IRSTI 28.23.01 https://doi.org/10.26577/jpcsit2023v1i4a2

N.M. Kassymbek[1*], A.S. Rakhimzhanova[2]

[1]U.A. Joldasbekov Institute of Mechanics and Engineering, Almaty, Kazakhstan
[2]Astana IT University, Astana, Kazakhstan
*e-mail: nuryslam.qassymbek@gmail.com

COMPARATIVE STUDY OF PARALLEL ALGORITHMS FOR MACHINE LEARNING METHODS

Abstract. In the modern world, as the amount of data used in machine learning is constantly growing, the task of accelerating the training of models on large datasets becomes relevant. To solve this problem, methods of parallel data processing are used. This paper discusses methods of parallel data processing for machine learning. Linear regression and random forest are considered as machine learning methods. Parallel algorithms based on the MPI interface were developed for each method. The results of the experiments showed that both methods give acceleration compared to the sequential algorithm. However, the acceleration in the case of random forest was significantly higher than in the case of linear regression. This is because random forest is a more computationally efficient method than linear regression. Therefore, it can be concluded that Random Forest is the most effective machine learning approach for parallel data processing. This statement is confirmed by the results of experiments conducted in this work. Overall, the experimental results show that the use of parallel algorithms in machine learning can significantly speed up model training when working with large data sets. Random forest is the most efficient method for parallel data processing, as it is more computationally efficient and has higher scalability.

Key words: Machine Learning, Linear Regression, Random Forest, Parallel Computing.

1 Introduction

Machine learning (ML) is a branch of artificial intelligence dedicated to creating algorithms that can learn from data without the need for explicit programming [1,2]. These algorithms find applications in various fields such as image recognition, natural language processing, and predictive analysis.

One of the most widely used ML methods is linear regression [3]. Linear regression is employed to forecast continuous values using a dataset containing pairs of independent and dependent variables [4]. Traditional linear regression training operates sequentially, and for substantial datasets, it can be computationally intensive. This is attributed to the quadratic nature of the matrix operation involved in the normal function, utilized to calculate the weights for linear regression, which scales with the number of observations in the dataset.

The random forest serves as an ensemble machine learning method applied to both classification and regression tasks. It involves numerous decision trees trained on diverse subsets of the dataset [5,6]. Conventional random forest training is likewise a sequential procedure, and its computational demands can be high, particularly for sizable datasets. This is due to the necessity of computing numerous data splits for each decision tree.

Parallel programming is a software development methodology that allows multiple tasks to be run simultaneously on multiple processors or computing nodes [7,8]. Parallel computing can significantly speed up the execution of tasks that can be divided into multiple independent parts. The importance of parallel computing in ML is due to the growth in the size and complexity of datasets used in ML applications. Large datasets can require significant computational resources to train ML algorithms. Parallel computing can help to speed up the training process and make it more accessible [9].

Parallel learning in machine learning is an urgent task. There are various approaches to parallelization of machine learning methods, for example, in [10-12] the authors developed a parallel algorithm for the linear regression method. In [10,12], an implementation was given for multithreaded systems, and in [11] for systems with distributed memory. A similar problem, but for the method of support vectors for regression, can be seen in [13,14]. In these papers, the authors used

the MPI (Message Passing Interface) standard for parallelization. And in the works [15,16] one can see the use of the same standard, but for the Random Forest method. These works show the relevance of the problem under consideration.

In this paper, we present a parallel linear regression training algorithm that uses MPI to parallelize the matrix operations of the normal function. Our algorithm can significantly speed up the training of linear regression for large datasets.

2 Linear regression

Linear regression stands out as one of the most extensively utilized machine learning methods, employed for predicting continuous values using a dataset that comprises pairs of independent and dependent variables. Linear regression has many applications in various fields, including:

- Demand forecasting
- Price forecasting
- Sports outcome forecasting
- Image classification
- Natural language processing

In traditional linear regression, the dependent variable y is assumed to be the linear sum of the independent variables $x_1, x_2, \dots, x_n$:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n.$$

The constants $\theta_0, \theta_1, \dots, \theta_n$ are called regression weights. They are calculated in such a way as to minimize the loss function, often formulated as the sum of the squares of the differences between the predicted and actual values of the dependent variable:

$$\sum_{i=1}^{n} (y_i - (\theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \cdots + \theta_n x_i^n))^2.$$

This loss function is called the normal function.

A solution to the problem of minimizing the normal function is offered by the least squares method. This method includes the following steps:

1. Calculate the matrix X, consisting of the observations of the independent variables.
2. Calculate the vector y, containing the actual values of the dependent variable.
3. Calculate the matrix W, the inverse of the matrix $X^T X$.
4. The regression weights are determined as follows:

$$\theta = W X^T y. \quad (1)$$

3 Random forest

Random Forest serves as an ensemble learning method designed for both classification and regression purposes. It includes many decision trees, each trained on separate subsets of the data set.

Leo Breiman introduced the random forest method in 2001, and since then it has become one of the widely used machine learning approaches, finding applications in various fields such as image recognition, natural language processing and prediction.

The learning algorithm for a random forest comprises the following steps:

1. Initialize a set of decision trees T.
2. For each decision tree:
   - Compute a random subset of size n_s from the original dataset.
   - Train the decision tree on the subset.
3. Make a prediction by voting the decision trees.

On step 2, the random forest algorithm uses bagging. Bagging is an ensemble learning method that trains a multitude of models on different subsets of the original dataset. This helps to reduce overfitting, which can occur when training a single model on the entire dataset.

On step 3, the random forest algorithm uses voting. Voting is a method that combines the predictions of multiple models to obtain a more accurate predictive result.

4 Parallel architecture

Parallel algorithms of machine learning methods were developed as part of the program. There are various parallel architectures. Modern cluster and supercomputer systems are built on a hybrid architecture, i.e. the system consists of computing nodes with shared memory, and each node can have a multicore CPU and GPU as an accelerator. For multicomputer systems and systems with a hybrid architecture, the MPI standard is usually used in parallel programs. MPI (Message Passing Interface) is a programming interface (API) for message passing that allows processes to exchange messages in a parallel computing system. MPI is the most widely used

standard for data exchange interfaces in parallel programming, and there are implementations for a large number of platforms. In this work, this standard was used for the parallel implementation of the methods under consideration.

There are two main approaches to implementing parallel algorithms: model parallelism and data parallelism. In the first approach, calculations are performed in parallel within the model. In the second approach, the data is divided into computing nodes, training takes place in parallel, and when forecasting, each node issues its own forecast, and the final decision can be made based on these forecasts by choosing the majority or calculating the average value.

A parallel linear regression algorithm has been developed. In this method, the weights can be found using gradient descent, and in some cases using the normal equation (1) by the analytical method. The equation consists of matrix multiplication, finding the inverse matrix and multiplying the matrix by a vector. These operations were parallelized using the MPI standard for shared memory systems. The Gauss-Jordan method was used to find the inverse matrix (W). The method itself is not well-parallelized, as the calculations have explicit dependencies. Therefore, a pipelined parallel algorithm was developed. To balance the computations, the matrix was distributed to the processes in a cyclic way. Matrix multiplication (X^TX and WX^T) and matrix-vector multiplication (result with y) were implemented with row-wise distribution of the matrix.

The most convenient method for parallelization on cluster systems is the Random Forest method. The method builds N decision trees on random subsamples. When forecasting, the average value from each tree is found. With a parallel implementation, N trees are divided into NP parallel processes and each process builds its decision trees independently of each other. As shown in Figure 1, independent decision trees are constructed. To do this, the dataset is sent to all processes using the MPI_Bcast functions, and the mean value, the solution to the problem, is combined using the MPI_Reduce functions.

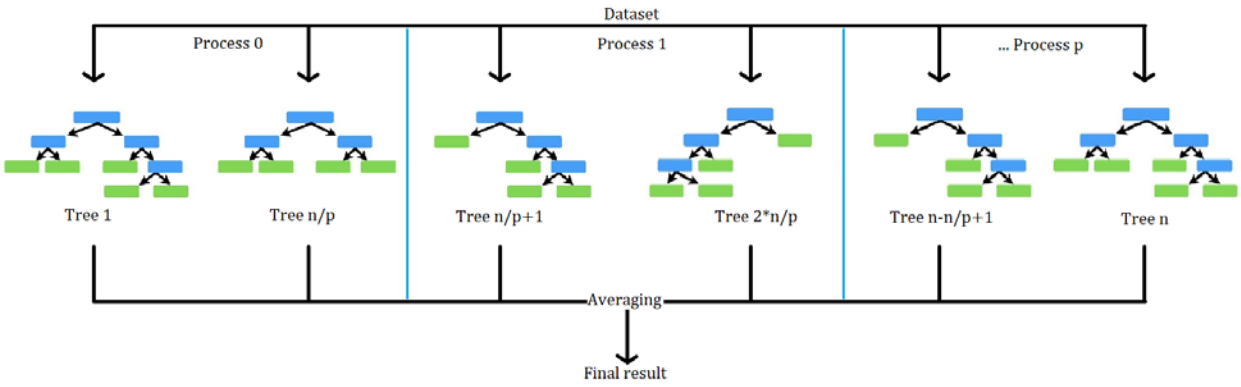

Figure 1 – Parallel Random Forest algorithm

5 Experimental results

Experiments were conducted on a dataset of the oil displacement problem. The characteristics of the dataset and the results obtained for sequential algorithms can be found in the work [17]. Tests of the developed parallel algorithms were conducted on a computer with an Intel® Core™ i7-10750H processor with 6 cores and 12 threads.

The results of parallelization of the Linear regression method can be seen in Table 1 and the results of parallelization of random forest can be seen in Table 2. The tests of the linear regression program were conducted on a dataset with a sample size of up to 2,400,000, and the tests of random forest up to 400,000. The random forest method, although more accurate, is computationally expensive. Therefore, the tests were conducted on a significantly smaller dataset size.

In all tables, the first column means the number of parallel processes, the first row means the sample size (the number of rows of the matrix X),

and each subsequent row is the elapsed time of the parallel program's execution. Based on the information presented in the tables, you can see that both algorithms gave acceleration relative to the sequential program. The acceleration of programs can be seen in Figure 2.

Table 1 – Execution time of the parallel LR algorithm, sec

NP\m	600000	1200000	2400000
1	4,22	7,52	14,93
2	1,61	4,05	8,44
4	1,29	3,18	6,18
8	1,24	2,83	5,74

Table 2 – Execution time of the parallel RF algorithm, sec

NP\m	100000	200000	400000
1	2,25	4,77	11,3
2	1,2	2,5	5,46
4	0,68	1,36	2,96
8	0,59	1,02	1,91

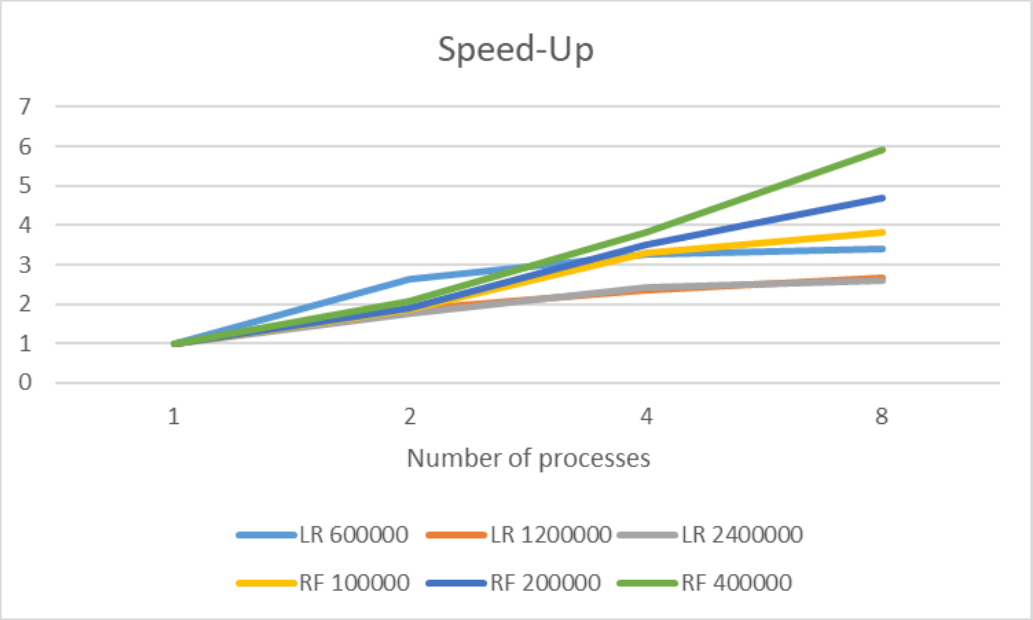

Figure 2 – Acceleration of LR and RF algorithms

From the graph, it can be seen that in linear regression, the acceleration decreases with increasing dataset size, while in random forest, the acceleration increases. Additionally, in random forest, a better acceleration was obtained in all cases. As mentioned earlier, random forest is the best machine learning method for parallelization. In linear regression, there is still a data dependency in the calculations, and the amount of communication is higher than in random forest. The increase in acceleration with increasing dataset size suggests that the parallel program could give even greater acceleration on even more data. This assumption requires experimental confirmation, which will be carried out in the future.

These experiments show that for a large dataset size on one processor, it is better to use the linear regression method, but if you have access to a system with multiple or more processors, in these cases it is better to use random forest.

6 Conclusion

In this work, methods of parallel data processing for machine learning were considered. Two of the most common machine learning methods were considered: linear regression and random forest. Parallel algorithms based on the MPI interface were developed for each method.

The experimental outcomes revealed that both methods give acceleration compared to the sequential algorithm. However, the acceleration in the case of random forest was significantly higher than in the case of linear regression. This is because random forest is a more computationally efficient method than linear regression. Hence, one can infer that random forest is the best machine learning method for parallel data processing. This conclusion is supported by the results of the experiments conducted in this work.

In addition, the experiments showed that the acceleration of linear regression decreases with increasing sample size, while the acceleration of random forest increases. This is because there is a data dependency in the calculations in linear regression, and the amount of communication is higher than in random forest.

Overall, the experimental findings indicate that employing parallel algorithms for machine learning can markedly expedite the training of models when dealing with extensive datasets. Random forest is the most efficient method for parallel data processing, as it is more computationally efficient and has higher scalability.

Acknowledgments

This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan, grant number BR18574136.

References

1. Murphy, Kevin P.. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013.
2. Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, 2009. https://doi.org/10.1007/978-0-387-84858-7.
3. Stock, James H., and Motohiro Yogo. “Testing for Weak Instruments in Linear IV Regression.” *Identification and Inference for Econometric Models*. Cambridge University Press, June 17, (2005). https://doi.org/10.1017/cbo9780511614491.006.
4. Weisberg, Sanford. “Applied Linear Regression.” *Wiley Series in Probability and Statistics*. Wiley, January 14, (2005). https://doi.org/10.1002/0471704091.
5. Tin Kam Ho. “Random Decision Forests.” *Proceedings of 3rd International Conference on Document Analysis and Recognition*. IEEE Comput. Soc. Press, n.d. https://doi.org/10.1109/icdar.1995.598994.
6. Breiman, L. “Random Forests.” *Machine Learning* 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
7. Jin, Haoqiang, Dennis Jespersen, Piyush Mehrotra, Rupak Biswas, Lei Huang, and Barbara Chapman. “High Performance Computing Using MPI and OpenMP on Multi-Core Parallel Systems.” *Parallel Computing*. Elsevier BV, September (2011). https://doi.org/10.1016/j.parco.2011.02.002.
8. Dalcin, Lisandro D., Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. “Parallel Distributed Computing Using Python.” *Advances in Water Resources*. Elsevier BV, September (2011). https://doi.org/10.1016/j.advwatres.2011.04.013.
9. Kamp, Michael, Mario Boley, Olana Missura, and Thomas Gärtner. “Effective Parallelisation for Machine Learning.” *arXiv*, (2018). https://doi.org/10.48550/ARXIV.1810.03530.
10. Shashank, Ojha and Kylee, Santos. Parallelizing Gradient Descent on Different Architectures. https://shashank-ojha.github.io/ParallelGradientDescent/ (accessed December 10, 2023)
11. Maleki, Saeed, Madanlal Musuvathi, and Todd Mytkowicz. “Parallel Stochastic Gradient Descent with Sound Combiners.” *arXiv*, (2017). https://doi.org/10.48550/ARXIV.1705.08030.

12. Mochurad, Lesia. “Optimization of Regression Analysis by Conducting Parallel Calculations.” *COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems*, April 22–23, 2021, Kharkiv, Ukraine

13. Lin, Chieh-Yen, Cheng-Hao Tsai, Ching-Pei Lee, and Chih-Jen Lin. “Large-Scale Logistic Regression and Linear Support Vector Machines Using Spark.” *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, October 2014. https://doi.org/10.1109/bigdata.2014.7004269.

14. Woodsend, K., and Gondzio, J. “Hybrid MPI/OpenMP parallel linear support vector machine training.” *Journal of Machine Learning Research*, 10, Article 1937-1953.

15. Mitchell, Lawrence, Terence M. Sloan, Muriel Mewissen, Peter Ghazal, Thorsten Forster, Michal Piotrowski, and Arthur S. Trew. “A Parallel Random Forest Classifier for R.” *Proceedings of the Second International Workshop on Emerging Computational Methods for the Life Sciences*. ACM, June 8, 2011. https://doi.org/10.1145/1996023.1996024.

16. Jie Hu. Parallelized Random Forests Learning, CSE 633 Course Project. https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Hu-Fall-2012-CSE633.pdf (accessed December 10, 2023)

17. Kenzhebek, Yerzhan, Timur Imankulov, Darkhan Akhmed-Zaki, and Beimbet Daribayev. “Implementation of Regression Algorithms for Oil Recovery Prediction.” *Eastern-European Journal of Enterprise Technologies*. Private Company Technology Center, April 30, 2022. https://doi.org/10.15587/1729-4061.2022.253886.