IRSTI 28.23.01 https://doi.org/10.26577/jpcsit2023v1i4a5

M.B. Mustafin[1*], Zh.S. Murataу[2], B.K. Orazov[3]

[1]U.A. Joldasbekov Institute of Mechanics and Engineering, Almaty, Kazakhstan
[2]Astana IT University, Astana, Kazakhstan
[3]Al-Farabi Kazakh National University, Almaty, Kazakhstan
*e-mail: mustafin.mb@gmail.com

HARNESSING GPU POWER FOR MACHINE LEARNING

Abstract. In the rapidly evolving landscape of data science and machine learning, the need for high-speed, efficient data processing is more critical than ever. This article delves into the pivotal role of graphics processing units (GPUs) in transforming the realm of data analytics and machine learning. GPUs, originally designed for rendering graphics in video games, have emerged as powerhouse tools in scientific computing due to their ability to perform parallel computations swiftly and effectively.

The work conducted a study and comparison of the performance of machine learning algorithms using the Scikit-Learn and RAPIDS cuML libraries on a GPU. Testing was carried out on various data volumes and the results confirmed significant execution speedup when using RAPIDS cuML. This highlights the practical importance of GPU acceleration for processing large data sets. In addition, the developed algorithms were successfully applied to predict the oil recovery factor based on the Buckley-Leverett mathematical model, demonstrating their effectiveness in the oil and gas industry. Overall, this article serves as a comprehensive overview of the current state and future prospects of GPU utilization in data processing and machine learning, providing valuable insights for both practitioners and researchers in the field.

Key words: GPU, Machine Learning, Linear Regression, Support Vector Machine Regression.

1 Introduction

In the modern world, the field of data and machine learning is becoming a key element in various areas, including science, industry, and technology. With the continuous increase in data volumes, there is a need for efficient methods and tools for processing and analyzing them. The use of graphics processing units (GPUs) has become an integral part of the data processing process, enabling parallel execution of complex computations.

The relevance of this topic is due to the rapid development of research in the field of optimization of machine learning algorithms using graphics processors. The authors of paper [1] describe modifications of the GPGPU-Sim simulator to support machine learning using PyTorch and cuDNN. This modification provided high accuracy of execution results and revealed new opportunities for optimizing the microarchitecture of GPUs for deep neural networks. In article [2], innovative methods are presented that accelerate feature generation and reduce model training time by 15.6 times. Using the RAPIDS.AI cuDF library and optimizations in PyTorch, the authors significantly reduced the prototyping time of deep learning-based recommendation systems. In work [3], a study is presented on the use of the cuML library, developed by NVIDIA to accelerate machine learning on GPUs, with particular attention to support for cluster systems with multiple GPUs. The authors propose a Python API for using MPI for communication in cuML, conduct analysis and benchmarking of algorithms.

This article presents an analysis of the effectiveness of various methods, including the use of CUDA technology for parallelizing matrix operations, the RAPIDS library set (cuML and cuDF) for data analysis and implementation of machine learning algorithms on GPUs [4, 5, 6].

This research also considers alternative machine learning methods, such as Linear Regression, SVM, implemented using the RAPIDS and Scikit Learn libraries [7, 8, 9]. The presented performance analysis highlights the advantages of using GPUs in training machine learning models, especially when processing extensive data volumes.

The developed algorithms were also tested on a dataset generated from the Buckley-Leverett mathematical model for predicting the oil recovery factor [10].

Based on the presented results, this article substantiates the importance and prospects of using graphics processors in the field of machine learning and highlights fundamental methods leading to improved performance and efficiency of algorithms.

2 Materials and Methods

Machine learning is a field of computer science that focuses on the development of algorithms and models that allow computers to learn from data and make predictions or make decisions without explicit programming. This field finds applications in various fields, including pattern recognition, data analysis, classification and forecasting.

Regression algorithms are used to solve the prediction problem when it is necessary to predict a continuous value based on input data. These algorithms build a model that describes the relationship between the input features and the output value. Examples of regression algorithms are linear regression, decision trees, support vector machine (SVM) and gradient boosting, each of which is suitable for different types of data and predictive tasks.

Scikit-learn, a widely used machine learning library for Python, provides powerful tools for applying regression algorithms. Among them is linear regression based on a model of linear dependence between the features and the target variable. Linear regression is often used in simple cases where a linear relationship between variables is assumed. Scikit-learn also provides an implementation of the Support vector Machine (SVM) for regression. In the case of SVR, the algorithm builds a hyperplane as far away from the data points as possible, and thus makes predictions. This method is effective when working with data, where it is difficult to identify linear patterns, and allows you to take into account nonlinear relationships.

Thus, Scikit-learn becomes an essential tool for machine learning specialists and researchers, providing user-friendly interfaces for learning, evaluating and selecting the best regression models depending on the specific requirements of forecasting tasks.

Scikit-Learn: LR and SVR algorithms

Linear regression in Scikit-learn is a model designed to describe a linear relationship between one or more features (independent variables) and a target variable (dependent variable) representing a continuous value.

Linear regression model

A linear regression model is a linear function that describes the relationship between input and output data. The visualization's example of linear regression model can be seen in Figure 1.

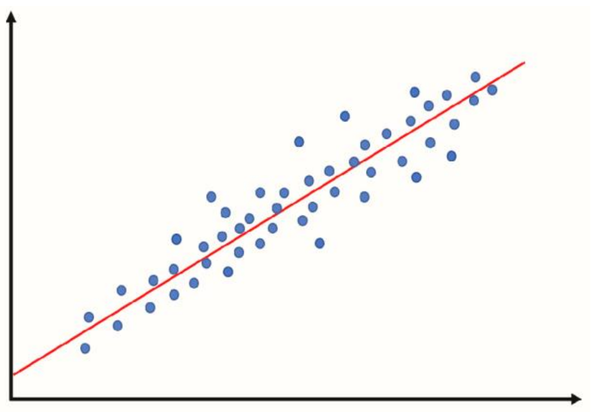

Figure 1 – Linear Regression (LR).

In general, the linear model looks like this:

$$y = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n,$$

where y – output value, w_0 – free term, w_i – weighting factor for the i-th feature, x_i – i-th attribute.

Weights in linear regression are used to account for the effect of each feature on the output feature. The weights of the linear regression model are determined during model training. The model learning algorithm minimizes the model error by selecting the optimal values of the weighting coefficients.

Support Vector Machine Regression (SVM) is a machine learning method that is used to predict continuous values based on a dataset with numerical features. SVR is an extension of the support vector machine, which is used for classification tasks.

The SVR model

The SVR model is a hyperplane that divides the feature space into two parts. The values of the output feature for points that are on one side of the hyperplane will be the same. The visualization's example of SVR model can be seen in Figure 2.

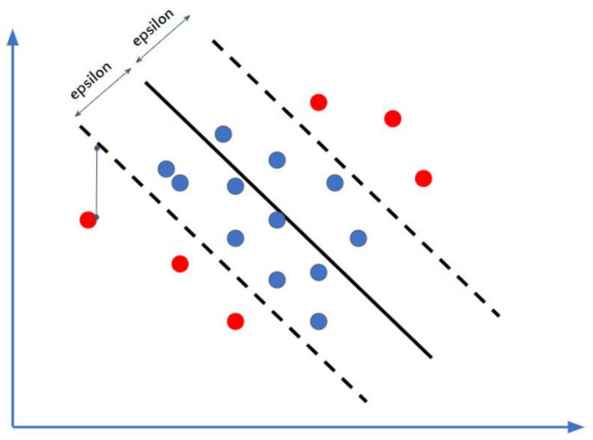

Figure 2 – Support vector regression (SVR).

The SVR model can be written as follows:

$$y = f(x) + \varepsilon,$$

where y – output value, x – input data, $f(x)$ – a function that describes the relationship between input and output data, ε – model error.

The function $f(x)$ can have different forms, depending on the method used to train the SVR model. The main goal here is to determine the boundary of the solution at a distance ε from the original hyperplane so that data points close to the hyperplane or being reference vectors are located within this boundary line.

RAPIDS: LR and SVR algorithms

RAPIDS is a set of open source libraries that accelerate machine learning and data analysis on the GPU. RAPIDS is based on the CUDA and cuda libraries and uses GPU capabilities to improve performance.

RAPIDS has the following libraries for use in machine learning:

- cuML – a library for machine learning on the GPU. It provides implementations of basic machine learning algorithms such as linear regression, logistic regression, k-means, k-nearest neighbors, etc.
- cuGraph – a library for processing graphs on the GPU. It provides tools for working with graphs, such as graph calculations, graph algorithms, etc.
- cuDF – a library for processing data on the GPU. It provides tools for working with data, such as data downloads, data transformations, calculations, etc.

RAPIDS cuML provides implementations of two main regression algorithms: linear regression (LR) and support vector machine regression (SVR).

Linear Regression is a function for training a linear regression model. This function takes as input a set of data with numeric attributes and a target attribute. The output result of the function is an object of the linear regression model

SVR is a function for training a regression model using the support vector machine. This function takes as input a set of data with numeric attributes and a target attribute. The output result of the function is an object of the regression model using the support vector machine.

The Linear Regression function and SVR use CUDA libraries to speed up GPU computing. CUDA is a set of libraries and tools that provide programmers with access to GPU hardware. CUDA libraries provide functions for performing various operations on the GPU, such as scalar operations, vector operations, and matrix operations. The linear regression model learning algorithm is also optimized for GPU computing. Special data structures are used that are optimized for accessing data from the GPU. In addition, special algorithms are used to speed up calculations.

predict is a function for predicting the values of a target feature for new data. This function takes as input an object of the linear regression model and a dataset with new data. The output result of the function is a set of predicted values of the target feature.

Implementation and Testing of Training Models using cuML

In this paper, two regression models based on the linear regression algorithm and the support vector machine (SVM) regression have been successfully implemented using the cuML library, specially designed for optimal performance on graphics processors (GPUs).

Implementations of LR and SVR using the cuML library in the RAPIDS environment on a graphics processor (GPU).

To begin with, random data was generated for the regression task using the make_regression function from the cuML library. The data was then converted to udf format for efficient processing on the GPU.

```
import cudf
from cuml.datasets import make_regression
from cuml.model_selection import train_test_split
n_samples = 80000
n_features = 20

X, y = make_regression(n_samples=n_samples, n_features=n_features, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train = cudf.DataFrame(X_train)
y_train = cudf.Series(y_train)
X_test = cudf.DataFrame(X_test)
y_test = cudf.Series(y_test)
```

Then, instances of the LR and SVM model were created using the Linear Regression and SVM class from the cuML library, and the model was trained on the training data.

```
from cuml import SVR
from cuml.linear_model import LinearRegression as LR
modelLR = LR( fit_intercept = True, normalize = True, algorithm = 'eig')
modelLR.fit(X_train, y_train)
modelSVR = SVR(C=1.0, kernel='rbf', gamma=0.1)
modelSVR.fit(X_train, y_train)
```

After training the model, its performance is evaluated. Predictions were made on the test data, and metrics were calculated for an objective assessment of the quality of the trained model. The developed algorithms have been successfully tested on a dataset based on the Buckley-Leverett mathematical model designed to predict the oil recovery coefficient. Input parameters include porosity, absolute permeability, phase viscosity, and time iterations.

3 Results and Discussion

The experiments were conducted on a personal computer with a high-performance Intel Core i9-10900KF processor, 32 gb of RAM and a discrete NVIDIA RTX 3070 GPU with 8 gb of video memory. This configuration provides powerful computing resources and allows efficient use of libraries optimized for working with the GPU. The results of Linear Regression testing using the scikit-learn and cuML libraries on the RAPIDS platform are presented in Table 1 and Table 2:

Table 1 – Linear Regression Test Results: scikit-learn vs cuML

Number of instances	Scikit-Learn	RAPIDS cuML
300000	0.10	0.004
600000	0.22	0.007
1200000	0.568	0.012
2000000	0.933	0.012

There is a significant acceleration of the Linear Regression model when using cuML on the RAPIDS platform compared to Scikit-Learn. The effectiveness of cuML is especially evident when the amount of data increases. It is noticeable that cuML on RAPIDS is successfully scaling with an increase in the number of data instances. This indicates the high efficiency of processing large amounts of data using a GPU. With the time difference, especially at 2,097,120 instances, cuML confirms its outstanding performance in processing large data. The graph of the results presented in Table 1 is shown in Figure 3.

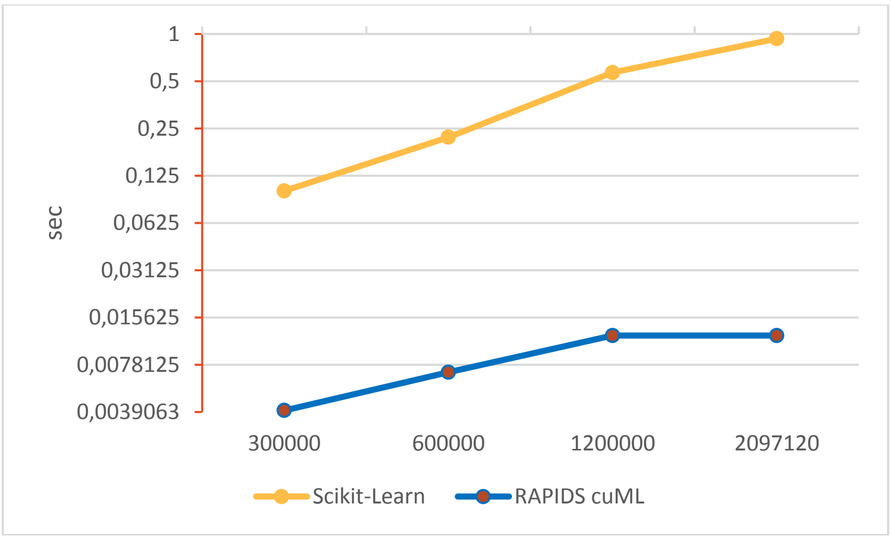

Figure 3 – The Effectiveness Of Linear Regression: Scikit-Learn vs RAPIDS cuML.

Table 2 – Test results of the Support Vector Machine (SVM): scikit-learn vs cuML

Number of instances	Scikit-Learn	RAPIDS cuML
10000	1.424	0.35
20000	5.795	0.62
40000	22.56	1.077
80000	83.9	3.562

The results clearly demonstrate a significant acceleration in the performance of the support vector machine (SVM) when using cuML on the RAPIDS platform compared to the scikit-learn library. The use of cuML is particularly advantageous when working with large amounts of data. With 80,000 instances, cuML demonstrates more than 20 times faster calculations compared to scikit-learn. The graph of the results presented in Table 2 is shown in Figure 4.

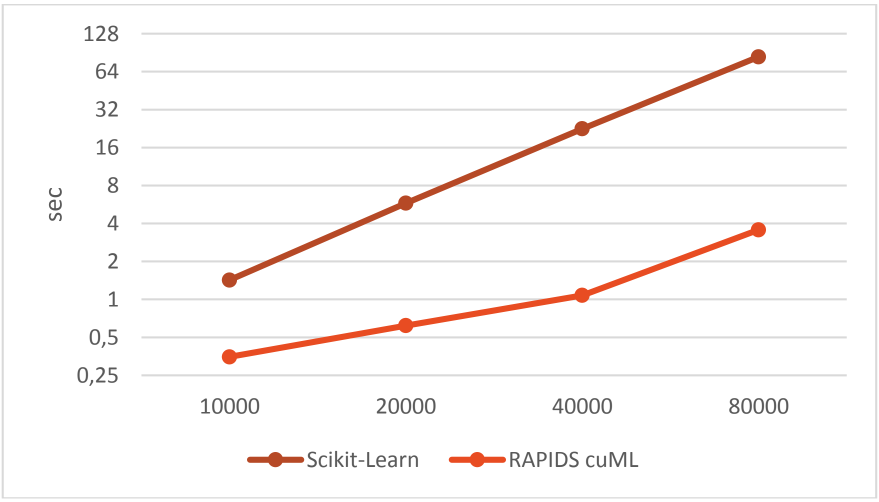

Figure 4 – The Effectiveness of the Support Vector Machine (SVM): Scikit-Learn vs RAPIDS cuML.

4 Conclusion

During the research, Linear Regression and Support Vector Machine (SVM) algorithms were developed and tested using the Scikit-Learn and cuML libraries on the RAPIDS platform. Experiments have been successfully conducted on data sets, including those generated from the Buckley-Leverett mathematical model for oil recovery factor prediction. Test results confirmed a significant acceleration of the algorithms when using cuML on the RAPIDS platform compared to the Scikit-Learn library. This is especially noticeable when processing large volumes of data, which demonstrates the outstanding performance of GPU computing.

Introducing GPU compute capabilities using RAPIDS and cuML opens new horizons in machine learning, delivering high performance and efficiency when processing big data. The results highlight the potential of these technologies to solve complex problems and provide a basis for further research and industrial applications.

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. J. Lew et al., "Analyzing Machine Learning Workloads Using a Detailed GPU Simulator," 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Madison, WI, USA, 2019, pp. 151-152, doi: 10.1109/ISPASS.2019.00028.

2. Sara Rabhi, Wenbo Sun, Julio Perez, Mads R. B. Kristensen, Jiwei Liu, and Even Oldridge. 2019. Accelerating recommender system training 15x with RAPIDS. In Proceedings of the Workshop on ACM Recommender Systems Challenge (RecSys Challenge '19). Association for Computing Machinery, New York, NY, USA, Article 8, 1–5. https://doi.org/10.1145/3359555.3359564

3. S. M. Ghazimirsaeed, Q. Anthony, A. Shafi, H. Subramoni and D. K. D. Panda, "Accelerating GPU-based Machine Learning in Python using MPI Library: A Case Study with MVAPICH2-GDR," 2020 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S), GA, USA, 2020, pp. 1-12, doi: 10.1109/MLHPCAI4S51975.2020.00010.

4. RAPIDS.AI. 2019. RAPIDS.AI cuDF repository. https://github.com/rapidsai/cuDF

5. T. Hricik, D. Bader and O. Green, "Using RAPIDS AI to Accelerate Graph Data Science Workflows," 2020 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 2020, pp. 1-4, doi: 10.1109/HPEC43674.2020.9286224.

6. N. Becker et al., "Streamlined and Accelerated Cyber Analyst Workflows with CLX and RAPIDS," 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 2011-2015, doi: 10.1109/BigData47090.2019.9006035.

7. G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller. 2015. Scikit-learn: Machine Learning Without Learning the Machinery. GetMobile: Mobile Comp. and Comm. 19, 1 (January 2015), 29–33. https://doi.org/10.1145/2786984.2786995.

8. Hao, J., & Ho, T. K. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. Journal of Educational and Behavioral Statistics, 44(3), 348-361. https://doi.org/10.3102/1076998619832248

9. Pedregosa, Fabian & Varoquaux, Gael & Gramfort, Alexandre & Michel, Vincent & Thirion, Bertrand & Grisel, Olivier & Blondel, Mathieu & Prettenhofer, Peter & Weiss, Ron & Dubourg, Vincent & Vanderplas, Jake & Passos, Alexandre & Cournapeau, David & Brucher, Matthieu & Perrot, Matthieu & Duchesnay, Edouard & Louppe, Gilles. (2012). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 12.

10.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