SAG-re: Faster Prototyping of Recommender Systems using Stochastic Average Gradient

James Lo
Computer Science
University of British Columbia
Vancouver, Canada
tklo@cs.ubc.ca

ABSTRACT

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

General Terms

Asymptotic Time Complexity, Asymptotic Space Complexity, Experimentation

Keywords

Recommender systems, collaborative filtering, matrix factorization, stochastic gradient

1. INTRODUCTION

Shopping, text advertising, display advertising, renting movies, listening to music... recommender systems are prevalent and ubiquitous in our daily lives. Matrix factorization (MF) is a popular technique in model-based recommender systems. MF has been utilized extensively in past research for handling both explicit [4, 9, 7] ratings, and implicit [1, 2, 8, 3, 9] feedback.

In recommender systems that utilize matrix factorization, most optimize an objective function. Full deterministic gradient (FG) and stochastic gradient (SG) are the two main gradient methods for optimization. All of the recommender systems that we cite above utilize either full deterministic gradient, or stochastic gradient.

Unfortunately, both full deterministic gradient and stochastic gradient have pitfalls when it comes to prototyping recommender systems. Full deterministic gradient can offer high quality optimizations. However, FG is slow because at each iteration of optimization, FG has to pass through all the samples in the dataset. Stochastic gradient is relatively fast; its iteration cost is low because each iteration of SG looks at only one or a few samples. However, the trade-off with stochastic gradient is that it often provides low quality optimizations. By chance, stochastic gradient may *eventually* yield a good quality optimization. If it ever happens, it is after a tremendous number of iterations. Thus stochastic gradient is also slow in terms of yielding a good quality optimization within a reasonable amount of time.

High quality optimizations within a short amount of time is important when building recommender systems. The first reason is that data scientists often have to run repeated experiments: e.g. with different objective functions, different metrics, different datasets, and different optimization parameters. The high level goal to run multiple experiments is that, through experimentation and comparing results of multiple trials, data scientists can ultimately get a sufficiently good mix of objective function and hyper parameters for fitting a dataset. The second reason is that product life cycles are shortening in the age of agile software engineering. Thus data scientists are faced with the challenge of running more experiments and producing high quality results with less time.

In this paper, we study the challenge from the perspective of convex-optimization. We propose and hypothesize using the stochastic average gradient (SAG) method [6, 5] as a viable alternative to using FG and SG during the prototyping process. SAG has the distinctive advantage that its optimization quality is proven to be much better than SG; at the same time SAG's iteration cost is asymptotically as low as SG. However, applying and adapting SAG to matrix factorization is not trivial because SAG requires previously-computed gradients; and storing these gradients can lead to very high asymptotic space complexity. We explore the challenge with space-complexity, and resolve it by proposing a re-computation approach (SAG-RE) that re-computes the previously-computed gradients on-the-fly, on-demand. SAG-RE preserves the fast convergence rate and low iteration cost of SAG. Moreover, the asymptotic space complexity of SAG-RE is as compact as memory-less gradient methods such as FG and SG.

To the best of our knowledge, we are the first to

- Identify pitfalls associated with using full deterministic gradient and stochastic gradient when data-scientists prototype model-based recommender systems.
- Propose Stochastic Average Gradient (SAG) as a viable alternative for yielding higher quality optimizations while enjoying a low iteration cost.
- Extend SAG into SAG-RE for matrix factorization, resolve the space complexity challenge in adapting SAG from the domain of large-scale supervised-machine-

learning into the domain of prototyping recommender algorithms.

- Prove in theory, that SAG-RE has a convergence rate as fast as the original *SAG*; SAG-RE has asymptotic time complexity as efficient as any gradient method with the lowest iteration cost, and SAG-RE has asymptotic space complexity as compact as any memoryless gradient method.
- Extensively evaluate and compare SAG-RE across multiple RecSys objective functions and diverse datasets.
- Demonstrate in practice that, even without any optimization or fine-tuning on the implementation, SAG-RE still yields the best optimization within the shortest time despite the additional time of re-computation, and that SAG-RE uses memory at a level similar to the memory-less stochastic gradient.
- Provide an follow-up evidence that both full deterministic gradient and stochastic gradient takes much longer to reach a quality of optimization similar to SAG-RE.

2. BACKGROUND AND TERMINOLOGY

To motivate our paper and the space complexity challenge, we first introduce the background and the terminology that we use.

Matrix Factorization. Model-based recommender systems approximate the *user-item* matrix A through the dot-product of the *user*-matrix U and the *item*-matrix V: $\hat{A} = U * V$.

The *user-item* matrix A is a *nRows*-by-*nCols* matrix. A can be *sparse*; thus we use N to indicate the number of non-zero entries in A.

The approximation matrix $\hat{A}$ also has *nRows* rows, and *nCols* columns. $\hat{A}$ is not a sparse matrix. The goal of model-based recommendation is to use the non-zero entries to approximate the missing entries in A.

The *user* matrix U is *nRows*-by-*nDims*: U has *nRows* rows, and *nDims* columns. *nDims* is the number of latent dimensions. The *item* matrix V is *nDims*-by-*nCols*.

Optimizing an Objective Function. The goal of matrix factorization is to find the best U and the best V whose dot product optimizes an objective function:

$$\arg\min_{U,V}(\text{or}\arg\max)\left[f(U,V) = \sum_{i=1}^{nRows}\sum_{j=1}^{nCols} f(\bar{u}_i, \bar{v}_j)\right] \quad (1)$$

When we take the gradient of the objective function with respect to a row in the *user* matrix U (e.g. $\bar{u}_i$), we sum up the gradient of all the entries in $\hat{A}$ that belong to the same row $\bar{u}_i$.

$$\frac{\mathrm{d}f(U,V)}{\mathrm{d}\bar{u}_i} = \sum_{j=1}^{nCols} \frac{\mathrm{d}f(\bar{u}_i, \bar{v}_j)}{\mathrm{d}\bar{u}_i} \quad (2)$$

Similarly, when we take the gradient with respect to a column of V (e.g. $\bar{v}_j$), we sum up the gradients across different rows that belong to the same column:

$$\frac{\mathrm{d}f(U,V)}{\mathrm{d}\bar{v}_j} = \sum_{i=1}^{nRows} \frac{\mathrm{d}f(\bar{u}_i, \bar{v}_j)}{\mathrm{d}\bar{v}_j} \quad (3)$$

Both $\frac{\mathrm{d}f(\bar{u}_i,\bar{v}_j)}{\mathrm{d}\bar{u}_i}$ and $\frac{\mathrm{d}f(\bar{u}_i,\bar{v}_j)}{\mathrm{d}\bar{v}_j}$ are vectors of length *nDims*, the number of latent dimensions. Specifically, $\frac{\mathrm{d}f(\bar{u}_i,\bar{v}_j)}{\mathrm{d}\bar{u}_i}$ is a *1*-by-*nDims* row vector; $\frac{\mathrm{d}f(\bar{u}_i,\bar{v}_j)}{\mathrm{d}\bar{v}_j}$ is a *nDims*-by-*1* column vector. Similarly, $\frac{\mathrm{d}f(U,V)}{\mathrm{d}\bar{u}_i}$ is a row vector, and $\frac{\mathrm{d}f(U,V)}{\mathrm{d}\bar{v}_j}$ is a column vector, of length *nDims*.

In *SAG*, storing only the summed-up gradients is not sufficient for matrix factorization. The reason is that, each iteration of *SAG* requires the fine-grain gradients of individual entries (e.g. $\frac{\mathrm{d}f(\bar{u}_i,\bar{v}_j)}{\mathrm{d}\bar{u}_i}$ and $\frac{\mathrm{d}f(\bar{u}_i,\bar{v}_j)}{\mathrm{d}\bar{v}_j}$) that we previously sampled at an iteration before *t*. As we will prove, when directly applied to matrix factorization without using our SAG-RE approach, *SAG* will have a asymptotic space complexity of $\theta(nDims*(\min(M,N)+nRows+nCols))$. M is the number of *distinct* samples we have previously visited. Usually, $M = B * t$, where B is the batch size per iteration, and t is the number of iterations previously done.

Here, we want to point out that SAG-RE preserves the low asymptotic time complexity as *SAG*; and SAG-RE reduces asymptotic space complexity to $\theta(nDims*(nRows+nCols)+N)$. We will prove that this asymptotic space complexity is as compact as any memory-less approach.

Gradient Methods in Matrix Factorization. Gradient methods are iterative methods of optimization. When we increase the number of iterations, we expect the quality of optimization to also increase over time. At each iteration, gradient methods sample a batch of B entries, calculate the gradients of these entries, and use the calculated gradients to update U and V for the next iteration:

$$U^{t+1} = U^t + \frac{\alpha^t}{B}\left(\sum_{b=1}^{B} \frac{\mathrm{d}f(\bar{u}_{entry(b).i}, \bar{v}_{entry(b).j})}{\mathrm{d}\bar{u}_{entry(b).i}}\right) \quad (4)$$

$$V^{t+1} = V^t + \frac{\alpha^t}{B}\left(\sum_{b=1}^{B} \frac{\mathrm{d}f(\bar{u}_{entry(b).i}, \bar{v}_{entry(b).j})}{\mathrm{d}\bar{v}_{entry(b).j}}\right) \quad (5)$$

At iteration *t*, U^t is the current approximation of U. We use the gradients of the sampled batch of entries to update U^t into U^{t+1} for iteration *t+1*.

α^t is the *learning-rate* or *step-size*, at iteration *t*. When the goal of our optimization is to maximize an objective function, we apply *gradient-ascent* on U and V; thus we set $\alpha^t > 0$. When we try to minimize an objective function, we apply *gradient-descent* and set $\alpha^t < 0$.

entry(b) is the *b*-th entry in our batch of samples. *entry(b).i* is the *row* number of the entry; *entry(b).j* is the *column* number of the entry sampled from A.

Full deterministic gradient (*FG*) takes all N samples at each iteration; $B = N$ in *FG*. Stochastic gradient (*SG*) takes only one or a few samples per iteration: B is usually a constant much less than N.

Stochastic Average Gradient. Stochastic Average Gradient (*SAG*) requires a memory of previously-computed gradients: e.g. $\bar{m}_U^t$ and $\bar{m}_V^t$ for matrix factorization. At each iteration, *SAG* uses the batch of samples to update the memory, and applies the updated memory $\bar{m}_U^{t+1}$ and $\bar{m}_V^{t+1}$ on U and V for gradient descent or gradient ascent:

$$\bar{m}_{entry(b).i}^{t+1} = \frac{\mathrm{d}f(\bar{u}_{entry(b).i}, \bar{v}_{entry(b).j})}{\mathrm{d}\bar{u}_{entry(b).i}} \quad (6)$$

$$\bar{m}_U^{t+1} = \bar{m}_U^t + \sum_{b=1}^{B}\left[\bar{m}_{entry(b).i}^{t+1} - \bar{m}_{entry(b).i}^{t}\right] \quad (7)$$

$$U^{t+1} = U^t + \frac{\alpha^t}{B}\left(\bar{m}_U^{t+1}\right) \tag{8}$$

$$\bar{m}_{entry(b).j}^{t+1} = \frac{\mathrm{d}f(\bar{u}_{entry(b).i}, \bar{v}_{entry(b).j})}{\mathrm{d}\bar{v}_{entry(b).j}} \tag{9}$$

$$\bar{m}_U^{t+1} = \bar{m}_U^t + \sum_{b=1}^{B}\left[\bar{m}_{entry(b).j}^{t+1} - \bar{m}_{entry(b).j}^t\right] \tag{10}$$

$$V^{t+1} = V^t + \frac{\alpha^t}{B}\left(\bar{m}_V^{t+1}\right) \tag{11}$$

$\bar{m}_{entry(b).i}^t$ and $\bar{m}_{entry(b).j}^t$ are the memory of previously computed gradients. Thus both $\bar{m}_{entry(b).i}^t$ and $\bar{m}_{entry(b).j}^t$ are vectors of length *nDims*, the number of latent dimensions.

We apply *SAG* into matrix factorization for two reasons. First, *SAG* has iteration cost as low as stochastic gradient (*SG*). Second, *SAG*'s convergence rate is faster than *SG*, and sometimes as fast as full deterministic gradient (*FG*). At a high level, a faster convergence rate implies a better optimization in a shorter amount of time when data-scientists prototype model-based recommender systems. In this paper, we minimize the drawbacks or costs of using *SAG* in matrix factorization while preserving *SAG*'s benefits.

3. CHALLENGE

A main challenge is to make the fine-grain gradients of previously-sampled entries available: e.g. from equation 7, and from equation 10

4. FUTURE WORK & CONCLUSION

This paper is the first in the series of our study on data scientists prototyping model-based recommender systems. We explored the convex-optimization perspective of the problem: we propose Stochastic Average Gradient as a viable alternative to Full Deterministic gradient and Stochastic gradient. In theory, we proved that our extension and adaptation of *SAG* preserves the fast convergence rate as the original *SAG*. Furthermore, SAG-RE has both asymptotic time complexity and asymptotic space complexity as efficient as stochastic gradient. In practice, through extensive evaluation we demonstrated that, even without any fine-tuning or optimization of the implementation, SAG-RE still outperforms both full deterministic gradient and stochastic gradient in terms of reaching the best quality optimization within the same amount of time. Following up, we provided evidence that full deterministic gradient and stochastic gradient would take much longer to reach a quality of optimization similar to SAG-RE.

Currently we are extending SAG-RE in two directions. Both directions relate to running an iteration of full deterministic gradient in SAG-RE. First, we are investigating if it is beneficial to run an iteration of full deterministic gradient more often. In our experiments, we observed that both *SG* and *SAG* may converge early; the optimization may get stuck at a local sub-optimum for a long number of iterations. Thus we are exploring if an iteration of full deterministic gradient would get the optimization back on track in case SAG-RE gets stuck. Secondly, we aim to investigate how well SAG-RE would perform in the production environment, and in distributed systems potentially running in parallel, because running full deterministic gradient even once can be prohibitive for full-scale datasets with millions to billions of non-zero entries.

In the future, we also aim to complete our ongoing work on the metrics perspective and on the software engineering perspective. Given a dataset, the quality of a recommender system is often evaluated in various metrics: e.g. precision, recall, area under curve, reciprocal rank, NDCG, and variants of the above such as top-K precision and top-K hit rate. Many papers in the literature claim their objective function is better by illustrating that their objective function performs in some of these metrics better than other objective functions. Therefore, in the metrics perspective, we are exploring and investigating which factors are more relevant and important towards scoring high in the various metrics: is it the objective function, the method for convex-optimization such as *SAG*, other fine-tuning mechanisms such as bootstrapping, or the hyper parameters that we use in convex-optimization but which can also be specific to the dataset. In the software engineering perspective, we study how to increase the productivity of data scientists. At this point, we are designing and developing a *mix-n-match* or *plug-n-play* framework that enables data scientists in a least effort way, to very rapidly prototype and experiment many different combinations of objective functions, datasets, gradient methods, hyper parameters and evaluation metrics.

5. REFERENCES

[1] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[2] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

[3] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[4] J. D. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[5] N. L. Roux, M. Schmidt, and F. R. Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[6] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[7] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic. Gapfm: Optimal top-n recommendations for graded relevance domains. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2261–2266. ACM, 2013.

[8] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize

reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146. ACM, 2012.

[9] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–722. ACM, 2010.