LUT University

LUT Computer Vision and Pattern Recognition Laboratory 2025-03-06

BM40A1401 GPU Computing

Lasse Lensu and Henri Petrow

Project

Machine Learning on GPU

1 Problem Statement

The task is to solve a *classification problem* using *machine learning (ML)*. When implementing the solution, the purpose is to benefit from parallel computing. Therefore, the implementations as *Jupyter notebooks* must use a *graphics processing unit (GPU)* for the steps where using one is appropriate.

The *implementations* must comply with the following requirements:

- The Jupyter notebooks must run on Google Colaboratory.
- In the experiments, the given data must be used according to the standard practices in the field.

The *data* will be published in Moodle and it can be characterised as follows:

- The data set consist of 4 000 observations and 7 numerical features.
- The observations represent 7 classes and the class labels are given as numbers in the last column of the file making the total number of columns 8.

2 Requirements

The project tasks include *programming* and *documentation*. The work is meant to be done in *groups of two students*.

To carry out the *programming task*, the following rules must be obeyed:

- Allowed:
  - The use of CUDA kernels with RawKernel function in CuPy.
  - The use of CuPy and PyTorch library functions.
  - The method implementations prepared by you or your group member for this course.
- Not allowed:
  - The use of "high-level" functions that perform the classification.
  - Any other source code from some source.
  - Using automated tools such as large language models for generating the code for the implementation.

To prepare the *documentation* of your work, the following rules must be obeyed:

- Allowed: You can use references for the documentation if 1) you acknowledge them (with a proper citation to the reference) and 2) you do not directly copy any text from a reference.
- Not allowed:
  - Use of any material prepared by others (without properly acknowledging the source).
  - Direct copying of sentences or their parts from a reference.
  - Using automated tools such as large language models for generating text for the documentation.

PO BOX 20, FI-53851 LAPPEENRANTA, FINLAND
E-mail: lasse.lensu@lut.fi

By returning the project results in the form of an implementation and document, you assure that:

- You acknowledge all sources.
- You have not used any forbidden material nor tools to create/generate them.

Any signs of plagiarism will be examined.

2.1 Programming

The classification task is solved using two different approaches for which background information will be given when introducing the project. The approaches to be implemented are as follows.

2.1.1 K-nearest-neighbour

K-nearest-neighbour (kNN) [2] is a supervised approach for classification. Even though it is one of the ML methods, it does not contain a learning step. This makes its operating principle very simple, but its performance has been shown to be very good in many applications and also theoretically [1].

One of the challenges of using the kNN classifier in practice is the curse of dimensionality, that is, how to implement it efficiently in the case of high-dimensional (large) data sets. In this project:

- Objective: realise a high-performance implementation of the method and show the advantage of using a GPU.
- Required:
  - Computing the distances between the observations by a CUDA kernel.
  - Implementing the other logic for the method in Python with functionality offered by the CuPy library.
  - Timing of method execution and comparing the result to a central processing unit (CPU) implementation.
- Optional: Optimising the model parameter.

2.1.2 Multilayer perceptron

A multilayer perceptron (MLP) is an artificial neural network (ANN) that consists of interconnected layers of perceptrons [3] that perform simple computations with a selected type of non-linearity. Despite the simplicity of the components, the model has been shown to be a universal approximator of any continuous function.

One of the challenges of using the MLPs is related to the computing resource requirements of training, especially when the model architecture is complex and contains a large number of parameters. However, when the architecture is kept small, one of the significant advantages is fast inference where the learned model is used in the feed-forward manner. In the project:

- Objective: realise as-simple-as possible MLP to solve the classification problem well enough and compare the performances of CPU and GPU implementations.
- Required:
  - Using PyTorch for the implementation on a CPU and GPU.
  - Using the `nn` module and `autograd` for implementing the model and the training loop.
  - Timing of the training and inference.
- Optional: Optimisation of the model and/or the training process.

It is necessary to properly comment your codes so that other people understand the purpose of each notebook cell.

2.2 Documentation

The implementation details and the project results are presented in a report written in English. In the beginning, the report contains the following information:

1. the course number and name,
2. project title,
3. date and
4. the student numbers, names and contribution/role of each group member.

In the report, it is not necessary to describe the theory behind the kNN method and the MLP model. However, the implementation details of the methods are to be described in such detail that a reader understands your choices and justifications behind them. If there are parameters involved, they must be presented clearly.

The provided data must be used according to the standard practices; how the data has been utilised is described in the report. The classification results and execution times are presented and complemented by an analysis what can be deduced from them.

At the end of your documentation, all the references are listed. Note that you are allowed to use any references/information you want, but all source code must be written by you or your group member for this project or for this course earlier.

3 Deadline and submission

The deadline of submitting the results of your work to Moodle is **Wednesday, 2 April 2025 at 12:00 EET**. The results consist of the document and the programmed implementations. The *requirements* are as follows (*GROUP* is the group number in Moodle):

1. The *document* is submitted in PDF format with filename *GROUP*`.pdf`.
2. The *implementations* are submitted as Jupyter notebooks with file names as *GROUP*`_knn.ipynb` and *GROUP*`_mlp.ipynb`. The notebooks must run on Google Colaboratory.

After the initial evaluation of the project results, a *review discussion* will be arranged for each group. In the review, *all group members must be present* and the teachers ask questions about the solution.

4 Evaluation

The submitted *report* and *implementations* will be evaluated according to the following criteria:

1. Design choices: The documentation contains enough details to understand the design choices and parameter values.
2. Proper implementations:
   (a) The implementations solve the problem and it is possible to execute them on Google Colaboratory.
   (b) The code is properly commented so that the grader can understand the implementations.
3. Results and analysis: The produced results are appropriately presented and analysed in depth to support the conclusions in the report.

4. Documentation: The content of the report is logical and relevant references are included.

In addition to the ones described above, the following criteria are taken into account based on the *implementation* and the *review discussion*:

- Optimisation of the method/model or the training process (if there is one).
- Novelty of the implementation. (If the idea comes from somebody else's work, you must acknowledge the source.)
- A well-designed demonstration or visualisation.
- Level of understanding of each group member about the solution and the methodology based on the review discussion. Even if the work has been divided, *each group member must understand and be able to explain the whole solution.*

To pass the course, the project result has to reach at least 50% of the maximum. The result affects the final grade of the course.

5 Notes and hints

Background information about the methods will be provided when the project is presented.

If there are any questions or problems with the assignment or the data, contact the person supervising the project before inventing your own interpretations about the instructions or making too radical assumptions.

References

[1] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 1967.

[2] Evelyn Fix and Joseph L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. Technical report, 1951.

[3] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.