MILESTONE 2: DISTRIBUTED ASYNCHRONOUS SGD

OVERVIEW

As specified before, for the second milestone, you have to extend the capabilities of your program to do the communication asynchronously. In this case, the workers do not block on sending/receiving updates to/from their peers, and they directly continue the computation after sending their updates. Remember your implementation has to support both synchronous and asynchronous distributed SGD with a maximally shared codebase. There should be an option in the configuration file that enables the user to specify the type of the communication, i.e., synchronous or asynchronous.

For the SGD algorithm, it should be as close as it gets to the HOGWILD! paper. However, please note that the HOGWILD! implementation runs on a single machine with a multicore processor, and its asynchronous manner relies on multiple threads communicating with each other using shared memory. In this project, your implementation runs on multiple machines, and communication occurs explicitly by sending/receiving messages, rather than implicitly using shared memory. Even though this approach has some similarities with the HOGWILD! paper in a sense that there is no locking and workers do not need to wait for each other before computing their values, the communication latency is higher when using the network as compared to shared memory.

For testing, you must run your implementation on the Reuters RCV1 dataset, on the binary text classification task CCAT. There are 804,414 examples split into 23,149 training and 781,265 test examples, and there are 47,236 features.

You must keep the same team as for milestone 1. You also need to submit a report describing what you did, how you implemented the algorithm, how you chose the learning rates and stopping criterion, along with a comparison between your choice and the choices made in the HOGWILD! paper. In your report, you should include a comparison between the execution time of the synchronous and asynchronous versions of your implementation, and justify your findings.

Similarly to the first milestone, there will be a session in which you will present and defend your work in front of the TAs, and you will be questioned about what you did, how you implemented the project, and about the contents of your report. We will inform you in time about how to sign up for a meeting with a TA to defend your submission, which will take place in the week of May 21st. Although we ask you to present and defend your work in front of TAs, we suggest you provide an easy way of running your project for reproducibility and our verification afterward.

DEPLOYMENT

For the deployment, we will use the IC cluster with Kubernetes. We will provide an extra document to give a short tutorial on Docker, Kubernetes, and how to set things up for the first time.

DEADLINES AND DELIVERABLES

The deadline for the milestone 2 is Monday, May 21, 23:55 pm Swiss time, and you have to upload a single zip file on moodle containing your implementation of SVM using distributed asynchronous/synchronous SGD along with your report. It is sufficient for one team member to upload the code. The zip file should contain a text file README.txt which lists the names of all team members.

We suggest you write the final report using the following structure:

- **Introduction**
  It serves as an outline of your entire project, with some highlights.
- **Design and Implementation**
  You should describe how you implemented your asynchronous SGD here, and how you changed your previous code base in milestone 1. You need to explain how you partition the computation and the dataset among workers, and how you manage the communication among them. Particularly, you should describe what is divided (partitioned), what is shared, and what is communicated among them. This part is very important, and we advise you to take time to think about your design and make your choices wisely. You should mention how and when you stop training the SVM. You should justify your design choices both in the report and in your presentation in front of the TAs.
- **Experiments**
  To demonstrate the correctness of the implementation, you need to include the following in your report:
  - Training and validation loss curves over the entire iterations.
  - Experiments conducted in various settings of hyper-parameters, such as learning rate, batch size, etc.
  - Comparison with synchronous SGD, regarding convergence time, hyper-parameter settings, etc. Is your asynchronous SGD faster than the synchronous one? You should analyze your program and explain why one implementation is faster than the other one.
  - Other necessary details to support your design.

Please follow the provided structure to the extent possible and explain things concisely. The report should not be long if you stick to this structure; having a long report does not win you any point.

For your submission, we suggest you submit the folder with the following structure:

```
Group<ID>-<SCIPER-1>-<SCIPER-2>-<SCIPER-3>
| --- report.pdf        # your eport
| --- README.txt        # lists of all team members
| --- run.sh            # with -async and -sync options
| --- src/              # source code of your SGD implementation
| --- Docker/           # docker files
| --- Kubernetes/       # kubernetes configuration files
```

You can also choose another style if it suits your development. However, you still need to make sure your file organization is easily interpretable, without any meaningless naming convention, (e.g., folder/file should not have names like xyz, abc, a.py, etc).