год.

Тираж: 100 экземпляров.

Адрес редакции: 050040 г. Алматы, ул. Манаса 34/1, каб. 709, тел: +7 (727) 244-51-09.

E-mail: ijict@iitu.edu.kz Сайт журнала: https://journal.iitu.edu.kz

«International Journal of Information and Communication Technologies»
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Owner: International Information Technology University JSC (Almaty).
The certificate of registration of a periodical printed publication in the Ministry of Information and Social Development of the Republic of Kazakhstan, Information Committee No. KZ82VPY00020475, issued on 20.02.2020.
Thematic focus: information technology, digital technologies in the development of socio-economic systems, information security and communication technologies
Periodicity: 4 times a year.
Circulation: 100 copies.
Editorial address: 050040. Manas st. 34/1, Almaty. +7 (727) 244-51-09. E-mail: ijict@iitu.edu.kz
Journal website: https://journal.iitu.edu.kz

МАЗМҰНЫ

АҚПАРАТТЫҚ ТЕХНОЛОГИЯЛАР

АҚПАРАТТЫҚ ҚАУІПСІЗДІК ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАРҒА АРНАЛҒАН

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

INFORMATION TECHNOLOGY

INFORMATION SECURITY AND COMMUNICATION TECHNOLOGIES

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Vol. 5. Is. 3. Number 19 (2024). Pp. 32–48
Journal homepage: https://journal.iitu.edu.kz
https://doi.org/10.54309/IJICT.2024.19.3.003
УДК: 004.85, 004.

MATHEMATICAL APPROACH OF THE BACKPROPAGATION METHOD FOR CONSTRUCTING ARTIFICIAL NEURAL NETWORKS

A.B. Yemberdiyeva[1]*, I.C. Young[2], S.Ye. Mamanova[1], S.B. Mukhanov[1]
[1]International Information Technology University, Almaty, Kazakhstan;
[2]Gachon University Seoul, Republic of Korea.
E-mail: s.mukhanov@edu.iitu.kz

Aknur Yemberdiyeva — Master in CSSE, lecturer, Computer Engineering, International Information Technology University
E-mail:a.yemberdiyeva@iitu.edu.kz;
Young I. Cho — PhD Computer Science, Professor, Gachon University, Seoul, the Republic of Korea
E-mail: yicho@gachon.ac.kr
Symbat Mamanova — Doctoral (PhD) student Computer Engineering, senior-lecturer, International Information Technology University
E-mail: s.mukhanov@edu.iitu.kz,
Samat B. Mukhanov — PhD CSSE, assistant-professor, Computer Engineering, International Infor-mation Technology University
E-mail: s.mamanova@iitu.edu.kz

Abstract. Backpropagation isthe core part of a neural network. This method is used to efficiently train a network using a chain rule that allows differentiation of complex functions. In other words, after each pass through the network, the backpropagation method performs a backward pass to adjust the model parameters, such as weights and biases. This article highlights the importance of using the backpropagation method from the point of view of mathematical formulas for neural networks. The importance of using the backpropaga-tion learning algorithm to calculate the gradient (gradient descent) and the need to use the activation function to minimize the loss function is mathematically described and calculated by formulas, and also proven by calculating the matrix products of vectors for each layer of parameters - weights and biases and applying complex differential equations.

Keywords: backpropagation method; loss function; ANN (artificial neural network); gradient descent, activation function; weights; biases; parameters

For citation: *A.B. Yemberdiyeva, I.C. Young, S.Ye. Mamanova, S.B. Mukhanov. MATHEMATICAL APPROACH OF THE BACKPROPAGATION METHOD FOR CONSTRUCTING ARTIFICIAL NEURAL NETWORKS//INTERNATIONAL JOUR-NAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES. 2024. Vol. 5. No. 19. Pp. 32–48 (In Eng.). https://doi.org/10.54309/IJICT.2024.19.3.003.*

ЖАСАНДЫ НЕЙРОНДЫҚ ЖЕЛІЛЕРДІ ҚҰРУ ҮШІН КЕРІ ТАРАЛУ ӘДІСІНІҢ МАТЕМАТИКАЛЫҚ ТӘСІЛІ

А.Б. Ембердіева[1]*, I.C. Young[2], С.Е. Маманова[1], С.Б. Муханов[1]
[1]International Information Technology University, Almaty, Kazakhstan;
[2]Gachon University Seoul, Republic of Korea.
E-mail: s.mukhanov@edu.iitu.kz

Ақнұр Ембердиева — Магистр, Халықаралық ақпараттық технологиялар университетінің ком-пьютерлік инженерия кафедрасының оқытушысы
E-mail: a.yemberdiyeva@iitu.edu.kz;
I.C. Young — «Информатика» Ғылымдарының Кандидаты, Профессор, Гачон Университеті, Сеул, Корея Республикасы
E-mail: yicho@gachon.ac.kr;
Сымбат Маманова — «Есептеу техникасы» мамандығы бойынша докторант (PhD), халықара-лық Ақпараттық Технологиялар Университетінің аға оқытушысы
E-mail: s.mukhanov@edu.iitu.kz;
Самат Б. Мұханов — ОӘК PhD Докторы, халықаралық ақпараттық технологиялар универси-тетінің есептеу техникасы кафедрасының доценті
E-mail: s.mamanova@iitu.edu.kz

Аннотация. Кері таралу нейрондық желінің негізгі бөлігі болуы мүмкін. Бұл әдіс күрделі мүмкіндіктерді ажырата алатын тізбекті ережені пайдаланып желіні тиімді оқыту үшін қолданылады. Басқаша айтқанда, желі арқылы әрбір өтуден кейін кері таралу әдісі салмақтар мен ауытқулар сияқты модель параметрлерін реттеу үшін кері өтуді орындайды. Бұл мақала нейрондық желілер үшін математикалық формулалар тұрғысынан кері таралу әдісін қолданудың маңыздылығын көрсетеді. Математикалық сипатталған және формулалармен есептелген, сонымен қатар параметрлердің әрбір қабаты үшін векторлардың матрицалық туындыларын есептеу арқылы дәлелденген — салмақтар мен қиғаштықтар және күрделі дифференциалданған теңдеулерді қолдану) градиентті (градиенттің төмендеуі) және есептеу үшін кері таралуды оқыту алгоритмін пайдаланудың маңыздылығы. функцияның жоғалуын азайту үшін белсендіру функциясын пайдалану қажет.

Түйін сөздер: кері таралу әдісі; жоғалту функциясы; ANN (Жасанды нейрондық желі); градиенттің түсуі, белсендіру функциясы; салмақ; ығысулар; параметрлері

Дәйексөз үшін: *А.Б. Ембердіева, I.C. Young, С.Е. Маманова, С.Б. Муханов. ЖАСАНДЫ НЕЙРОНДЫҚ ЖЕЛІЛЕРДІ ҚҰРУ ҮШІН КЕРІ ТАРАЛУ ӘДІСІНІҢ МАТЕМАТИКАЛЫҚ ТӘСІЛІ//ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ. 2024. Т. 5. No. 19. 32–48 бет. (ағылшын тілінде). https://doi.org/10.54309/IJICT.2024.19.3.003.*

МАТЕМАТИЧЕСКИЙ ПОДХОД МЕТОДА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ДЛЯ ПОСТРОЕНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

А.Б. Ембердиева[1]*, I.C. Young[2], С.Е. Маманова[1], С.Б. Муханов[1]
[1]International Information Technology University, Almaty, Kazakhstan;
[2]Gachon University Seoul, Korea.
E-mail: s.mukhanov@edu.iitu.kz

Акнур Ембердиева — магистр CASE, преподаватель кафедры компьютерной инженерии Международного университета информационных технологий
E-mail: a.yemberdiyeva@iitu.edu. kz;
I.C. Young — доктор философии по специальности «Компьютерные науки», профессор Университета Гачон, Сеул, Республика Корея
E-mail: yicho@gachon.ac.kr
Сымбат Маманова — докторант по специальности «Компьютерная инженерия», старший преподаватель Международного университета информационных технологий
E-mail: s.mukhanov@edu.iitu.kz;
Самат Б. Муханов — доктор PhD, ассистент-профессор кафедры компьютерной инженерии Международного университета информационных технологий
E-mail: s.mamanova@iitu.edu.kz

Аннотация. Метод обратного распространения ошибки, вероятно, является основной частью нейронной сети. Этот метод применяется для эффективного обучения сети, используя цепное правило, которое позволяет дифференцировать сложные функции. Другими словами, после каждого прохода через сеть метод обратного распространения выполняет обратный проход, чтобы скорректировать параметры модели, такие как веса и смещения. В данной статье освящается важность применения метода обратного распространения ошибки с точки зрения математических формул для нейронных сетей. Математически описана и доказана расчетами матричных произведений векторов для каждого слоя параметров важность применения алгоритма обучения метода обратного распространения ошибок для вычисления градиента (gradient descent) и необходимость применения функции активаций для минимизации функции потерь.

Ключевые слова: метод обратного распространения; loss function; ANN (Artificial neural network); градиентный спуск, функция активации; веса; смещения; параметры

Для цитирования: *А.Б. Ембердиева, И.Чо. Янг, С.Е. Маманова, С.Б. Муханов. МАТЕМАТИЧЕСКИЙ ПОДХОД МЕТОДА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ДЛЯ ПОСТРОЕНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ//МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ. 2024. Т. 5. No. 19. Стр. 32–48. (На анг.). https://doi.org/10.54309/IJICT.2024.19.3.003.*

Introduction

Backpropagation is one of the well-known methods used for deep learning of feedforward neural networks, also called multilayer perceptrons. This method is related to supervised learning, which requires setting target values in training examples. In this article, we will consider what backpropagation is, how it is implemented, and its pros and cons (Mukhanov et al., 2020: 31–37).

Modern feedforward neural networks are used to solve many complex problems. In training such networks using the backpropagation method, two types of passes are used: forward and backward. During the forward pass, the input vector is fed to the input layer of the network, after which the signals are propagated through the layers of the network, forming a set of output signals, which are the network's response to a given input image (Mukhanov et al., 2023: 16–27).

At this stage, all synaptic weights are fixed. The backward pass involves adjusting the synaptic weights according to error correction rules: the difference between the actual and desired outputs is calculated and an error signal is formed. This signal is then propagated back through the network, in the direction opposite to the direction of the synaptic connections. That is why this method is called the backpropagation algorithm. The weights are adjusted so that the network output signals are as close as possible to the desired values (Mukhanov et al., 2023: 15–27).

Problem, relevance

There are several issues and relevance aspects to consider with backpropagation:

Issues of backpropagation:

Vanishing gradient problem:

In deep neural networks, gradients can decrease exponentially during backpropagation, especially in layers closer to the input. This makes it difficult to train these layers and can lead to insufficient weight updates, which slows down or stops training (Kenshimov et al., 2021: 44–54).

Exploding gradient problem:

In some cases, gradients can increase too quickly, which can lead to instability in training and large fluctuations in weight values.

Dependence on hyperparameter selection:

The effectiveness of the method depends on the correct choice of hyperparameters, such as learning rate, regularization, and initialization of weights. Incorrectly setting these parameters can significantly worsen training results.

Time and computational resources:

Training deep neural networks using backpropagation is computationally expensive and time-consuming, especially when working with large amounts of data and complex models.

Local Minima:

The loss function may have many local minima, and backpropagation may get stuck in these local minima instead of finding the global minimum, resulting in suboptimal solutions (Uskenbayeva et al., 2020: 1–6; Bazarevsky et al., 2019; Vidyanova, 2022).

Relevance of Backpropagation:

A Foundational Method for Deep Learning:

Despite its limitations, backpropagation remains a fundamental and widely used method for training deep neural networks. Its efficiency and ease of implementation have made it a standard in the field of machine learning.

Support for Modern Architectures:

Backpropagation is the basis for many modern deep learning architectures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers.

Development of new methods and improvements:

The problems associated with backpropagation have inspired the development of new methods and improvements, such as optimizers (e.g., Adam, RMSprop), improved weight initialization methods, and architectural innovations such as layer-wise normalization and dropout (Wang et al., 2020; Lee et al., 2020).

Importance to Practical Applications:

Backpropagation remains relevant due to its importance in practical applications of deep learning, including image processing, speech recognition, and natural language recognition, making it a key tool in modern AI systems (Bilgin et al., 2019; Kudubaeva et al., 2016; Liukai et al., 2022: 103364).

Thus, backpropagation plays a central role in training neural networks despite its existing problems and remains relevant and in demand in modern research and applications (Yuanguo et al., 2023: 103688; Baiju et al., 2023: 119042).

Materials and methods

A key stage in training neural networks involves using backpropagation algorithm, which correct errors through a process known as backpropagation. This technique applies gradient descent in multilayer feedforward networks. Its central idea is to efficiently compute the partial derivatives of the network function F (w, x) with respect to each element of the weight vector W, based on a given input vector X. The algorithm's goal is to determine the error gradient for all parameters in the model (Guoxiang et al., 2023: 118912; Laura-Bianca et al., 2023: 84–90; Yeo et al., 2013).

We will focus on standard fully connected networks for the classification task. However, many of these principles are also relevant to other types of neural networks and to any differentiable computational graphs in general. When computing within a single fully connected layer, let us first consider working with row-oriented vectors instead of column-oriented ones:

$$x = [x_1 \quad x_2], \tag{1}$$

$$h = [h_1 \quad h_2 \quad h_3], \tag{2}$$

$$b = [b_1 \quad b_2 \quad b_3], \tag{3}$$

Thus, the output vector h is calculated using the nonlinear activation function:

$$h = F(xW + b), \tag{4}$$

To calculate, for example, first component (element) of the vector h_1, you need to perform the following steps:

x_1 and x_2 for w_{11} and w_{21} from the matrix:

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}, \tag{5}$$

We get

$$h_1 = F(x_1 w_{11} + x_2 w_{21} + b_1), \tag{6}$$

In the same way, h_2 and h_3 are calculated.

Let us split the calculations of one layer into two stages: linear and nonlinear. Let us assume that after the linear part we get a vector t. Then the nonlinear function applied to each element transforms it into the final vector h then, $t = xW + b$, $h = F(t)$. Let us break this down into components for one element:

$$t_1 = x_1 w_{11} + x_2 w_{21} + b_1, \quad (7)$$

$$h_1 = F(t_1), \quad (8)$$

In our case, it is important to realize that fully connected layers of a neural network are just special cases of computational graphs. Let us look at an example of a graph visualization for a fully connected layer in question. This will help us later understand how we navigate the graph during backpropagation.

The computation graph for a fully connected layer look like this. Nodes with data $x, t \text{ and } h$, and to compute node t we need nodes with parameters $w \text{ and } b$.

To optimize the parameters of a neural network using an optimization algorithm, we need the error gradient vector for all trainable parameters of our model:

$$\frac{\partial E}{\partial \Omega} = \left\{\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_2}, \dots\right\}, \quad (9)$$

The number of elements in the gradient corresponds to the number of trainable parameters, and for convenience we can divide the gradient into groups corresponding to different unions of parameters. For example, if we have a weight matrix W, we can consider it as a separate object with its own set of parameters, and therefore it is necessary to have a part of the gradient of the same size for it. Let us denote it as $\frac{\partial E}{\partial W}$ that is, as the partial derivative of the error E with respect to the matrix W for our current layer. Similarly, for a vector b, the corresponding part of the gradient is $\frac{\partial E}{\partial b}$. For each layer and for each object, we train them with parameters: for b_1 it will be $\frac{\partial E}{\partial b_1}$, for W_2 it will be $\frac{\partial E}{\partial W_2}$, for b_2 it will be $\frac{\partial E}{\partial b_2}$. Thus, it is a scalar, a vector, a matrix, or a tensor.

It is important to realize that in most cases the learning algorithm, such as Stochastic Gradient Descent, and the gradient computation algorithm (BACKPROP) can be completely independent. The learning algorithm wants to get the gradient, and it does not care how it was computed. In the process of computing the gradient, we do not care how it will be used during training. So, Let us focus on computing the gradient. Suppose that this layer is part of a certain model that we are training, and we need to find $\frac{\partial E}{\partial W}$ and $\frac{\partial E}{\partial b}$. For this layer, we use the Chain Rule from calculus and work backwards. Let us assume that we are already given $\frac{\partial E}{\partial h}$

in numerical form. That is, given the input to the network with given weights, we got $\frac{\partial E}{\partial h}$. Its dimension is the same as the vector h, that is, it is a row vector of three elements:

$$\left[\frac{\partial E}{\partial h_1} \quad \frac{\partial E}{\partial h_2} \quad \frac{\partial E}{\partial h_3}\right], \tag{10}$$

Then we can calculate $\frac{\partial E}{\partial t}$. This will also be a vector of three elements, like the vector t. Now, understanding $\frac{\partial E}{\partial t}$, we can calculate $\frac{\partial E}{\partial w}$ and $\frac{\partial E}{\partial b}$, the quantities we need. These will have the same dimensions as the original matrices and vectors. $\frac{\partial E}{\partial w}$ – will be 2x3 matrix, $\frac{\partial E}{\partial b}$ –will be a vector of three elements. Additionally, we can also compute the gradient with respect to the input, $\frac{\partial E}{\partial x}$.

This is necessary to propagate the gradient back to the previous layer and preform similar computations for its parameters. Here, vector x is the input to our layer, and h is the output from the previous layer. In this example, $\frac{\partial E}{\partial x}$ consists of two-elements vectors.

Now, we need to compute and output these gradients in sequence.

Let us start with $\frac{\partial E}{\partial t}$, given that $\frac{\partial E}{\partial h}$ is available. The most accurate approach is to track the individual components of the gradient vector. For instance, consider $\frac{\partial E}{\partial t_1}$, which represents the partial derivative of the error function E with respect to t_1. We will utilize the fact that the error function E depends on h_1, and h_1 depends on t_1. Therefore, applying the chain rule:

$$\frac{\partial E}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot \frac{\partial h_1}{\partial t_1}, \tag{11}$$

It all comes down to numerical calculations. It would be beneficial to fully write out the derivative using the rule for differentiating a complex function with multiple variables, considering the intermediate variables that link t_1 with the error E. However, from the diagram, it is evident that this connection is only through h_1, while h_2 and h_3 are independent of t_1.Therefore, no additional explanation is needed. Now, Let us examine the results we have obtained:

$$\frac{\partial E}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot \frac{\partial h_1}{\partial t_1} = \frac{\partial E}{\partial h_1}, \tag{12}$$

What does $\frac{\partial h_1}{\partial t_1}$. As shown, $\frac{\partial h_1}{\partial t_1}$ is connected through the scalar function $h_1 = F(t_1)$. This means that $\frac{\partial h_1}{\partial t_1}$ is simply the derivative of the function F evaluated at t_1:

$$\frac{\partial E}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot \frac{\partial h_1}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot F'(t_1), \qquad (13)$$

subsequently, in a similar manner

$$\frac{\partial E}{\partial t_2} = \frac{\partial E}{\partial h_2} \cdot F'(t_2) \text{ и } \frac{\partial E}{\partial t_3} = \frac{\partial E}{\partial h_3} \cdot F'(t_3), \qquad (14)$$

This is sufficient for calculating the vector $\frac{\partial E}{\partial t}$. However, this expression can be represented in a more compact form. In this case, we perform element-wise multiplication of the two vectors.

$$\frac{\partial E}{\partial h} = \left[\frac{\partial E}{\partial h_1} \quad \frac{\partial E}{\partial h_2} \quad \frac{\partial E}{\partial h_3}\right], \qquad (15)$$

on a vector consisting of derivatives of the function at different points:

$$F'(t_1) = [F(t_1)' \quad F(t_2) \quad 'F(t_3)')], \qquad (16)$$

This is equivalent to applying the function $F'(t)$ elementwise to the vector t. The resulting final vector then has the following gorm:

$$\frac{\partial E}{\partial t} = \frac{\partial E}{\partial h} \odot F'(t) \qquad (17)$$

This vector also has three elements. Such an element-wise multiplication is often referred to as the Hadamard product. Thus, we have derived an expression for computing the vector $\frac{\partial E}{\partial t}$.

Next, Let us examine the gradient with respect to the matrix W.Similarly, we track the individual elements of the matrix and apply the chain rule of differentiation. We start with $\frac{\partial E}{\partial w_{11}}$. The weight w_{11} connects the input x_1 to the output t_1. Therefore, it does not contribute to t_2 or t_3. Consequently:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial w_{11}}, \qquad (18)$$

What is $\frac{\partial t_1}{\partial w_{11}}$. This represents the differential of t_1:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial w_{11}} = \frac{\partial E}{\partial t_1} \cdot x_1, \qquad (19)$$

Let us consider the derivative for another matrix element. We will start by changing the first index, which means we will move downwards:

$$\frac{\partial E}{\partial w_{21}} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial w_{21}} = \frac{\partial E}{\partial t_1} \cdot x_2, \qquad (20)$$

If we move along the columns and increase the second index, this weight now connects the input to the output t_2. Therefore, this derivative will be equal to:

$$\frac{\partial E}{\partial w_{12}} = \frac{\partial E}{\partial t_2} \cdot \frac{\partial t_2}{\partial w_{12}} = \frac{\partial E}{\partial t_2} \cdot x_1, \qquad (21)$$

Similarly, we can apply the same approach to all other elements of the gradient $\frac{\partial E}{\partial w}$. It is important to note that there is a correlation between the weight index w and the indices x and t. The first index corresponds to the x index, while the second index corresponds to the t index. This process closely resembles matrix multiplication. In fact, we can compactly represent the final matrix $\frac{\partial E}{\partial w}$ using matrix multiplication. Let us consider:

$$x^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad (22)$$

$$\frac{\partial E}{\partial t} = \begin{bmatrix} \frac{\partial E}{\partial t_1} & \frac{\partial E}{\partial t_2} & \frac{\partial E}{\partial t_3} \end{bmatrix}, \qquad (23)$$

then

$$\frac{\partial E}{\partial W} = x^T \cdot \frac{\partial E}{\partial t}, \qquad (24)$$

Now that we have the gradients with respect to the weight matrix, Let us look at the vector $\frac{\partial E}{\partial b}$. The contribution to the first element comes solely from the component in t_1:

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial b_1} = \frac{\partial E}{\partial t_1} \cdot 1 \quad (25)$$

Similarly for the other two elements:

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial t_2} \text{ и } \frac{\partial E}{\partial b_3} = \frac{\partial E}{\partial t_3}, \quad (26)$$

then

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial t} \quad (27)$$

Both vectors consist of three elements.

Now, the most crucial step: computing the gradient with respect to the input. This process will be more complex. We need to determine $\frac{\partial E}{\partial x}$, a two-element vector, given $\frac{\partial E}{\partial t}$. We start with $\frac{\partial E}{\partial x_1}$, as x_1 contributes to all elements of the vector t.

Next, we must apply the chain rule for differentiating a function of several variables. This will result in a sum over intermediate variables.

$$\frac{\partial E}{\partial x_1} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial x_1} + \frac{\partial E}{\partial t_2} \cdot \frac{\partial t_2}{\partial x_1} + \frac{\partial E}{\partial t_3} \cdot \frac{\partial t_3}{\partial x_1}, \quad (28)$$

Note that a sum over all elements has appeared, which includes the contribution from $\frac{\partial E}{\partial x_1}$. By using the same relationships, we can obtain the corresponding weights W:

$$\frac{\partial t_1}{\partial x_1} = w_{11}, \frac{\partial t_2}{\partial x_1} = w_{12}, \frac{\partial t_3}{\partial x_1} = w_{13}, \quad (29)$$

These weights are precisely those that connect x_1 to t_1, t_2 and t_3:

$$\frac{\partial E}{\partial x_1} = \frac{\partial E}{\partial t_1} \cdot w_{11} + \frac{\partial E}{\partial t_2} \cdot w_{12} + \frac{\partial E}{\partial t_3} \cdot w_{13}, \quad (30)$$

Similarly, for x_2 there will already be other weights w:

$$\frac{\partial E}{\partial x_2} = \frac{\partial E}{\partial t_1} \cdot w_{21} + \frac{\partial E}{\partial t_2} \cdot w_{22} + \frac{\partial E}{\partial t_3} \cdot w_{23}, \quad (31)$$

Let us also note the hidden matrix multiplication here and express everything in a compact form:

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial t} \cdot w^T . \qquad (32)$$

Now we understand how to find gradients for the parameters of a fully connected layer and, furthermore, how to propagate the gradients to the previous layer to perform similar computations for its parameters, etc. To do this, we calculated $\frac{\partial E}{\partial x}$. However, it is not clear what assumptions we have made about knowing $\frac{\partial E}{\partial h}$. If we have $\frac{\partial E}{\partial h}$ for the final layer, we can sequentially compute all gradients for the preceding layers. So, how do we obtain $\frac{\partial E}{\partial h}$, from which we need to start? Since this is the last layer, its output is related to the final error E. We need to compute the corresponding derivative. In the final layer, we do not use an activation function, so our task reduces to finding $\frac{\partial E}{\partial t}$, and we will proceed from there. To do this, we need to understand how the final t is related to the error E. After performing certain operations, we should obtain a one-dimensional array(scalar) E, the error, which always represents a single number, regardless of other factors. To start, Let us compute the final predictions of our model:

$$Z = \mathrm{Softmax}(t) = S(t) = \left\{\frac{e^{t_i}}{\sum_j e^{t_i}}\right\}, \qquad (33)$$

We applied the exponential function to each element of the vector, mapping them monotonically to the range from zero to positive infinity. We then divided by the sum to ensure that the final probabilities sum to one. Now that we have the probabilities provided by the neural network as its output, we can calculate the prediction error. For this, we also need the known correct answer, which we will denote as y. Recall that y is a vector of zeros with one in the position corresponding to the index of the correct class (in this case, 0,1 or 2). This represents the true distribution we aim to match with the given neural network input. The error can then be calculated as follows:

$$E = \mathrm{CrossEntropy}(z,y) = -\sum_i y_i \ln z_i , \qquad (34)$$

y – is the correct answer, z – is the output from Softmax.

Thus, we have a combination of $\mathrm{Softmax}$ and $\mathrm{CrossEntropy}$ in this case we can substitude one into the other, simplify, and the differentiation process will be simplified:

$$E = \mathrm{CrossEntrophy}(S(t), y) = -\sum_i y_i \ln \frac{e^{t_i}}{\sum_j e^{t_j}} = -\sum_i y_i\left(t_i - \ln \sum_j e^{t_j}\right) = -\sum_i y_i t_i + \sum_i y_i \ln \sum_j e^{t_j} = -\sum_i y_i t_i + \ln \sum_j e^{t_j}, \quad (35)$$

We derived a straightforward relationship between E and t:

$$E = \mathrm{CrossEntropy}(S(t), y) = -\sum_i y_i t_i + \ln \sum_j e^{t_j}, \quad (36)$$

We can calculate the required vector $\frac{\partial E}{\partial t}$. Let us express one of its elements:

$$\frac{\partial E}{\partial t_k} = -y_k + \frac{1}{\sum_j e^{t_k}} \cdot e^{t_k}, \quad (37)$$

If we examine closely, this corresponds to one of the Softmax elements:

$$\frac{\partial E}{\partial t_k} = -y_k + \frac{1}{\sum_j e^{t_k}} \cdot e^{t_k} = S(t)_k - y_k, \quad (38)$$

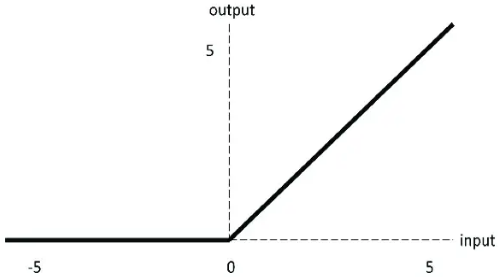
Now Let us express this for the entire vector:

$$\frac{\partial E}{\partial t} = S(t) - y = Z - y, \quad (39)$$

Activation Function: We will use one of the simplest and most popular functions, ReLU:

$$F(t) = \mathrm{ReLU}(t) = \max(o, t), \quad (40)$$

$$F'(t) = \begin{cases} 1, t \geq 0 \\ 0, t < 0 \end{cases}, \quad (41)$$



Picture 1 – Activation Function ReLU

Now that we have all the necessary information, we can put it together. Consider a neural network with two fully connected layers. It there are more layers, the process would be similar. Let us create a computational graph for this neural network- a feedforward graph. The input to the network is x, which is transformed into t_1. For this transformation, we need a matrix W_1 and a vector b_1:

$$t_1 = xW_1 + b_1, \tag{42}$$

In this context, the index refers to the layer number rather then element number as before. t_1 is transformed into h_1, which is the output of the first layer. This output is then fed into the second layer and transformed into t_2 using the matrix W_2 and the bias vector b_2:

$$h_1 = F(t_1), \tag{43}$$

$$t_2 = h_1 W_2 + b_2 \tag{44}$$

In the final layer, no activation function is applied. Instead, we directly obtain the probabilities Z using the $\mathrm{Softmax}$ function. We then compute the error using $\mathrm{CrossEntropy}$:

$$Z = S(t_2), \tag{45}$$

$$E = CE(z,y). \tag{46}$$

After computing the error, we proceed by calculating gradients in the reverse direction along the computational graph, which is the essence of the backpropagation algorithm.

We know how to determine $\frac{\partial E}{\partial t_2}$, t_2 – is the final vector t in the last layer. Given $\frac{\partial E}{\partial w_2}$, we can compute $\frac{\partial E}{\partial b_2}$, the two parameters of the second layer. We can also calculate $\frac{\partial E}{\partial h_1}$, the gradients at the output of the first layer, which allows us to find $\frac{\partial E}{\partial t_1}$. From we can then determine $\frac{\partial E}{\partial w_1}$ and $\frac{\partial E}{\partial b_1}$.

There is no need to compute $\frac{\partial E}{\partial x}$ since it is not propagated further. Now, Let us summarize everything we have derived so far:

$$\frac{\partial E}{\partial w_2} = S(t_2) - y = Z - y \tag{47}$$

$$\frac{\partial E}{\partial w_2} = h_1^T \cdot \frac{\partial E}{\partial t_2} \quad (48)$$

The next step is to calculate the derivative of the vector for the second layer:

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial t_2} \quad (49)$$

Next, we calculate the derivative by multiplying by the weights for the first layer:

$$\frac{\partial E}{\partial h_1} = \frac{\partial E}{\partial t_2} \cdot W_2^T \quad (50)$$

Next, we calculate the derivative with the Hadamard product for the derivative of the vector function t of the first layer:

$$\frac{\partial E}{\partial h_1} = \frac{\partial E}{\partial h_1} \odot F'(t_1) \quad (51)$$

Then, we calculate x transposed by the derivative for the second layer:

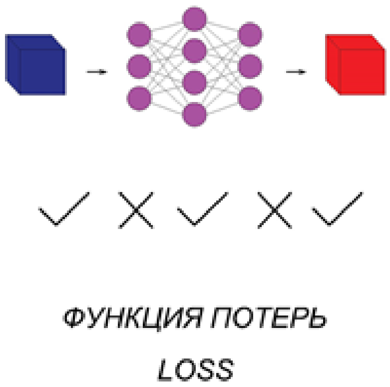
$$\frac{\partial E}{\partial w_2} = x^T \cdot \frac{\partial E}{\partial t_2} \quad (52)$$

Finally, we complete the last step with derivatives for the second layer:

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial t_1} \quad (53)$$

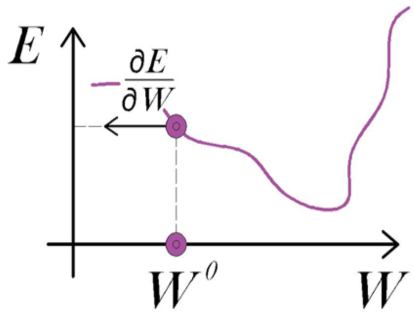
We have performed the calculations to determine the gradients using partial differential equations and the chain rule for complex functions, which involve computing gradients for matrices and vectors. This intricate and extensive process is part of the backpropagation algorithm. The upcoming chapters will cover the technical implementation of this algorithm in Python using machine learning libraries, including for convolutional and recurrent neural networks.

Next, we will visualize the results through graphs, starting with a depiction of the loss function (Picture 2).



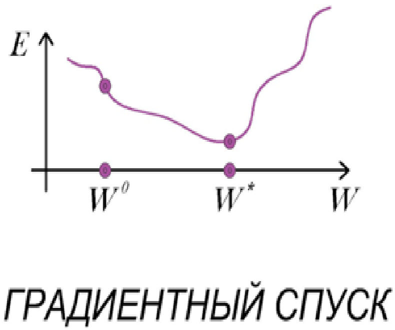
Picture 2 – Loss function in the structure of neural networks

The following graph shows the error function being found:



Picture 3 – Error function in gradient

E – is the error function (cross-entropy), w – weight.
Then we calculate the gradients descent using the following 3.61.



Picture 4 – Gradient descent

$$W^{t+1} = W^{t} - \alpha * \frac{\partial E}{\partial w} \cdot (W^{t}) \quad , \tag{54}$$

Where W – weights, α – is the learning rate, $\frac{\partial E}{\partial w}$ – is the gradient of the error, t – is a vector.

Below are records of the mathematical calculations, equation formulas, and concepts studied from textbooks on linear algebra, analytical geometry, mathematical analysis, statistics, mathematical logic, and algorithm theory, all of which are applied in neural networks.

Picture 5 – Calculations of complex architecture in layers of neural networks

In conclusion, all the mathematical calculations performed for each layer of neural networks have been visualized. The theoretical and analytical studies conducted will be further tested through experimental validation. Additionally, software for implementing recognition tasks will be developed.

Conclusion

The advantages of the method include its ease of implementation and resistance to outliers and anomalies in the data. However, there are also disadvantages:

- long training time;
- the possibility of "network paralysis", when at large values the activation function falls into the sigmoid saturation region, and its derivative tends to zero, which slows down the weight update and slows down the learning process;
- a tendency to get stuck in local minima of the error function.

The introduction of this algorithm was an important step in the development of neural networks, as it is an effective method for training multilayer perceptrons from the point of view of computational processes. However, it would be a mistake to think that the algorithm offers an ideal solution to all possible problems.

REFERENCES

Anna Vidyanova (2022). "In the USA, they are interested in the development of Kazakhs for the deaf". — Capital. — 2022. https://kapital.kz/business/105455/v-ssha-zainteresovalis-razrabotkoykazakhstantsev-dlya-glukhikh.html.

Bazarevsky V., Fan Zh. (2019). On-device, real-time hand tracking with mediapipe. Google AI Blog. — Available at: https:// ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html.

Baiju Yan, Peng Wang, Lidong Du, Xianxiang Chen, Zhen Fang, YirongWu (2023). "mmGesture: Semi-supervised gesture recognition system using mmWave radar". — 2023. — Vol. 213. — P. B. — P. 119042.

Bilgin M. & Mutludogan K. (2019). American Sign Language character recognition with capsule networks. Proceedings of the 3 rd International Symposium on Multidisciplinary Studies and Innovative Technologies. — Ankara, Turkey. https://doi.org/10.1109/ismsit.2019.8932829.

Guoxiang Tong, Yueyang Li, Haoyu Zhang, Naixue Xiong (2023). A Fine-grained Channel State Information-based Deep Learning System for Dynamic Gesture Recognition // Information Sciences. — 2023. — Vol. 636. — P. 118912.

Kenshimov C., Mukhanov S., Merembayev T., Yedilkhan D. (2021). A Comparison of Convolutional Neural Networks for Kazakh Sign Language Recognition Eastern-European // Journal of Enterprise Technologies. — 2021. — Vol. 5. — № 2. —113. — Pp. 44–54.

Kudubaeva S.A., Ryumin D.A. and Kalzhanov M.U. (2016). Support vector machine for sign speech recognition using the KINECT sensor. —Volume 91. — No. 3. (2016): Bulletin of KazNU. Series "Mathematics, mechanics, computer science". https://bm.kaznu.kz/index.php/kaznu/article/view/541.

Lee A.R., Cho Y., Jin S. & Kim N. (2020). Enhancement of surgical hand gesture recognition using a capsule network for a contactless interface in the operating room. Computer methods and programs in biomedicine. — 190. — 105385. https://doi.org/10.1016/j.cmpb.2020.105385.

Liukai Xu, Keqin Zhang, Genke Yang, Jian Chu (2022). Gesture recognition using dual-stream CNN based on fusion of sEMG energy kernel phase portrait and IMU amplitude image // Biomedical Signal Processing and Control. — 2022. — Vol. 73. — P. 103364.

Laura-Bianca Bilius, Ştefan-Gheorghe Pentiuc, Radu-Daniel Vatavu (2023). TIGER: A Tucker-based instrument for gesture recognition with inertial sensors // Pattern Recognition Letters. — 2023. — Vol. 165. — Pp. 84–90.

Mukhanov S.B., Uskenbayeva R.K. (2020). Pattern Recognition with Using Effective Algorithms and Methods of Computer Vision Library // Advances in Intelligent Systems and Computing. — 2020. — №1. — Pp. 31–37.

Mukhanov Samat, Uskenbayeva Raissa, Im Cho Young, Dauren Kabyl, Les Nurzhan, Amangeldi Maqsat (2023). Gesture Recognition of Machine Learning and Convolutional Neural Network Methods for Kazakh Sign Language // — *Вестник Scientific Journal of Astana IT Univers ity.* — 2023. — Vol. 15. — Pp. 16–27.

Mukhanov S.B., Lee A.S., Zheksenov D.B., Yevdokimov D.D., Amirgaliev E.N., Kalzhigitov N.K., Kenshimov Sh. (2023). Comparative analysis of neural network models for gesture recognition methods hands // Bulletin of NIA RK. Information and communication technologies. — 2023. — No. 2(88). — Pp. 15–27.

Uskenbayeva R.K. & Mukhanov S.B. (2020). Contour analysis of external images. Proceedings of the 6th International Conference on Engineering & MIS 2020. — 1–6. https://doi.org/10.1145/3410352.3410811.

Wang Y., Wang H. & He X. (2020). Sign language recognition based on deep convolutional neural network". — IEEE Access. — 8. — 64990–64999. 2020. https://doi.org/10.3390/electronics12040786.

Yuanguo Zhou, Shan Shui, Yijun Cai, Chengying Chen, Yingshi Chen, Reza Abdi-Ghaleh (2023). An improved all-optical diffractive deep neural network with less parameters for gesture recognition // — *Journal of Visual Communication and Image Representation.* — 2023. — Vol. 90. — P. 103688.

Yeo H.S., Lee B.G., Lim H. (2013). Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware // Multimed. Tools Appl. — 2013. https://link.springer.com/article/10.1007/s11042-013-1501-1 01.11.2022.

ХАЛЫҚАРАЛЫҚ АҚПАРАТТЫҚ ЖӘНЕ КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР ЖУРНАЛЫ

МЕЖДУНАРОДНЫЙ ЖУРНАЛ ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Правила оформления статьи для публикации в журнале на сайте:
https://journal.iitu.edu.kz
ISSN 2708–2032 (print)
ISSN 2708–2040 (online)
Собственник: АО «Международный университет информационных технологий» (Казахстан, Алматы)

ОТВЕТСТВЕННЫЙ РЕДАКТОР
Мрзабаева Раушан Жалиқызы

КОМПЬЮТЕРНАЯ ВЕРСТКА
Асанова Жадыра

Подписано в печать 14.09.2024.
Формат 60х881/8. Бумага офсетная. Печать - ризограф. 9,0 п.л. Тираж 100
050040 г. Алматы, ул. Манаса 34/1, каб. 709, тел: +7 (727) 244-51-09).

Издание Международного университета информационных технологий
Издательский центр КБТУ, Алматы, ул. Толе би, 59