Asynchronous Distributed Matrix Factorization with Similar User and Item Based Regularization

Bikash Joshi† Franck Iutzeler‡ Massih-Reza Amini†

University of Grenoble Alps - CNRS
† Computer Science Laboratory (LIG)
‡ Applied Mathematics Laboratory (LJK)
{FirstName.LastName}@imag.fr

ABSTRACT

We introduce an asynchronous distributed stochastic gradient algorithm for matrix factorization based collaborative filtering. The main idea of this approach is to distribute the user-rating matrix across different machines, each having access only to a part of the information, and to asynchronously propagate the updates of the stochastic gradient optimization across the network. Each time a machine receives a parameter vector, it averages its current parameter vector with the received one, and continues its iterations from this new point. Additionally, we introduce a similarity based regularization that constrains the user and item factors to be close to the average factors of their similar users and items found on subparts of the distributed user-rating matrix. We analyze the impact of the regularization terms on MovieLens (100K, 1M, 10M) and NetFlix datasets and show that it leads to a more efficient matrix factorization in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), and that the asynchronous distributed approach significantly improves in convergence time as compared to an equivalent synchronous distributed approach.

Keywords

Recommender Systems, Asynchronous Distributed Optimization, Similarity-based Regularization

1. INTRODUCTION

The immense growth of e-commerce in the last decade has resulted in an increasing popularity for Recommender Systems, helping customers to identify products that best fit their personal tastes. Many Recommender Systems approaches were proposed in the recent years among which collaborative filtering based algorithms have been found to be very effective [6]. The underlying hypothesis $\mathcal{H}$ of these algorithms is that if two users share the same opinion over a set of items, they are likely to share the same opinion over new items. Two popular collaborative filtering techniques that have been developed based on this idea are 1) Neighborhood methods [5], which generate prediction for unseen items using the past ratings of similar users or items; and 2) Matrix Factorization techniques [3], that approximate the matrix of ratings with the product of two matrices corresponding to users and items in some latent space.

In this study, we tackle this problem by splitting the rating matrix across different machines of a network and propose a novel distributed asynchronous framework for matrix factorization using the Stochastic Gradient Optimization (`SGO`) method for recommender systems. The rational is that machines connected in a network are generally loaded differently and synchronous updates may become very slow as every machine has to wait for the slowest machine to finish before updating. Moreover, in order to enhance the underlying hypothesis $\mathcal{H}$ stated above, we add to the classical RMSE objective two similarity-based regularization terms that constrain respectively user and item factors to be close to the factors of their most similar users and items found in the subpart of the rating matrix stored in the machine. In our asynchronous framework, each machine performs `SGO` in parallel on different parts of the rating matrix and sends its updated weights to the other machines in the network. Once a machine receives an updated weight, it averages the parameter weights with its own updates and continues applying `SGO` from the new averaged point. This guarantees that each machine keeps an overall view over the whole data.

Using Movielens and Netflix datasets, we show that the addition of the proposed user and item regularization terms lead to a more effective matrix factorization in terms of RMSE and MAE. We also demonstrate the attractive convergence properties of our asynchronous distributed update scheme compared to an equivalent synchronous distributed approach [8, 4].

In the remainder of the paper, we define in Section 2 the Matrix Factorization problem with our proposed similarity-based regularization. Then, in Section 3, we describe the proposed distributed asynchronous framework. Finally, we discuss the experimental settings and results in Section 4.

2. MATRIX FACTORIZATION WITH USER AND ITEM BASED REGULARIZATION

In this section, we introduce basic definitions and the objectives that we address in our setting of matrix factorization. Users and items are respectively represented by the sets $\mathcal{U}$ and $\mathcal{I}$, where u_i and i_j are respectively the i^{th} user and the j^{th} item. Furthermore, we denote by r_{ij} the rat-

RecSys '16, September 15-19, 2016, Boston , MA, USA

DOI: http://dx.doi.org/10.1145/2959100.2959161

ing of item i_j by user u_i and we define the ratings matrix R as the matrix of the r_{ij} whenever they exist. The user $|\mathcal{U}| \times K$ factor matrix (resp. item $|\mathcal{I}| \times K$) in a latent space of dimension K is denoted by P (resp. Q), and p_i (resp. q_j) is the size-k vector corresponding to the i^{th} user/line (resp. j^{th} item/line).

Matrix factorization with Stochastic Gradient Optimization (SGO) has been proved to be a successful approach for recommender systems, notably after winning the Netflix prize [3]. The premise behind this approach is to approximate the large rating matrix R with the multiplication of low-dimensional factor matrices PQ^{T}. These factor matrices try to model user preference for items with small number K of implicit factors. For a pair of user and item (u_i, i_j) for which a rating r_{ij} exists, this approach is based on the minimization of the ℓ_2-regularized quadratic error:

$$\ell(u_i, i_j, P, Q) = \left(r_{ij} - q_j^\top p_i\right)^2 + \lambda(||p_i||^2 + ||q_j||^2) \quad (1)$$

where $\lambda \geq 0$ is the regularization parameter. The whole matrix factorization problem thus writes

$$\min_{P,Q} \sum_{i,j:r_{ij}\text{exists}} \ell(u_i, i_j, P, Q) \quad (2)$$

Note that the error $\ell(u_i, i_j, P, Q)$ depends only on P and Q through p_i and q_j; however, item i_j may also be rated by user $u_{i'}$ so that the optimal factor q_j depends on both p_i and $p_{i'}$. The Stochastic Gradient Optimization (SGO) method thus proceeds as follows: at each iteration t, i) select a random user/item pair (u_{i^t}, i_{j^t}) for which a rating exists; ii) perform a gradient step on $\ell(u_{i^t}, i_{j^t}, P, Q)$.

In order to enhance the assumption that similar users have similar tastes, we impose that the factor vector of each user (resp. item) should be close to the average factor vector of its similar users (resp. items). For computing the most similar users (or items) we considered a modified version of Pearson correlation coefficient [2] which for two users u_i and u_j writes:

$$sim(u_i, u_j) = \frac{\sum_{i_k \in I_c}(r_{ik} - \overline{r}_{i.})(r_{jk} - \overline{r}_{j.})}{\sqrt{\sum_{i_k \in I_c}(r_{ik} - \overline{r}_{i.})^2}\sqrt{\sum_{i_k \in I_c}(r_{jk} - \overline{r}_{j.})^2}}$$

Where, I_c is the items co-rated by both users, $\overline{r}_{i.}$ and $\overline{r}_{j.}$ denote the average ratings for u_i and u_j respectively.

Hence, we are able to find the N most similar users for u_i, denoted by N_i (resp. N_j for items similar to i_j). We now propose a slight modification of the individual ratings objective function ℓ of Eq. (1) above by adding another regularization term. For a pair of user and item (u_i, i_j) for which a rating r_{ij} exists, the similarity-regularized individual objective writes:

$$\ell_1(u_i, i_j, P, Q) = (r_{ij} - q_j^\top p_i)^2 + \lambda(||p_i||^2 + ||q_j||^2)$$
$$+\lambda_u \left\| p_i - \frac{1}{|N_i|}\sum_{m \in N_i} p_m \right\|^2 + \lambda_i \left\| q_j - \frac{1}{|N_j|}\sum_{n \in N_j} q_n \right\|^2 \quad (3)$$

where $\lambda_u \geq 0$ and $\lambda_i \geq 0$ are the regularization parameters linked to the similar-user and similar-item regularizations respectively. Performing the same updates as the conventional SGO but replacing ℓ by ℓ_1, we get Algorithm 1 for minimizing the whole matrix factorization problem (Eq. 2) where ℓ is replaced by ℓ_1 i.e. the similarity-regularized problem:

$$\min_{P,Q} \sum_{i,j:r_{ij}\text{exists}} \ell_1(u_i, i_j, P, Q). \quad (4)$$

Algorithm 1 Similar based regularization

procedure MODIFIED SGO
  Input: $R, \lambda, \lambda_u, \lambda_i$
  Initialize: P and Q randomly
  while not converged **do**
    Choose randomly $(u_i, i_j) \in R$
    $N_i = GetSimilarUsers(i, N)$
    $N_j = GetSimilarItems(j, N)$
    Update p_i and q_j by a gradient step on $\ell_1(u_i, i_j, \cdot, \cdot)$ (Eq. 3)

3. ASYNCHRONOUS DISTRIBUTED SGO

Even though stochastic gradient optimization offers a high prediction accuracy on recommender system datasets, there are some computational challenges associated with it. Performing SGO sequentially on a single machine takes unacceptably large amount of time to converge. So, there is a need to perform SGO in a distributed manner for large datasets. However, parallelizing SGO is not trivial. A drawback of a straightforward implementation is that updates on factor matrices might not be independent. For example, for the training points that lie on same rows (or columns), they update the same corresponding row (or column) of P and Q matrices simultaneously. So, efficient communication between the computing nodes is necessary to synchronize the updates on factor matrices.

One common approach is to divide the rating matrix into several blocks and run SGO on each of the blocks on distinct machines. Factor matrices are updated on each machine for the corresponding ratings. Even though the rating matrix parts on each machine are different, the factor matrix updates are not independent. So, after each epoch the factor matrices present in each machine are synchronized. We refer to this method as "Synchronized SGO", as all machines synchronize their factor matrices after every epoch (in [4] this method is referred to as Asynchronous SGD because the parameters are communicated asynchronously, but however, the epochs are synchronized as aforementioned). But, such approaches might not be efficient mainly because the computation time for one epoch of SGO is different depending on the machine it is performed. So, this approach of synchronous propagation between machines after each epoch can considerably be slowed down because the faster machines have to wait for the slower ones in order to synchronize.

We propose here a distributed algorithm for SGO with asynchronous propagation of parameter matrices after each epoch. Each of the machines contain a part of the rating matrix corresponding to clusters of similar users, that could be found efficiently offline with large-scale graph clustering techniques [7]. Factor matrices are hence updated on each machine using only the part of the rating matrix it contains. As opposed to the synchronous method, we broadcast the updated factor matrices as soon as the machine finishes one epoch. So, whenever a new epoch begins, it collects the most recent factor matrices from other machines and after finishing one epoch it broadcasts its updated factor matrices to

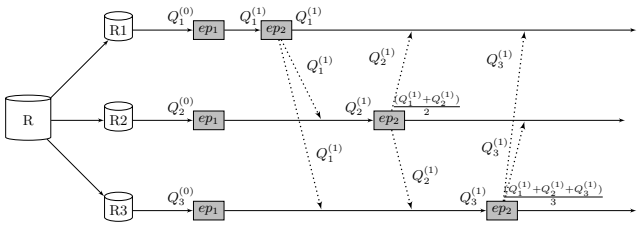
Table 1: MAE and RMSE measures for different methods on MovieLens and NetFlix datasets. Best results are shown in bold.

Dataset		SGO		Modified SGO With Similarity Based Regularization			
				Similar Users and Items, $\lambda_u = \lambda_i$		Similar Users Only, $\lambda_i = 0$	
		MAE	RMSE	MAE	RMSE	MAE	RMSE
ML-100K	ua	0.7490	0.9478	**0.7390**	**0.9332**	0.7404	0.9359
	ub	0.7619	0.9660	**0.7555**	**0.9564**	0.7540	0.9590
ML-1M	ra	0.7324	0.9706	0.7208	0.9517	**0.7188**	**0.9487**
	rb	0.6973	0.8861	**0.6928**	**0.8787**	0.6946	0.8799
ML-10M	rb	0.6523	0.8415	**0.6488**	**0.8384**	0.6512	0.8402
NF-Subset	NA	0.6498	0.8287	**0.6469**	**0.8256**	0.6477	0.8267

all other machines and begins another epoch immediately, independently of the state of other machines. In this way, all the machines are running independently of each other. So, the faster machines will perform their epochs faster, whereas the slower slower machines will be lagging on time but after finishing each epoch they will receive the most updated factor matrices from the faster machines.

In our setting, we cluster the rating matrix R into m blocks of lines as in [4]. Where m is the number of machines (or processes) available. The P matrix is also blocked in the same way. At each node we keep: one block of R, the corresponding block of P, and a local copy of Q. At each epoch, we update that block of P and local Q matrix based on the block of R and the individual loss functions (ℓ or ℓ_1). The overall procedure is depicted in Figure 1, showing an example distributed network of three machines. R_1, R_2 and R_3 represent the rating blocks on each machine. *ep* represents the end of previous epoch and beginning of new epoch on different machines. As shown, after finishing one epoch each machine broadcasts its updated Q matrix to all other machines, whereas at the beginning of new epoch each machine averages its Q update with the received updated Q matrices from other machines (if any). In this way the updates from faster machines help the slower machines to converge faster, thus enhancing the overall convergence time of the distributed network.

Figure 1: Block Diagram of Asynchronous Distributed SGO



4. EXPERIMENTS

We conducted a number of experiments aimed at evaluating what are the impacts of similar user and item regularization terms on the matrix factorization, and how the proposed asynchronous framework can help to speed up the convergence of the SGO algorithm.

Datasets: We performed experiments on MovieLens datasets (ML-100K, ML-1M and ML-10M) and a subset of the NetFlix collection (see Table 2). For ML-100K and ML-1M we used both sets (ra and rb, or ua and ub), whereas for ML-10M and Netflix we used a single dataset.

Table 2: Characteristics of Datasets used in our experiments. $|\mathcal{U}|$ and $|\mathcal{I}|$ denote respectively the number of users and items.

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	K	training size	test size	sparsity
ML-100K	943	1682	20	90570	9430	93.7 %
ML-1M	6040	3952	40	939809	60400	95.8 %
ML-10M	71567	10681	100	9301274	698780	98.7 %
NF-Subset	28978	1821	40	3255352	100478	93.7 %

Implementation: We implemented the distributed framework using PySpark version 1.5.1. To demonstrate the communication between disparate machines, we connected five servers with different computational loads.

Platform and Parameters: Three of the machines have Intel Xenon E5-2640 2.60 GHz processors and 256 GB memory each. The other two machines have Intel Xenon E5-2643 3.40 GHz processors and 128 GB memory each. Even though some of the machines have identical configuration, they were running different workloads on them. One of the machines dispatches the rating matrix across all the others as well as on itself. We stop the running algorithm on each machine, when the RMSE difference between two epochs on a validation set falls below a predefined threshold, ε, fixed as 10^{-4} in our experiments. The convergence of the proposed asynchronous approach is set when the slowest machine finishes its job, while for the synchronous SGO all machines finish at the same time.

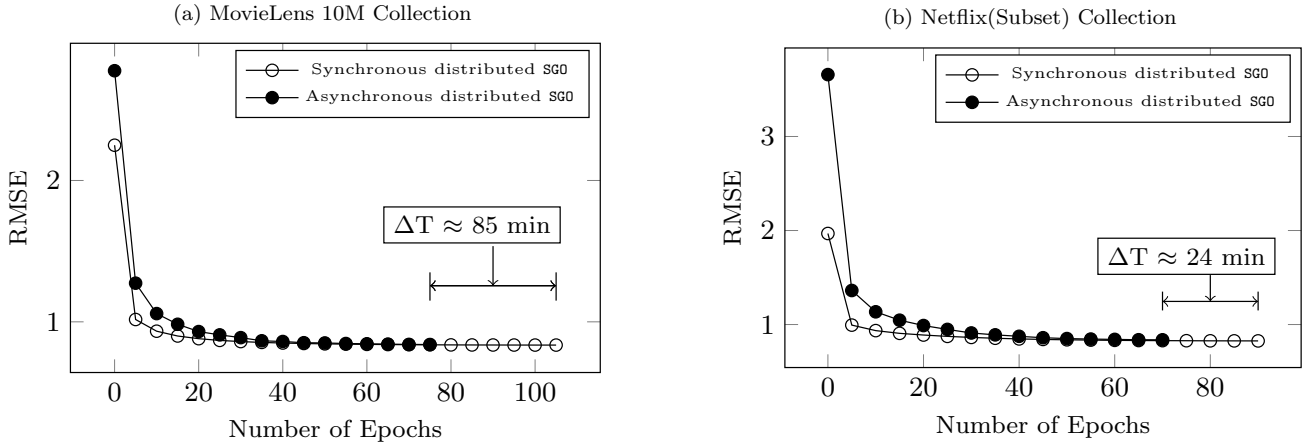
Following [1], the number of latent factors for dataset ML-10M was fixed to 100, and for the other smaller datasets, it was chosen experimentally for best result/speed compromise as 40, 40 and 20 respectively. The learning rate and the regularization parameter λ were fixed to 0.005 and 0.05 as in [1]. For our proposed similarity-based regularization λ_u, λ_i, and the number of similar users/items N were chosen with values that led to the best RMSE on validation sets for each collection chosen among $\{10^{-1}, 5.10^{-2}, 10^{-2}, 5.10^{-3}, 10^{-3}, 5.10^{-4}, 10^{-4}\}$ for λ_u, λ_i and $\{10, 20, 30, 40, 50\}$ for N.

Evaluation Measures: In our experiments, we used the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) as performance measures. Also, we compared the convergence time for each of the algorithms in the synchronous and the asynchronous distributed settings.

4.1 The effect of similar based regularization

First, we compare the results between the traditional SGO method and the proposed Modified SGO with Similarity Based Regularization. The difference here is solely on the objective function that is minimized (Pbs. (2) and (4) respectively).

Figure 2: RMSE vs Number of Epochs, until convergence of synchronous and asynchronous distributed SGO approaches with similar user and item based regularization for matrix factorization.



We tested two scenarios: i) where only the users are regularized with similarity ($\lambda_i = 0$); and ii) when both users and items are regularized with the same parameter ($\lambda_u = \lambda_i$). Table 1 shows the complete results of our experiments.

It comes out that forcing the vectors of users and items to lie within the centroids of their most similar users and items found by the Pearson similarity measure is effective as the final RMSE and MAE with Algorithm 1 are always better than with classical SGO. Thus, there is a significant benefit to use this regularization in terms of learning. We also report results by looking at the effect of the similar user regularization and not items ($\lambda_u > 0, \lambda_i = 0$). As shown in Table 1, this user-only regularization also gives uniformly better results than traditional SGO, and even better than the user and item regularization on one dataset.

4.2 Evaluation of Convergence time

Now, we evaluate the proposed asynchronous distributed method by comparing its convergence time with the synchronous distributed SGO method. We evaluate the convergence speed on the problem with user and item similarity regularizations in order to show its scalability. The evolutions of RMSE with respect to the number of epochs of our proposed algorithm and the synchronous SGO on the ML-10M dataset and the subset of the Netflix collection are shown in Figure 2. In the case of the asynchronous approach, we show the convergence time of the slowest machine. As expected, the proposed approach converges in less epochs as the slowest machine gets updated values from the faster machines, which helps it to converge faster, whereas in the case of the synchronous method, the faster machines have to wait for the slower ones after every epoch.

5. CONCLUSION

In this work, we presented an asynchronous distributed framework for SGO. Though generic, the framework was applied to matrix factorization for recommender systems. Furthermore, we proposed two additional regularization terms for this task. Experimental results on different MovieLens and NetFlix collections tend to validate the interest of this similarity-based regularization as well as the convergence speedup of the proposed asynchronous approach.

6. ACKNOWLEDGEMENT

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.

7. REFERENCES

[1] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A learning-rate schedule for stochastic gradient methods to matrix factorization. In *Advances in Knowledge Discovery and Data Mining*, pages 442–455. Springer, 2015.

[2] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[3] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[4] F. Makari, C. Teflioudi, R. Gemulla, P. Haas, and Y. Sismanis. Shared-memory and shared-nothing stochastic gradient descent algorithms for matrix completion. *Knowledge and Information Systems*, 42(3):493–523, 2015.

[5] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *National Conference on Artificial Intelligence*, pages 187–192, 2002.

[6] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *International conference on World Wide Web*, pages 285–295. ACM, 2001.

[7] J. J. Whang, X. Sui, and I. S. Dhillon. Scalable and memory-efficient clustering of large-scale social networks. In *International Conference on Data Mining*, pages 705–714, 2012.

[8] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.