EFFICIENT IMPLEMENTATION OF NEURAL NETWORK LEARNING ALGORITHMS USING THE CONCEPT OF A Q-DETERMINANT

V. N. Aleeva, A. S. Sapozhnikov

South Ural State University (National Research University),
454080, Chelyabinsk, Russia

DOI: 10.24412/2073-0667-2025-3-5-16
EDN: NGOUCS

The paper is the first to consider an efficient implementation of neural network learning algorithms using the concept of a Q-determinant. Let's describe the necessary information about the concept of the Q-determinant. These are the following notions.

Let B be the input data of the algorithm, and Q be the operations used by the algorithm. An expression over B and Q is a term in the standard sense of mathematical logic. A chain of length n is the result of applying some associative operation from Q to n expressions. Let's define an algorithm for solving an algorithmic problem with a set of parameters of dimension N. If this problem has no dimension parameters, then $N = \varnothing$. Otherwise, $N = \{n_1, \ldots, n_k\}$ is a set of dimension parameters for this problem. Let the set $\bar{N} = \{\bar{n}_1, \ldots, \bar{n}_k\}$ consists of the specified values of the dimension parameters and $\{\bar{N}\}$ is the set of all such tuples of $\bar{N}$.

Now let's define the concept of a Q-term, which can be unconditional, conditional and conditional infinite. If $N = \varnothing$, then any expression w over B and Q is an unconditional Q-term. If $N \neq \varnothing$ and V is the set of all expressions over B and Q, then any mapping $w : \{\bar{N}\} \to V \cup \varnothing$ is also called an unconditional Q-term. $w(\bar{N}) = \varnothing$ means that the value of $w(\bar{N})$ is undefined. A conditional Q-term consists of a finite or countable set of pairs of unconditional Q-terms, and in each pair the first Q-term has a logical type. Therefore, they will be called logical Q-terms. If the number of pairs in a Q-term is infinite, then it is called an infinite conditional Q-term.

We can calculate the value of the Q-term given the input data. Let m be the number of output variables. Let the algorithm calculate the values of each output variable y_i ($i \in \{1, \ldots, m\}$) if the value of the corresponding Q-term f_i ($i \in \{1, \ldots, m\}$) is calculated. Then the set of Q-terms $\{f_i\}_{i \in \{1,\ldots,m\}}$ is called the Q-determinant of the algorithm. The system of equations $y_i = f_i$ ($i \in \{1, \ldots, m\}$) is called the representation of the algorithm in the form of a Q-determinant.

The process of computing Q-terms $\{f_i\}_{i \in \{1,\ldots,m\}}$ is called an implementation of algorithm. An implementation of an algorithm is called parallel if there are operations that are executed simultaneously. An implementation of an algorithm is called Q-effective if Q-terms $\{f_i\}_{i \in \{1,\ldots,m\}}$ are computed simultaneously, operations are executed as they are ready, and chain operations are computed using the doubling scheme. A Q-effective implementation of the algorithm uses parallelism resource of this algorithm completely. The height and width of the algorithm characterize its parallelism resource. If a finite number of operations are performed simultaneously during the implementation of the algorithm, then this implementation of the algorithm is called realizable.

In this paper we describe a method for designing Q-effective programs that use the parallelism resource of algorithms completely. This method is used for effective implementation of algorithms.

It has three steps: construction of the Q-determinant of the algorithm, description of the Q-effective implementation of the algorithm, development of a program for an realizable Q-effective implementation of the algorithm. A program is called Q-effective if it is developed using this method. A program is also called Q-effective if it performs a Q-effective implementation of an algorithm. The same set of programs corresponds to these two definitions.

The application of the method of designing Q-effective programs is shown on the example of algorithms implementing stochastic gradient descent and error back propagation methods. These methods are often used to learn neural networks. Q-effective programs for shared and distributed memory parallel computing systems have been developed that implement these methods. The acceleration and efficiency of the developed programs have been evaluated using computational experiments. Computational experiments were performed on the supercomputer «Tornado» of the South Ural State University. We present conclusions based on the obtained evaluation of the dynamic characteristics of the developed programs. The values of the dynamic characteristics of a Q-effective program depend on the implemented algorithm and the conditions of development and execution of the program. The paper provides a recommendation to the developer of a Q-effective program in the case where he wants to improve the values of the dynamic characteristics of the program being developed.

Therefore, the research shows that the method of designing Q-effective programs can be applied to efficiently implement neural network learning algorithms.

Key words: neural network learning, stochastic gradient descent method, error back propagation method, Q-determinant of algorithm, Q-effective implementation of algorithm, Q-effective program.

References

1. Aleeva V.N. Analiz parallel'nyx chislennyx algoritmov. Preprint N 590. Novosibirsk: VC SO AN SSSR, 1985. 23 s. (in Russian)

2. Valentina Aleeva, Rifkhat Aleev. Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. V. 10. N 2, Article number 8. P. 1–64. DOI: 10.1145/3583755.

3. Ershov YU. L., Palyutin E. A. Matematicheskaya logika. M.: Nauka, 1987. 336 s. (in Russian)

4. Aleeva V. N. Improving Parallel Computing Efficiency // Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.

5. Aleeva V. Designing a Parallel Programs on the Base of the Conception of Q-Determinant // Supercomputing. RuSCDays 2018. Communications in Computer and Information Science. 2019. Vol. 965. P. 565–577. DOI: 10.1007/978-3-030-05807-4_48.

6. Gudfellou Ya, Bendzhio I., Kurvill' A. Glubokoe obuchenie. M.: DMK Press, 2018. 652 s. (in Russian)

7. Nielsen M. A. Neural Networks and Deep Learning. [Electron. Res.]: `http://neuralnetworksanddeeplearning.com/chap2.html`. Accessed: 11.02.2025.

8. Nikolenko S. I., Kadurin A. A., Arxangelskaya E. O. Glubokoe obuchenie. SPb.: Piter, 2018. 480 s. (in Russian)

9. Superkomp'uter "Tornado YuUrGU". [Electron. Res.]: `http://supercomputer.susu.ru/computers/tornado/`. Accessed: 11.02.2025. (in Russian)

10. Otkrytaya enciklopediya svojstv algoritmov. [Electron. Res.]: `https://algowiki-project.org/ru`. Accessed: 11.02.2025. (in Russian)

ЭФФЕКТИВНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ С ПОМОЩЬЮ КОНЦЕПЦИИ Q-ДЕТЕРМИНАНТА

В. Н. Алеева, А. С. Сапожников

Южно-Уральский государственный университет (НИУ),
454080, Челябинск, Россия

УДК 004.021, 004.032.24, 004.051, 004.272
DOI: 10.24412/2073-0667-2025-3-5-16
EDN: NGOUCS

В статье впервые рассматривается эффективная реализация с помощью концепции Q-детерминанта алгоритмов обучения нейронных сетей. Для эффективной реализации алгоритмов применяется метод проектирования Q-эффективных программ, использующих ресурс параллелизма реализуемых ими алгоритмов полностью. Применение метода показано на примере алгоритмов, выполняющих часто используемые методы стохастического градиентного спуска и обратного распространения ошибки. Для этих алгоритмов разработаны Q-эффективные программы для общей и распределенной памяти параллельных вычислительных систем. С помощью вычислительных экспериментов выполнена оценка ускорения и эффективности разработанных программ. Вычислительные эксперименты проводились на суперкомпьютере «Торнадо» Южно-Уральского государственного университета.

Ключевые слова: обучение нейронных сетей, метод стохастического градиентного спуска, метод обратного распространения ошибки, Q-детерминант алгоритма, Q-эффективная реализация алгоритма, Q-эффективная программа.

Введение. Проблема эффективной реализации алгоритмов, в том числе, используемых для решения задач искусственного интеллекта, является актуальной. Подход, основанный на концепции Q-детерминанта, является одним из подходов к решению этой проблемы. В рамках данного подхода был разработан метод проектирования Q-эффективных программ, использующих ресурс реализуемых алгоритмов в полной мере. Исследование, описанное в данной статье, является первым исследованием по эффективной реализации алгоритмов, применяемых для решения задач искусственного интеллекта.

Цель исследования, описанного в статье, заключается в том, чтобы показать применение метода проектирования Q-эффективных программ к алгоритмам обучения нейронных сетей. Она предполагает решение следующих задач.

— Разработка Q-эффективных программ для общей и распределенной памяти параллельных вычислительных систем (ПВС), реализующих методы стохастического градиентного спуска и обратного распространения ошибки.

— Оценка ускорения и эффективности разработанных Q-эффективных программ с помощью вычислительных экспериментов на ПВС.

Статья организована следующим образом. Раздел 1 содержит теоретические основы исследования.

В разделе 2 показано применение метода проектирования Q-эффективных программ для алгоритмов обучения нейронных сетей. В разделе 3 описаны разработка и экспериментальное исследование Q-эффективных программ. В заключении подводятся итоги исследования и формулируется направление дальнейших исследований.

1. Теоретические основы исследования. Исследования данной статьи базируются на концепции Q-детерминанта, которая впервые была изложена в работе [1]. В дальнейшем она развивалась и в настоящее время наиболее полно представлена в работе [2]. Опишем кратко понятия концепции Q-детерминанта, используемые в данной работе.

Определение 1. Выражение над множеством входных данных B алгоритма и множеством операций Q, используемых алгоритмом, определим, как терм в стандартном смысле математической логики [3].

Определение 2. Цепочкой длины n будем называть выражение, представляющее собой результат применения некоторой ассоциативной операции из Q к n выражениям.

Пусть $N = \{n_1, \ldots, n_k\}$ — множество параметров размерности алгоритмической проблемы, решаемой алгоритмом. Через $\bar{N} = \{\bar{n}_1, \ldots, \bar{n}_k\}$ обозначим кортеж, где $\bar{n}_i$ — некоторое заданное значение параметра n_i для каждого $i \in \{1, \ldots, k\}$, а через $\{\bar{N}\}$ множество всех кортежей $\bar{N}$. Алгоритмическая проблема может не иметь параметров размерности. В этом случае $N = \varnothing$.

Определим понятие Q-терма. Q-термы могут быть безусловными, условными и условными бесконечными.

Определение 3. Если $N = \varnothing$, то любое выражение w над B и Q называется безусловным Q-термом. Пусть $N \neq \varnothing$ и V — множество всех выражений над B и Q. Любое отображение $w : \{\bar{N}\} \to V \cup \varnothing$ также называется безусловным Q-термом. Здесь $w(\bar{N}) = \varnothing$ означает, что значение $w(\bar{N})$ не определено.

Условные Q-термы состоят из конечного множества пар, а условные бесконечные Q-термы из бесконечного множества пар безусловных Q-термов. Первые безусловные Q-термы пар принимают значения логического типа, поэтому называются логическими Q-термами.

Q-термы можно вычислять.

Определение 4. Если алгоритм состоит в том, что для вычисления значения каждой выходной переменной y_i ($i \in \{1, \ldots, m\}$) нужно вычислить значение соответствующего Q-терма f_i ($i \in \{1, \ldots, m\}$), где m — количество выходных переменных, то множество Q-термов $\{f_i\}_{i \in \{1,\ldots,m\}}$ называется Q-детерминантом алгоритма.

Определение 5. Система уравнений $y_i = f_i$ ($i \in \{1, \ldots, m\}$) называется представлением алгоритма в форме Q-детерминанта.

Определение 6. Процесс вычисления Q-термов $\{f_i\}_{i \in \{1,\ldots,m\}}$ называется реализацией алгоритма.

Определение 7. Реализация алгоритма называется параллельной, если существуют операции, которые выполняются одновременно.

Определение 8. Реализация алгоритма называется Q-эффективной, если Q-термы $\{f_i\}_{i \in \{1,\ldots,m\}}$ вычисляются одновременно, операции при их вычислении выполняются по мере готовности, при этом, если несколько операций цепочки готовы к выполнению, то они выполняются по схеме сдваивания.

Замечание 1. Определение Q-эффективной реализации показывает, что она полностью использует ресурс параллелизма алгоритма.

Ресурс параллелизма алгоритма характеризуют его высота и ширина. Эти понятия рассматриваются в работах [2, 4].

Определение 9. Реализация алгоритма называется выполнимой, если одновременно должно выполняться конечное число операций.

Метод проектирования Q-эффективных программ состоит из трех этапов. *Этап* 1. Построение Q-детерминанта алгоритма. *Этап* 2. Описание Q-эффективной реализации алгоритма. *Этап* 3. Разработка программы для выполнимой Q-эффективной реализации алгоритма.

Определение 10. Программа называется Q-эффективной, если она разработана с помощью данного метода.

Дадим еще одно определение Q-эффективной программы.

Определение 11. Программа называется Q-эффективной, если она выполняет Q-эффективную реализацию алгоритма.

Определениям 10 и 11 соответствует одно и то же множество программ. Итак, понятие Q-эффективной программы можно определять как с помощью определения 10, так и с помощью определения 11.

Подробно метод проектирования Q-эффективных программ изложен в работах [4, 5].

2. Применение метода проектирования Q-эффективных программ. Метод проектирования Q-эффективных программ был применен для эффективной реализации методов стохастического градиентного спуска (СГС) [6] и обратного распространения ошибки [7], используемых для обучения нейронных сетей.

Нейронная сеть — это математическая модель, построенная по принципу организации биологических нейронных сетей. Существуют разные типы архитектур нейронных сетей. Под архитектурой понимается общая структура сети: сколько в ней должно быть блоков (по-другому, слоев) и как эти блоки связаны между собой [6].

Для данного исследования была выбрана полносвязная нейронная сеть. Она состоит из входного слоя, одного или нескольких скрытых слоев и выходного слоя. При этом все нейроны текущего слоя связаны с нейронами предыдущего слоя. Будем обозначать номер текущего слоя через l, а номер последнего слоя через L. L соответствует количеству слоев в нейронной сети.

Связь между нейронами разных слоев выражается в виде синаптических весов. Для текущего слоя l все связи будут выражены в виде матрицы синаптических весов

$$W^{(l)} = \left[w_{ij}^{(l)}\right]_{i=1,\ldots,h;j=1,\ldots,r},$$

где h — количество нейронов в слое $l-1$, r — количество нейронов в слое l.

Помимо этого, у каждого нейрона слоя l есть смещение. Для всего слоя l смещение можно представить в виде вектора

$$B^{(l)} = (b_1^{(l)}, \ldots, b_r^{(l)}).$$

Под обучением нейронной сети понимают подбор весов $W^{(l)}$ и смещений $B^{(l)}$ таким образом, чтобы увеличить точность работы нейронной сети при выполнении текущей задачи.

Чтобы провести обучение, необходимо сначала определить точность нейронной сети. Для этого применяют метод прямого распространения, заключающийся в следующем. Для каждого слоя, кроме первого, рассчитывается активационный потенциал $Z^{(l)}$ по формуле

$$Z^{(l)} = A^{(l-1)} \times W^{(l)} + B^{(l)},$$

где $A^{(l-1)} = (a_1^{(l-1)}, \ldots, a_h^{(l-1)})$ — вектор выходных сигналов предыдущего слоя.

Вектор выходных сигналов текущего слоя может быть рассчитан следующим образом

$$A^{(l)} = activation(Z^{(l)}),$$

где $A^{(l)} = (a_1^{(l)}, \ldots, a_r^{(l)})$, $activation$ — функция активации нейрона.

В зависимости от целей обучения, функция активации [8] может быть разной. Для данных исследований применяется сигмоида, которая вычисляется по формуле

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Метод прямого распространения завершается, когда будет вычислен вектор выходных сигналов A^L последнего слоя L.

Для обучения используется обучающая выборка. Она состоит из некоторого количества образцов. Каждый образец имеет свой набор характеристик X и вектор меток класса y. Перед началом обучения вся выборка разделяется на несколько подвыборок (мини-пакетов), причем размер у всех подвыборок одинаков и равен v.

На каждом шаге обучения (по-другому, эпохе) нейронная сеть с помощью метода прямого распространения вычисляет вектор выходных сигналов последнего слоя A^L для каждого образца мини-пакета. После этого производится расчет точности работы нейронной сети. Точность работы нейронной сети оценивается разными метриками. Для данного исследования была выбрана среднеквадратичная ошибка

$$C = \frac{1}{v} \sum_{i \in \{1,\ldots,v\}} \frac{|A^L - y|_i^2}{2},$$

где v — размер мини-пакета, $y = (y_1, \ldots, y_r)$ — вектор меток класса для образца i.

С помощью метода СГС можно осуществить подбор новых весов и смещений на основе точности

$$W^{*(l)} = W^{(l)} - \epsilon \cdot \nabla_W C^{(l)},$$
$$B^{*(l)} = B^{(l)} - \epsilon \cdot \nabla_B C^{(l)},$$

где ϵ — скорость обучения, положительный скаляр, определяющий длину шага; $\nabla_W C^{(l)}$ и $\nabla_B C^{(l)}$ — матрицы частных производных целевой функции C.

Для вычисления матриц частных производных целевой функции C воспользуемся методом обратного распространения ошибки.

После обработки нейронной сетью одного набора входных значений X вычисляется мера влияния нейронов выходного слоя на величину ошибки δ^L по формуле

$$\delta^L = \frac{\partial C}{\partial A^L} \cdot \sigma'(Z^L). \tag{1}$$

Для среднеквадратической ошибки формула (1) выглядит следующим образом

$$\delta^L = (A^L - y) \cdot \sigma'(Z^L).$$

Далее рассчитывается мера влияния нейронов каждого слоя l от $L-1$ слоя и до первого по формуле

$$\delta^{(l)} = \sigma'(Z^{(l)}) \cdot (\delta^{(l+1)} \cdot W^{(l+1)T}).$$

По полученным значениям рассчитываются градиенты весов и смещений для каждого слоя по формулам

$$\nabla_W C^{(l)} = \delta^{(l)} \cdot A^{(l-1)}, \tag{2}$$

$$\nabla_B C^{(l)} = \delta^{(l)}. \tag{3}$$

Опишем применение к рассмотренным методам обучения нейронных сетей метода проектирования Q-эффективных программ.

Этап 1. Q-детерминант метода СГС с учетом формул (2) и (3) представляет собой два множества безусловных Q-термов

$$w_{ij}^{(l)*} = w_{ij}^{(l)} - \epsilon \cdot \delta_i^{(l)} \cdot a_j^{(l-1)},$$
$$b_i^{(l)*} = b_i^{(l)} - \epsilon \cdot \delta_i^{(l)},$$

где $i \in \{1,\ldots,r\}$, r — количество нейронов на слое l, $j \in \{1,\ldots,h\}$, h — количество нейронов на слое $l-1$.

Q-детерминант метода обратного распространения ошибки состоит из одного множества безусловных Q-термов

$$\delta_i^{(l)} = \sigma'(z_i^{(l)}) \cdot (\delta_j^{(l+1)} \cdot w_{ji}^{(l+1)}),$$

где $i \in \{1,\ldots,r\}$, r — количество нейронов на слое l, $j \in \{1,\ldots,k\}$, k — количество нейронов на слое $l+1$.

Этап 2. Опишем Q-эффективную реализацию алгоритмов, выполняющих исследуемые методы. В методе СГС веса

$$\left\{\{w_{11}^*,\ldots,w_{1j}^*\},\ldots,\{w_{i1}^*,\ldots,w_{ij}^*\}\right\},$$

где $i \in \{1,\ldots,h\}$ — количество нейронов на слое $l-1$, $j \in \{1,\ldots,r\}$ — количество нейронов на слое l, будут вычисляться одновременно. Аналогичным образом поступим и со смещениями $\{b_1^*,\ldots,b_i^*\}$, где $i \in \{1,\ldots,r\}$.

В методе обратного распространения ошибки меру влияния нейронов на величину ошибки $\{\delta_1^{(l)},\ldots,\delta_i^{(l)}\}$, где $i \in \{1,\ldots,r\}$. будем вычислять одновременно.

Этап 3. Описанные на этапе 2 Q-эффективные реализации можно использовать для разработки Q-эффективных программ для систем с общей памятью. Опишем процесс реализации методов для системы с распределенной памятью с использованием принципа “master-slave”. Узел “master” обозначим через M, а узлы “slave” через S.

Общими данными для методов СГС и обратного распространения ошибки являются матрица весов W, вектор выходных сигналов нейронов A и вектор активационных потенциалов Z. Так как вектор A может быть получен путем применения функции активации к

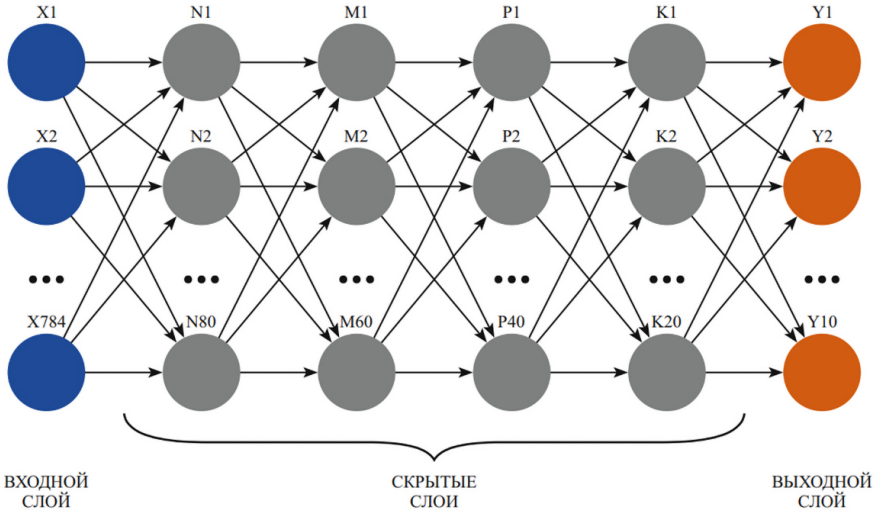

Рис. 1. Архитектура нейронной сети

вектору Z, общими данными остаются только матрица весов W и вектор активационных потенциалов Z.

Данные между узлами S будут распределены следующим образом. Матрица W будет разделена на строки, и каждому узлу S будет передано некоторое количество строк, обозначим это количество через t. Вектор Z будет также разделен, и каждому узлу S достанется t элементов.

Теперь перейдем к частным данным, которые нужны каждому методу отдельно. Метод СГС также использует вектор смещений B, размерность которого такая же, как и у вектора Z. Поэтому вектор B будет распределен между узлами S перед началом вычислений аналогично вектору Z. Вектор меры влияния нейронов на величину ошибки, обозначим его через $\vec{\delta}$, необходимо передать всем узлам S целиком, так как этого требуют вычисления. Передача всех общих и частных данных для метода СГС будет осуществляться либо перед началом обучения нейронной сети, либо перед началом работы метода.

Для метода обратного распространения ошибки требуется вектор меры влияния нейронов на величину ошибки $\vec{\delta}$, полученный из предыдущей итерации этого же метода, причем в полном объеме. Распределим вычисление вектора $\vec{\delta}$ так, чтобы каждый узел вычислял некоторое количество элементов этого вектора, а в конце каждой итерации все элементы были собраны узлом M в один вектор $\vec{\delta}$ и распределены снова по всем узлам S. Так как к моменту начала метода СГС все векторы $\vec{\delta}$ будут вычислены и пересланы всем узлам, дополнительно пересылать их не придется.

3. Разработка и экспериментальное исследование Q-эффективных программ. Для выполнения описанных Q-эффективных реализаций метода СГС и метода обратного распространения ошибки были разработаны Q-эффективные программы для систем с общей и распределенной памятью.

Для разработки Q-эффективных программ применялся язык программирования C++. Кроме того, для систем с общей памятью использовалась технология OpenMP, а для систем с распределенной памятью технологии OpenMP и MPI. При тестировании разработанных программ использовалась нейронная сеть прямого распространения, архитектура которой представлена на рис. 1.

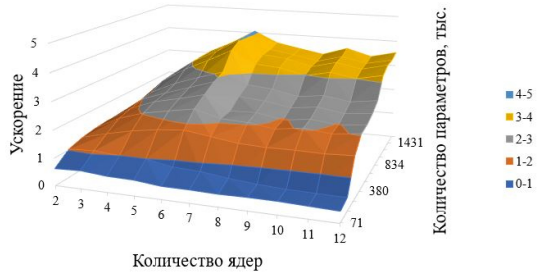

Рис. 2. Ускорение Q-эффективной программы для метода обратного распространения ошибки для системы с общей памятью

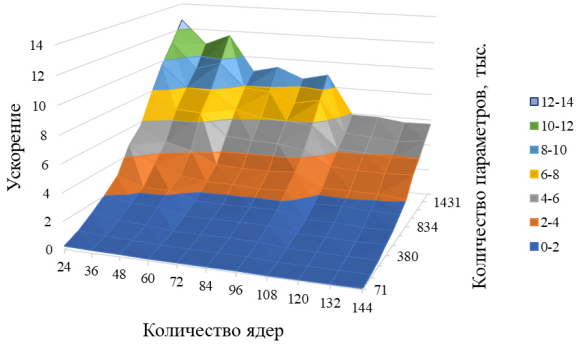

Рис. 3. Ускорение Q-эффективной программы для метода обратного распространения ошибки для системы с распределенной памятью

Вычислительные эксперименты были проведены на суперкомпьютере «Торнадо» Южно-Уральского государственного университета [9]. Для выполнения Q-эффективных программ для общей памяти использовался один вычислительный узел, который содержит два центральных процессора Intel Xeon X5680 с частотой 3.33 GHz, каждый из которых имеет 6 ядер и поддерживает 12 потоков, оперативная память узла 24 Гб ECC DDR3 Full buffered. Для выполнения Q-эффективных программ для распределенной памяти использовалось от 2-х до 12 вычислительных узлов.

Разработанные Q-эффективные программы полностью используют ресурс параллелизма алгоритмов. С помощью вычислительных экспериментов были оценены их ускорение и эффективность [10]. Ускорение программы вычислялось по формуле

$$S_p = \frac{T_1}{T_p},$$

здесь T_1 — время выполнения программы на одном вычислительном ядре, T_p — время выполнения программы на p вычислительных ядрах. Для оценки эффективности программы использовалась формула

$$E_p = \frac{S_p}{p},$$

где S_p — ускорение программы, p — количество используемых вычислительных ядер.

На рисунках 2 и 3 представлены графики зависимости ускорения Q-эффективных программ для метода обратного распространения ошибки от количества используемых ядер и параметров нейронной сети (сумма всех смещений и весов всех слоев) для систем с общей и распределенной памятью.

Для Q-эффективных программ, реализующих метод СГС, аналогичные графики показаны на рисунках 4 и 5 соответственно.

На рисунках 6 и 7 показана зависимость эффективности Q-эффективных программ для метода обратного распространения ошибки от количества ядер и параметров нейронной сети для систем с общей и распределенной памятью.

Для Q-эффективных программ для метода СГС аналогичные графики показаны на рисунках 8 и 9 соответственно.

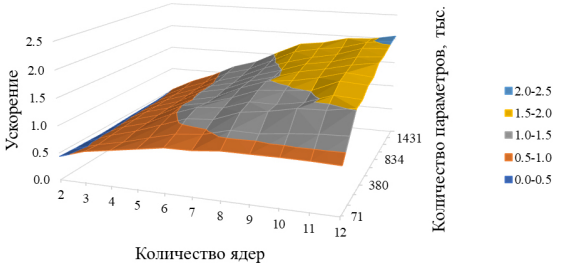

Рис. 4. Ускорение Q-эффективной программы для метода СГС для системы с общей памятью

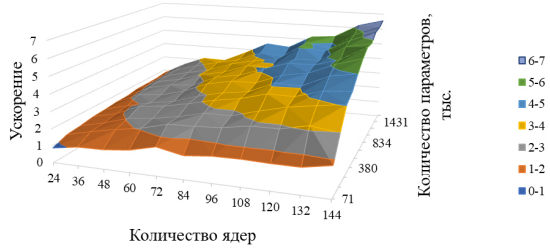

Рис. 5. Ускорение Q-эффективной программы для метода СГС для системы с распределенной памятью

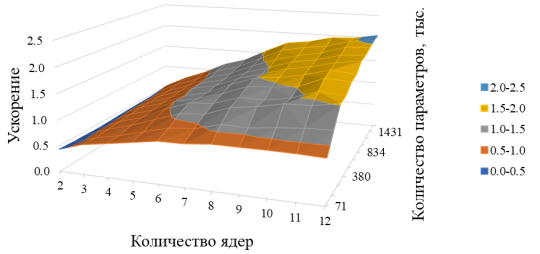

Рис. 6. Эффективность Q-эффективной программы для метода обратного распространения ошибки для системы с общей памятью

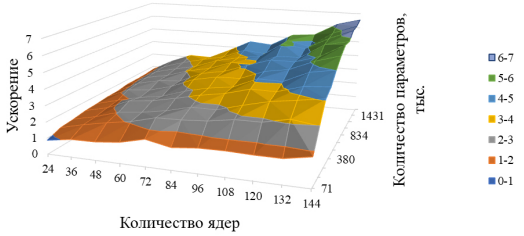

Рис. 7. Эффективность Q-эффективной программы для метода обратного распространения ошибки для системы с распределенной памятью

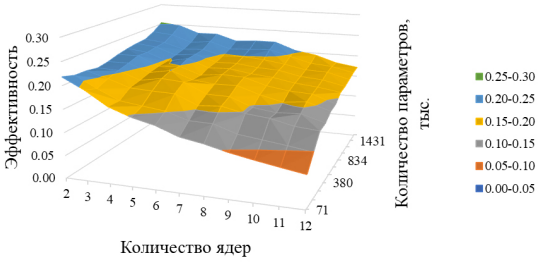

Рис. 8. Эффективность Q-эффективной программы для метода СГС для системы с общей памятью

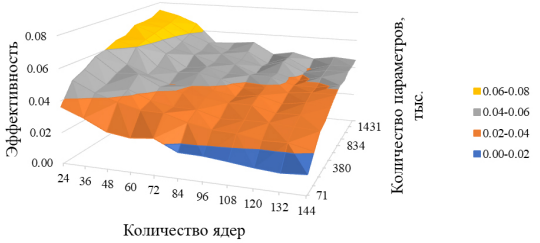

Рис. 9. Эффективность Q-эффективной программы для метода СГС для системы с распределенной памятью

Приведем некоторые выводы по результатам экспериментов. Исследования на основе концепции Q-детерминанта [2] показывают, что ускорение и эффективность Q-эффективной программы зависят от ресурса параллелизма реализуемого алгоритма и от вычислительной инфраструктуры программы — условий ее разработки и выполнения. Влиять на исследуемые характеристики разработанных Q-эффективных программ могут любые составляющие их вычислительной инфраструктуры.

Как можно заметить, ускорение Q-эффективной программы для метода обратного распространения ошибки не зависимо от того, имеет система общую или распределенную память, со временем выходит на плато, в то время как для Q-эффективной программы для метода СГС совсем иная ситуация: ускорение продолжает расти.

По графикам ускорения метода обратного распространения ошибки (рисунки 2 и 3) можно сделать вывод, что этот метод полностью исчерпывает свой ресурс параллелизма при условиях экспериментов. Иначе говоря, начиная с определенного момента, добавление новых ресурсов не приводит к их использованию программой, так как ресурс параллелизма реализуемого программой алгоритма не позволяет это сделать. В то же время показанный на рисунках 4 и 5 рост ускорения метода СГС на системах с общей и распределенной памятью объясняется тем, что метод СГС не использует полностью свой ресурс параллелизма при условиях экспериментов.

Исследование влияния на характеристики разработанных Q-эффективных программ конкретных составляющих их вычислительной инфраструктуры в данной работе не предусматривалось. Например, для метода обратного распространения ошибки падение ускорения на системе с распределенной памятью может быть связано с наличием пересылки данных между вычислительными узлами во время каждой итерации метода. При реализации метода СГС пересылки данных нет, поэтому она не входит в вычислительную инфраструктуру Q-эффективных программ.

Заключение. В статье описано первое исследование по эффективной реализации алгоритмов с помощью концепции Q-детерминанта, проводимое для решения задач искусственного интеллекта. Исследование заключается в том, что показано применение к алгоритмам обучения нейронных сетей метода проектирования Q-эффективных программ, использующих ресурс параллелизма реализуемых алгоритмов полностью. В результате были разработаны Q-эффективные программы для общей и распределенной памяти ПВС, выполняющие методы стохастического градиентного спуска и обратного распространения ошибки. Кроме того, оценены ускорение и эффективность разработанных Q-эффективных программ с помощью вычислительных экспериментов на ПВС.

Метод проектирования Q-эффективных программ констатирует факт, каковы значения характеристик разработанных программ для исследуемого алгоритма и вычислительных инфраструктур программ. Однако следует помнить, что в случае, если значения характеристик Q-эффективной программы не устраивают, их можно улучшить, либо изменив вычислительную инфраструктуру программы, либо заменив алгоритм другим алгоритмом с меньшей высотой. Возможны также и оба изменения одновременно.

Успешное применение концепции Q-детерминанта для исследования эффективной реализации описанных в статье алгоритмов открывает перспективы для исследования и других алгоритмов, применяемых для решения задач искусственного интеллекта.

Исходный код разработанных Q-эффективных программ доступен по URL-адресу: `https://github.com/Snezinka/Parallel-neural-network`.

Список литературы

1. Алеева В.Н. Анализ параллельных численных алгоритмов. Препринт № 590. Новосибирск: ВЦ СО АН СССР, 1985. 23 с.

2. Valentina Aleeva, Rifkhat Aleev. Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. Vol. 10. N 2, Article number 8. P. 1–64. DOI: 10.1145/3583755.

3. Ершов Ю. Л., Палютин Е. А. Математическая логика. М.: Наука, 1987. 336 с.

4. Aleeva V. N. Improving Parallel Computing Efficiency // Proceedings — 2020 Global Smart Industry Conference, GloSIC 2020. IEEE. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.

5. Aleeva V. Designing a Parallel Programs on the Base of the Conception of Q-Determinant // Supercomputing. RuSCDays 2018. Communications in Computer and Information Science. 2019. V. 965. P. 565–577. DOI: 10.1007/978-3-030-05807-4_48.

6. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. М.: ДМК Пресс, 2018. 652 с.

7. Nielsen M. A. Neural Networks and Deep Learning [Электронный ресурс]: http://neuralnetworksanddeeplearning.com/chap2.html. Дата обращения: 11.02.2025.

8. Николенко С. И., Кадурин А. А., Архангельская Е. О. Глубокое обучение. СПб.: Питер, 2018. 480 с.

9. Суперкомпьютер «Торнадо ЮУрГУ». [Электронный ресурс]: http://supercomputer.susu.ru/computers/tornado/. Дата обращения: 11.02.2025.

10. Открытая энциклопедия свойств алгоритмов. [Электронный ресурс]: https://algowiki-project.org/ru. Дата обращения: 11.02.2025.

Алеева Валентина Николаевна — e-mail: aleevavn@susu.ru; тел.: +7-351-267-90-89. В 1972 году окончила Новосибирский государственный университет с присвоением квалификации «Математика, прикладная математика». В 1986 году решением Совета при Вычислительном центре СО АН СССР ей присуждена ученая степень кандидата физико-математических наук. В настоящее время она является доцентом кафедры системного программирования Южно-Уральского государственного университета (Национальный исследовательский университет).

Имеет 2 изобретения по архитектуре параллельных вычислительных систем и более 40 публикаций по распараллеливанию численных алгоритмов. Научные интересы: архитектура вычислительных систем, ресурсы параллелизма алгоритмов, проектирование эффективных программ, эффективность параллельных вычислений, в том числе в автоматизированном режиме.

Valentina Nikolaevna Aleeva — e-mail: aleevavn@susu.ru; tel.: +7-351-267-90-89. Graduated from Novosibirsk State University with the Master of Mathematics and Applied Mathematics degree in 1972 and defended his candidate dissertation in physical and mathematical sciences in Computing Centre of SB RAS in 1986. Currently, she is an Associate Professor of the Department of System Programming at South Ural State University (National Research University).

She has 2 inventions on the architecture of parallel computing systems and more than 40 publications on parallelization of numerical algorithms. The scientific interests include the architecture of computing systems, parallelism resources of algorithms, the design of effective programs, the efficiency of parallel computing, particularly in automated mode.

Сапожников Андрей Сергеевич — e-mail: eikar@bk.ru; тел.: +7-951-467-66-27. В 2024 году окончил Южно-У-ральский государственный университет (национальный исследовательский университет) и получил степень магистра.

Научные интересы: параллельные вычисления, искусственные нейронные сети и машинное обучение.

Sapozhnikov Andrey Sergeevich — e-mail: eikar@bk.ru; tel.: +7-951-467-66-27. Graduated from South Ural State University (National Research University) in 2024 and received a master's degree.

The scientific interests include parallel computing, artificial neural networks and machine learning.

Дата поступления — 14.02.2025