Logistic Regression via Different Computational Methods

Student: Elizaveta Lazareva

Introduction. The classification problem is widespread in machine learning world, while the logistic regression method is one of the most popular and simple methods for solving the classification problem. That is why, the study of the computational effectiveness of the training of logistic regression is of great value.

Goals of the project. The main goal of the project is to compare the training rates of logistic regression in different implementations: sklearn, numpy (using loops), numpy (using matrix computation), cupy, numba acceleration (for numpy implementation with loops), torch (which uses one GPU and with parallelization on 2 GPUs).

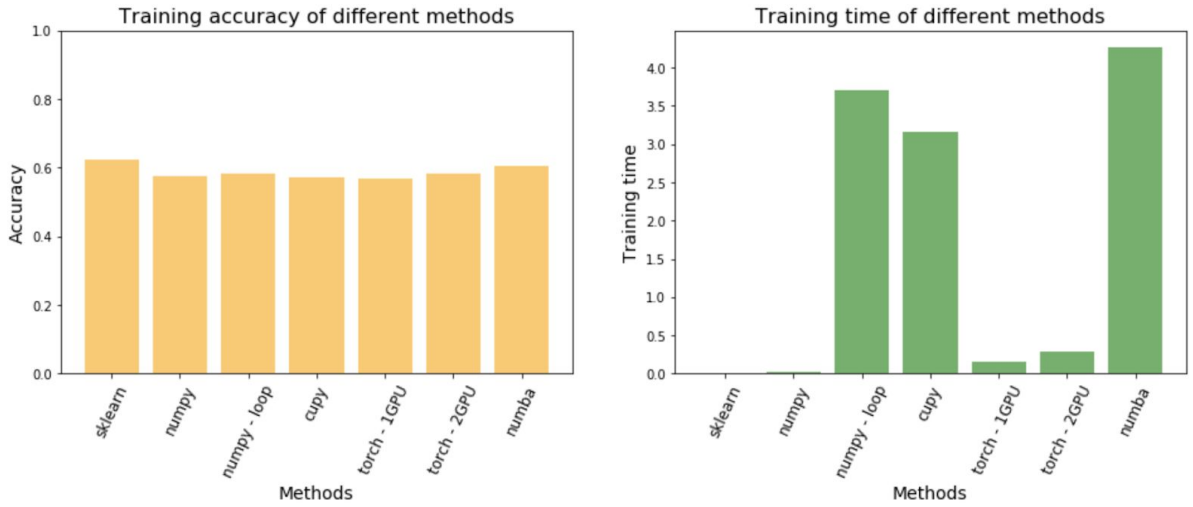
Data. For our experiments, we used 2 datasets:

- Heart Disease Dataset: the biomedical data of 303 patients, the target is diagnosis of heart disease, which is one of 5 classes.
- Titanic Dataset: the data of ? passengers of Titanic with binary target “survived”.

Both datasets are not big enough to see significant acceleration of GPU methods. That is why, the Titanic Dataset was extended by 10 and 100 times.

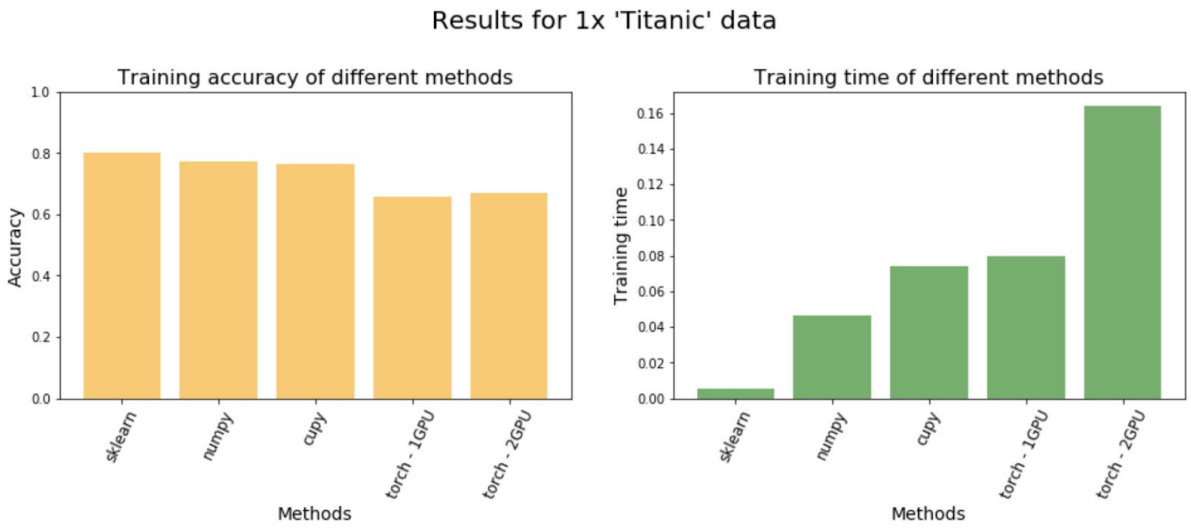
Results.

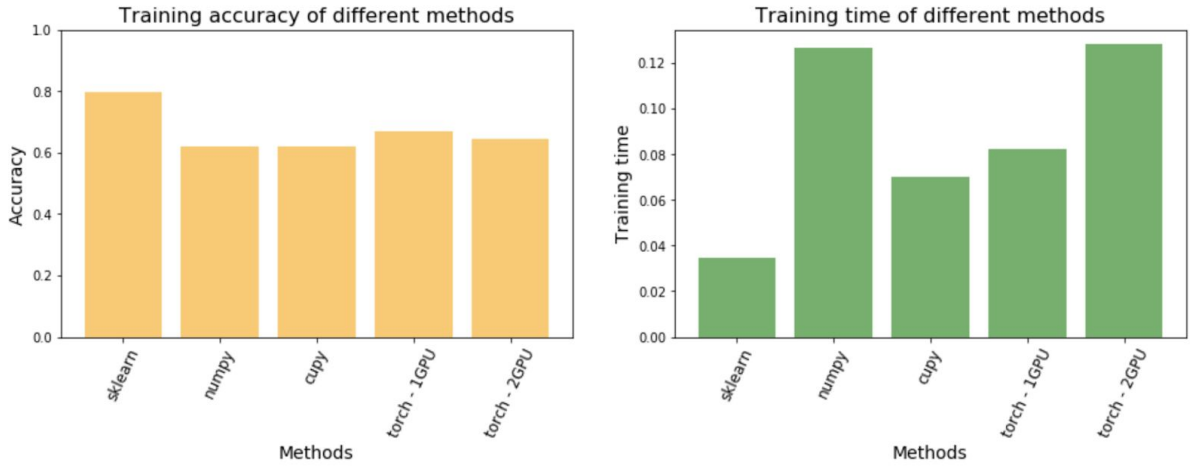
- Heart Disease Dataset

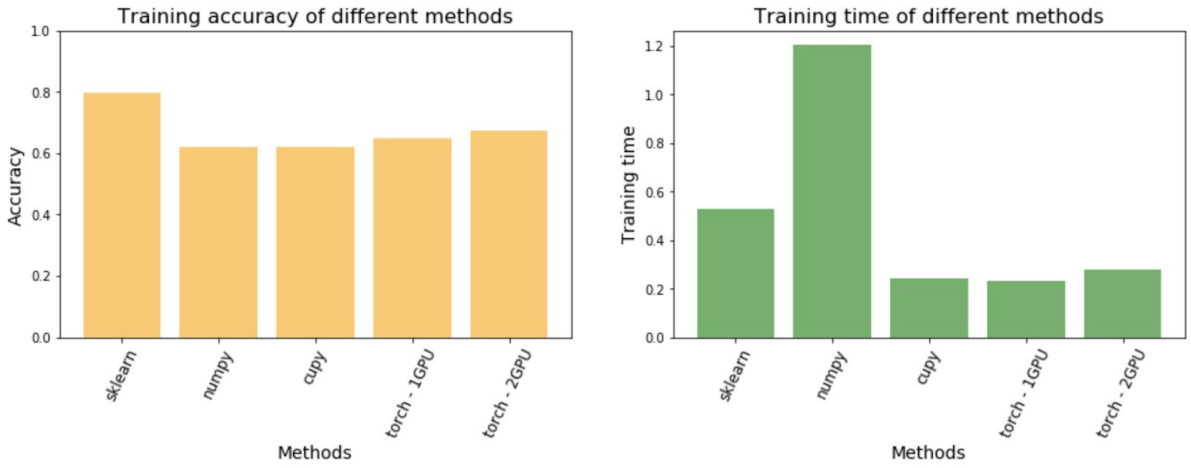


The fastest methods are Sklearn and NumPy implementation with NumPy matrix multiplication. Computation using PyTorch takes a little more time for transferring data to GPUs: we can see that time of training on 2 GPUs is twice bigger than for one GPU. CuPy implementation takes much more time to transfer the data to GPU than PyTorch. The code with NumPy (with loops) implementation using njit failed in some reason. The time of use jit is presented in the plot, which is bigger that just using a NumPy “with loops” implementation. It is because of the time spent to transform the code into machine code and attempts to compile it.

- Titanic Dataset







Here we exclude the results for Numba and NumPy (with loops) implementations because the training times for them is 100 times bigger than for other methods, which is very expected result. For the big number of data (100x 'Titanic' data) NumPy implementation is the slowest one. The CuPy and PyTorch implementations performs the same. We can see that there is not enough data to PyTorch parallelism become useful for this task: the cost for transferring data between different GPUs is not compensated by speed-up in computation.

Conclusion.

1. For small data (~1.000 rows x 10 features) it is more efficient to use Sklearn rather than other implementation because the cost of transferring the data to GPU is too big.
2. Starting from approximately 50.000 rows x 10 features data it is more efficient to use GPU computation, because time(transfering to GPU) + time(computation on GPU) < time(computation on CPU).
3. The transferring the small number data to GPU using CuPy, probably, takes more time than using GPU.
4. More data than 100.000 rows x 10 features is needed to make computation and transfering the data to 2 GPUs more efficient than computation on one GPU.

Link to the code. https://github.com/liza9797/Logistic-Regression