

Master project proposal: Speeding up distributed matrix completion with strategic work division and communication optimization

1 Introduction

Matrix completion [1] is one of the classical problems studied in machine learning. The input is a matrix A having m rows and n columns. Some of the entries in A are known, but the remaining entries are missing. The goal of matrix completion is to fill the missing entries in A with the help of mathematics and computation.

The best example of matrix completion in real life is perhaps the “Netflix problem”, where each known entry A_{ij} is the rating given by customer i of movie j . Certainly, there are many missing rating values in matrix A , because each customer has only rated some of the movies (impossible to watch all the movies). Matrix completion here means: try to deduce the missing rating values!

One of the commonly used strategies for matrix completion is to compute two new matrices W and H , see e.g. [2]. Matrix W has m rows and k columns, whereas matrix H has n rows and k columns. Normally, we have $k \ll \min(m, n)$, and k is also called the dimension of the latent space in machine learning. The goal is to have the result of matrix-matrix multiplication WH^T (where H^T denotes the transpose of H) to approximate A , thereby “filling in” the missing entries of A .

2 Stochastic gradient descent (SGD)

The SGD algorithm can be used to *iteratively* compute matrices W and H . Starting with some initial guesses of the entries in W and H , the SGD algorithm randomly goes through the known entries of A , where for every known entry $A_{i_t j_t}$ SGD updates the i_t -th row in W (denoted as $\mathbf{w}_{i_t}$) and the j_t -th row in H (denoted as $\mathbf{h}_{j_t}$) using the following formulas:

$$\mathbf{w}_{i_t} \leftarrow \mathbf{w}_{i_t} - s_t[(A_{i_t j_t} - \mathbf{w}_{i_t} \cdot \mathbf{h}_{j_t})\mathbf{h}_{j_t} + \lambda \mathbf{w}_{i_t}], \quad (1)$$

$$\mathbf{h}_{j_t} \leftarrow \mathbf{h}_{j_t} - s_t[(A_{i_t j_t} - \mathbf{w}_{i_t} \cdot \mathbf{h}_{j_t})\mathbf{w}_{i_t} + \lambda \mathbf{h}_{j_t}]. \quad (2)$$

In the above, t can be considered as an index of the iterations, s_t is the step size (which is decided beforehand), λ is a so-called regularization constant.

3 Distributed SGD

A parallelization of SGD was proposed in [2]. The basic idea is to let each computing unit (called a “*worker*”) be exclusively responsible for a subset of the rows in matrix W , making use of the corresponding rows of matrix A as data (i.e., the known entries on these rows). However, each row of matrix H , say $\mathbf{h}_j$, is a “shared” responsibility, because the known entries on column j in matrix A are typically divided among multiple workers.

The resulting software, called NOMAD [2], utilizes elaborate MPI communication to let each row of matrix H be “blindly passed around” among the workers, such that at any given time only one worker is allowed to update a particular row $\mathbf{h}_j$, whereas different workers can update different rows of H (as well as the workers’ own parts of matrix W) at the same time.

4 The new master project

One philosophy of NOMAD is to spread the responsibility of updating each $\mathbf{h}_j$ to many workers. This leads to the fact that the amount of computation a worker can do each time is rather small, because it is proportional to the number of known values on column j of matrix A that the worker is assigned with (decided when the rows of A and W are initially divided among the workers). Therefore, the ratio of communication overhead versus computational work can be high, leading to low parallel efficiency.

To improve the parallel efficiency, the new master project will use a strategic division of the rows of matrices A and W , such that the number of workers that are responsible for each $\mathbf{h}_j$ is reduced. This will consequently reduce the ratio of communication overhead versus computational work, thereby increasing the parallel efficiency. Moreover, we will make sure that $\mathbf{h}_j$ is always passed to one of the workers who can update it, instead of blindly passing $\mathbf{h}_j$ around. In the latter case, which is implemented in NOMAD, additional wasted communication overhead arises.

5 Proposed progression

The following suggestive plan assumes that the master student spans the research work over three semesters, where 33% of the first semester, 67% of the second semester and 100% of the third semester can be devoted to research: Semester 1: Literature reading and testing with the original NOMAD code.

Semester 2: Implementation of strategic division of the rows of matrices A and W , study of the impact of this new work division with respect to the average number of workers that can contribute to updating each $\mathbf{h}_j$; At the same time, first-round modification of NOMAD to avoid passing $\mathbf{h}_j$ to workers who cannot contribute. Second-round modification of NOMAD by combining strategic division of A and W with the modified NOMAD code from Month 2.

Semester 3: Detailed performance profiling and analysis of the resulting new code of distributed SGD, with the purpose of understanding the pros and cons of the code modifications implemented. Experiments with large-scale data sets (e.g., Netflix data) and detailed comparisons of the new distributed SGD against NOMAD; Start drafting the thesis. Thesis writing.

References

- [1] WIKIPEDIA. Matrix completion.
https://en.wikipedia.org/wiki/Matrix_completion.
- [2] Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, S. V. N. Vishwanathan, and Inderjit Dhillon. NOMAD: Non-locking, stOchastic Multi-machine algorithm for Asynchronous and Decentralized matrix completion. In *Proceedings of the VLDB Endowment*, volume 7, pages 975–986, 2014.