HAMSI: Distributed Incremental Optimization Algorithm Using Quadratic Approximations for Partially Separable Problems

Umut Şimşekli[1], Hazal Koptagel[2], Figen Öztoprak[3], Ş. İlker Birbil[4], Ali Taylan Cemgil[2]
1: LTCI, CNRS, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France
2: Dept. of Computer Engineering, Boğaziçi University, 34342, Bebek, İstanbul, Turkey
3: Dept. of Industrial Engineering, Bilgi University, İstanbul, Turkey
4: Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey

Abstract

We present HAMSI, a provably convergent incremental algorithm for solving large-scale partially separable optimization problems that frequently emerge in machine learning and inferential statistics. The algorithm is based on a local quadratic approximation and hence allows incorporating a second order curvature information to speed-up the convergence. Furthermore, HAMSI needs almost no tuning, and it is scalable as well as easily parallelizable. In large-scale simulation studies with the MovieLens datasets, we illustrate that the method is superior to a state-of-the-art distributed stochastic gradient descent method in terms of convergence behavior. This performance gain comes at the expense of using memory that scales only linearly with the total size of the optimization variables. We conclude that HAMSI may be considered as a viable alternative in many scenarios, where first order methods based on variants of stochastic gradient descent are applicable.

1 Introduction

We present a distributed incremental method for solving problems of the form

$$\min_{x} f(x) \equiv \min_{x \in \mathbb{R}^{|\mathcal{J}|}} \sum_{i \in \mathcal{I}} f_i(x), \tag{1}$$

where each component function f_i for $i \in \mathcal{I}$ of the overall objective function f are twice continuously differentiable functions and $\mathcal{I} \equiv \{1, 2, \ldots, |\mathcal{I}|\}$ is an index set with typically a very large cardinality $|\mathcal{I}|$. Additionally, in many applications, each f_i depends only on a subset of the elements of x; that is, $f_i(x) = f_i(x_{\alpha_i})$. Then, the objective function in (1) is called *partially separable*. Here α_i are index sets such that for all $i \in \mathcal{I}$, $\alpha_i \subseteq \mathcal{J} \equiv \{1, 2, \ldots, |\mathcal{J}|\}$. Then, each singleton $j \in \mathcal{J}$ corresponds to a unique component of x, denoted as x_j. Thus, when $\alpha = \{j_1, j_2, \ldots, j_A\}$, we have a vector $x_\alpha = (x_{j_1}, x_{j_1}, \ldots, x_{j_A})$. This notation allows us to rewrite our overall problem as

$$\min_{x \in \mathbb{R}^{|\mathcal{J}|}} \sum_{i \in \mathcal{I}} f_i(x_{\alpha_i}). \tag{2}$$

The rather generic form given by (2) covers various optimization problems arising in machine learning, data mining or inferential statistics. A simple example clarifies the notation.

Example 1.1 *Consider the following matrix factorization problem:*

$$\min_{x} \left\| \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \\ y_5 & y_6 \end{pmatrix} - \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \begin{pmatrix} x_4 & x_5 \end{pmatrix} \right\|_F^2$$

where $\|\cdot\|_F$ is the Frobenius norm. Then using our notation, the objective function becomes

$$\sum_{i\in\mathcal{I}} f_i(x_{\alpha_i}) = (y_1 - x_1x_4)^2 + (y_2 - x_1x_5)^2 + \cdots + (y_6 - x_3x_5)^2,$$

where $\mathcal{I} = \{1, 2, \ldots, 6\}$ and $\mathcal{J} = \{1, 2, \ldots, 5\}$ with the subsets $\alpha_1 = \{1, 4\}$, $\alpha_2 = \{1, 5\}$, $\alpha_3 = \{2, 4\}$, $\alpha_4 = \{2, 5\}$, $\alpha_5 = \{3, 4\}$, and $\alpha_6 = \{3, 5\}$.

In this paper, our aim is to come up with a distributed and parallel algorithm. To serve this purpose, we further define a two level partitioning of the component functions. Formally, we let $\mathcal{I} = \bigcup_{k=1}^{K} \bigcup_{b=1}^{B_k} S_{k,b}$, where $S_{k,b} \cap S_{k,b'} = \emptyset$ for all k and $b \neq b'$. Hence, (1) is written as

$$\min_{x\in\mathbb{R}^{|\mathcal{J}|}} \sum_{k=1}^{K} \sum_{b=1}^{B_k} \sum_{i\in S_{k,b}} f_i(x_{\alpha_i}). \tag{3}$$

Our distributed algorithm relies on the fact that the objective function in (3) is separable over the second summation indexed by the block index b. This important point becomes clear, if we define

$$\boldsymbol{\alpha}_{k,b} \equiv \bigcup_{i\in S_{k,b}} \alpha_i \quad \text{for all } k = 1, \ldots, K; b = 1, \ldots, B_k,$$

with $\boldsymbol{\alpha}_{k,b} \cap \boldsymbol{\alpha}_{k,b'} = \emptyset$ for $b \neq b'$ and $\bigcup_{b=1}^{B_k} \boldsymbol{\alpha}_{k,b} \subseteq \mathcal{J}$ for all $k = 1, \ldots, K$. Then, we have

$$f_{k,b}(x_{\boldsymbol{\alpha}_{k,b}}) = \sum_{i\in S_{k,b}} f_i(x_{\alpha_i}). \tag{4}$$

This construction leads to the final form of our optimization problem that we shall consider in the subsequent part of our discussion:

$$\min_{x\in\mathbb{R}^{|\mathcal{J}|}} \sum_{k=1}^{K} \sum_{b=1}^{B_k} f_{k,b}(x_{\boldsymbol{\alpha}_{k,b}}). \tag{5}$$

The problem (5) is generally hard to solve because all or some of the terms in the objective function are non-convex. Due to lack of convexity, we can at best hope finding a local minimum to this problem. Still, several standard unconstrained optimization methods like gradient descent can be employed to obtain a local solution. However, in many applications, the cardinalities of the index sets $\mathcal{I}$ and $\mathcal{J}$ are very large, that makes even evaluating the objective function very costly.

In such settings (random) incremental and incremental aggregated methods can be used as the objective function is the sum of a finite number of functions [1, 2, 3, 4, 5]. In this study, we present an incremental and parallel algorithm that incorporates (approximate) curvature information for distributed large-scale optimization. Our experiences have confirmed that using second order information can help fast convergence even with incremental gradients. To gather second order information, the inner problems of our algorithm are modeled by quadratic functions. Similar to incremental and aggregate methods, our algorithm exploits the structure of the objective function and evaluates the gradient only for a subset of the component functions at each iteration, and it chooses the subsets of component functions in a way that provides separability of the inner problems. This helps to distribute the computations over a cluster of computers and enables doing step computations on subdomains in parallel.

The idea of second order incremental methods has been investigated before. Bertsekas proposed a method specifically designed for the least squares problem [6]. An extension of this method for general functions has recently been proposed by Gürbüzbalaban et al. [7]. They have shown linear convergence for the method under strong convexity and gradient growth assumptions. Moreover, their method requires the computation and inversion of exact Hessian matrices of component functions. In another study [8], an incremental aggregated quasi-Newton algorithm has been proposed, where the main idea is to update the quadratic model of one component function at each iteration.

2 Proposed Algorithm

The proposed algorithm uses incremental gradients and incorporates a second order information into the optimization steps. This second order information comes from an approximation to the Hessian

Algorithm 1: HAMSI (Hessian Approximated Multiple Subsets Iteration)

1 **input**: η, γ, $x^{(1)}$
2 $t \leftarrow 1$
3 **repeat**
4 Update $\beta^{(t)}$
5 $z^{(t,1)} \leftarrow x^{(t)}$
6 $H^{(t)} \leftarrow$ An approximate Hessian matrix at $x^{(t)}$
7 **for** $k = 1, 2, \cdots, K$ **do**
8 **for** $b = 1, 2, \ldots B_k$ **do in parallel**
9 $z^{(t,k)}_{\boldsymbol{\alpha}_{k,b}} \leftarrow \arg\min_z Q(z; z^{(t,k)}_{\boldsymbol{\alpha}_{k,b}}, \nabla f_{k,b}(z^{(t,k)}_{\boldsymbol{\alpha}_{k,b}}), [H^{(t)}]_{\boldsymbol{\alpha}_{k,b}}, \beta^{(t)})$
10 $z^{(t,k+1)} \leftarrow z^{(t,k)}$
11 $x^{(t+1)} \leftarrow z^{(t,K+1)}$
12 $t \leftarrow t + 1$
13 **until** *convergence or* $t >$ *max_epochs*

of the objective function. As we also work on multiple subsets of $|\mathcal{I}|$ functions, the algorithm is aptly called Hessian Approximated Multiple Subsets Iteration (HAMSI).

The key idea of the algorithm is using a local *convex* quadratic approximation

$$Q(z; \bar{x}, g, H, \beta) \equiv (z - \bar{x})^\top g + \frac{1}{2}(z - \bar{x})^\top H (z - \bar{x}) + \frac{1}{2}\beta \|z - \bar{x}\|^2 \tag{6}$$

for step computation. Here, g is an incremental gradient, H is (an approximation to) the Hessian of the objective function. The parameter β is crucial not only to bound the step length but also to control the oscillation of the incremental steps.

Algorithm 1 gives the generic form of HAMSI. We denote the k^{th} inner iterate of the t^{th} outer iteration with $z^{(t,k)}$, and $x^{(t)}$ are the outer iterates. It is important to note that the inner loop in Algorithm 1 (lines 8-9) computes the blocks of each inner step in parallel. The algorithm passes through the subsets of component functions in a cyclic manner. Once a cycle is complete, one outer iteration is finished and the outer iterate is updated (line 11). Note that the same (approximate) matrix $H^{(t)}$ is employed at all inner iterations during the t^{th} cycle. However, the inner iterations use different blocks of $H^{(t)}$ denoted by the submatrix $[H^{(t)}]_{\boldsymbol{\alpha}_{k,b}}$, where $[H]_{\boldsymbol{\alpha}} = \{H(i,i') : i, i' \in \boldsymbol{\alpha}\}$. $\beta^{(t)}$ is also constant during the inner iterations and it is updated with each outer iteration (line 4).

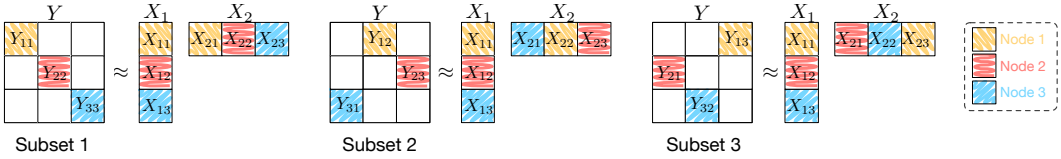
Above description of the algorithm overlooks several important implementation details; in particular, how to construct the quadratic approximation and how to solve the corresponding subproblems. The curious reader is referred to the longer version of this paper [9], in which we provide a convergence proof for HAMSI under the quite generic setting given in Algorithm 1. In [9], we also provide an example implementation of HAMSI where the approximate Hessian matrices $H^{(t)}$ are obtained using BFGS quasi-Newton update formula. In particular, the compact form of limited memory BFGS (L-BFGS) [10] is used in inner iterations to form the quadratic models, and to obtain their analytical solutions directly. L-BFGS allows the computation of $(H^{(t)} + \beta^{(t)}\mathbb{I})^{-1}v$ for a given vector v without forming any $|\mathcal{J}| \times |\mathcal{J}|$ matrices, and without any $\mathcal{O}(|\mathcal{J}|^2)$ operations.

3 Application on Distributed Matrix Factorization

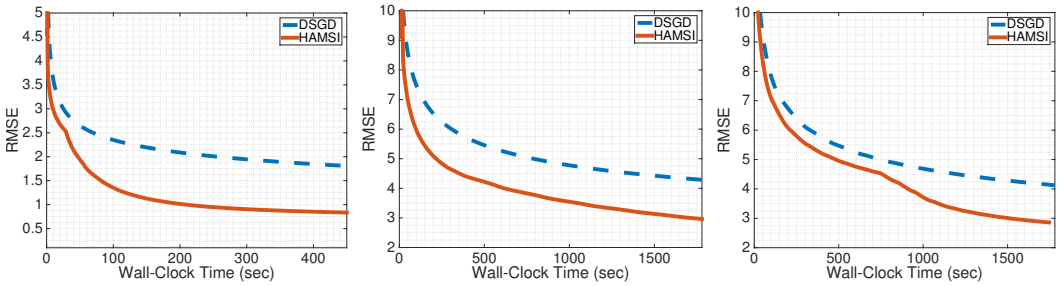
In this section, we present the performance of HAMSI on a large-scale, distributed matrix factorization (MF) application. The aim of a MF model is to decompose an observed data matrix $Y \in \mathbb{R}^{I \times J}$ in the form: $Y \approx X_1 X_2$, where $X_1 \in \mathbb{R}^{I \times P}$ and $X_2 \in \mathbb{R}^{P \times J}$ are the factor matrices, known typically as the dictionary and the weight matrix, respectively. A typical example with quadratic error is given as follows:

$$(X_1, X_2)^\star = \arg\min \|Y - X_1 X_2\|_F^2. \tag{7}$$

The relation to problem (1) becomes clear as we set $|\mathcal{I}| = IJ$, $|\mathcal{J}| = IP + JP$, $x \equiv [\mathbf{vec}(X_1)^\top, \mathbf{vec}(X_2)^\top]^\top$.



(a) Illustration of the *subsets* and the *blocks*. Given the blocks in a subset, the corresponding blocks in X_1 and X_2 become conditionally independent, as illustrated in different textures.



(b) MovieLens 1M, $K = 8$ (c) MovieLens 10M, $K = 15$ (d) MovieLens 20M, $K = 60$

Figure 1: (a) Partitioning schema (b)-(d) RMSE values on MovieLens datasets.

Figure 1(a) illustrates the partitioning schema that we use in our implementation. This idea of partitioning has already been studied previously in the literature [11, 12, 13]. In this example, the observed matrix Y is partitioned into $K = 3$ disjoint subsets, where each subset is further partitioned into $B_1 = B_2 = B_3 = 3$ blocks. The latent factors X_1 and X_2 are partitioned accordingly into 3 blocks each.

We have implemented HAMSI in C using OpenMPI. In our implementation, each subset has the same number of blocks $B_k = K$, where K is also the number of available nodes. We evaluate HAMSI on three large movie ratings datasets, namely MovieLens 1M, 10M, and 20M (`grouplens.org`), where these datasets contain 1, 10, and 20 million ratings, respectively. In all our experiments, we set latent dimension, $P = 50$ and L-BFGS memory size, $M = 5$. Further details about the experimental setup can be found in [9].

We compare HAMSI with the state-of-the-art distributed optimization algorithm for MF, namely, the distributed stochastic gradient descent (DSGD) [12]. In this experiment, on each dataset, we report the root mean squared error (RMSE) between Y and X_1X_2 after running each algorithm for a fixed computation time. Figures 1(b)-1(d) shows the RMSE values of HAMSI and DSGD on the three datasets as function of wall-clock time. A single iteration of HAMSI is computationally heavier than DSGD: First, the compact form L-BFGS update requires more computation than simple gradient evaluation. Formally, this overhead is in the order of $\mathcal{O}((M^2 \max(I, J)P)/K^2)$. Second, the communication cost of HAMSI is $\mathcal{O}(JP(2M+1)/K)$ per iteration, whereas the communication cost of DSGD is $\mathcal{O}(JP/K)$. However, the use of second order information compensates quickly for this slight increase in computational complexity as it helps HAMSI converge much faster than DSGD. This is clearly seen by the significant gap in RMSE values between the two methods.

4 Conclusion

We have presented HAMSI, a provably convergent distributed incremental quasi-Newton algorithm for unconstrained optimization. HAMSI is particularly suited for large-scale optimization problems where the overall objective function can be written as the sum of a large number of component functions, and each component function depends only on a subset of the optimization variables. Such structured non-separable problems are ubiquitous in machine learning; besides matrix factorization problems, maximum a-posteriori state estimation in certain exponential family graphical models have also this form. The algorithm is scalable as neither the exact gradient nor an approximate Hessian matrix of the original objective is explicitly required and is easily parallelizable on modern distributed computing infrastructures. Our main conclusion is that HAMSI may be considered as a viable alternative in many scenarios where first order methods based on variants of stochastic gradient descent are applicable.

References

[1] Dimitri P Bertsekas, "Incremental gradient, subgradient, and proximal methods for convex optimization: A survey," *Optimization for Machine Learning*, vol. 2010, pp. 1–38, 2011.

[2] Paul Tseng, "An incremental gradient (-projection) method with momentum term and adaptive stepsize rule," *SIAM Journal on Optimization*, vol. 8, no. 2, pp. 506–531, 1998.

[3] Mikhail V Solodov, "Incremental gradient algorithms with stepsizes bounded away from zero," *Computational Optimization and Applications*, vol. 11, no. 1, pp. 23–35, 1998.

[4] Doron Blatt, Alfred O Hero, and Hillel Gauchman, "A convergent incremental gradient method with a constant step size," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 29–51, 2007.

[5] Nicolas L Roux, Mark Schmidt, and Francis R Bach, "A stochastic gradient method with an exponential convergence _rate for finite training sets," in *Advances in Neural Information Processing Systems*, 2012, pp. 2663–2671.

[6] Dimitri P Bertsekas, "Incremental least squares methods and the extended Kalman filter," *SIAM Journal on Optimization*, vol. 6, no. 3, pp. 807–822, 1996.

[7] Mert Gürbüzbalaban, Asuman Ozdaglar, and Pablo Parrilo, "A globally convergent incremental Newton method," *arXiv preprint arXiv:1410.5284*, 2014.

[8] Jascha Sohl-Dickstein, Ben Poole, and Surya Ganguli, "Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods," in *Proceedings of the 31th International Conference on Machine Learning (ICML)*, 2014, pp. 604–612.

[9] Umut Şimşekli, Hazal Koptagel, Figen Öztoprak, Ş İlker Birbil, and Ali Taylan Cemgil, "Hamsi: Distributed incremental optimization algorithm using quadratic approximations for partially separable problems," *arXiv preprint arXiv:1509.01698*, 2015.

[10] Richard H Byrd, Jorge Nocedal, and Robert B Schnabel, "Representations of quasi-Newton matrices and their use in limited memory methods," *Mathematical Programming*, vol. 63, no. 1-3, pp. 129–156, 1994.

[11] Chao Liu, Hung chih Yang, Jinliang Fan, Li-Wei He, and Yi-Min Wang, "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce," in *Proceedings of the 19th International World Wide Web Conference*, April 2010.

[12] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *ACM SIGKDD*, 2011.

[13] Benjamin Recht and Christopher Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 201–226, 2013.