Expected Tensor Decomposition with Stochastic Gradient Descent

Takanori Maehara[1,3] **Kohei Hayashi**[2,3] **Ken-ichi Kawarabayashi**[2,3]
1) Shizuoka University, Shizuoka, Japan
2) National Institute of Informatics, Tokyo, Japan
3) JST, ERATO, Kawarabayashi Large Graph Project

Abstract

In this study, we investigate *expected CP decomposition*—a special case of CP decomposition in which a tensor to be decomposed is given as the sum or average of tensor samples $\mathcal{X}^{(t)}$ for $t = 1, \ldots, T$. To determine this decomposition, we develope stochastic-gradient-descent-type algorithms with four appealing features: efficient memory use, ability to work in an online setting, robustness of parameter tuning, and simplicity. Our theoretical analysis show that the solutions do not diverge to infinity for any initial value or step size. Experimental results confirm that our algorithms significantly outperform all existing methods in terms of accuracy. We also show that they can successfully decompose a large tensor, containing billion-scale nonzero elements.

1 Introduction

A tensor represents data that have multiple indices or *modes*, such as co-occurrence frequencies among `subjects`, `objects`, and `verbs` in documents (Kang et al. 2012; Fried, Polajnar, and Clark 2015) or climate data for separate `latitude`, `longitude`, and `measurement` types (e.g., temperature and the velocity of wind) (Bahadori, Yu, and Liu 2014). *CP decomposition* (Carroll and Chang 1970; Harshman 1970), also known as canonical decomposition (CANDECOMP) or parallel factor (PARAFAC), is a key tool for analyzing such data tensors. Given a three-mode tensor[1] $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, rank-$R$ CP decomposition is written as

$$\mathcal{X}_{ijk} \approx [\![U, V, W]\!] := \sum_{r=1}^{R} U_{ir} V_{jr} W_{kr}, \qquad (1)$$

where $U \in \mathbb{R}^{I \times R}$, $V \in \mathbb{R}^{J \times R}$, and $W \in \mathbb{R}^{K \times R}$ are factor matrices capturing linear relationships in $\mathcal{X}$. CP decomposition is used for feature extraction, missing-value prediction, data compression, and visualization in many application fields (Kolda and Bader 2009).

1.1 Motivation

In some application fields, $\mathcal{X}$ is given as the *sum* or *average of multiple tensors*. For example, co-occurrence frequencies $\mathcal{X}$ is the total number of all triplets (`subject`, `object`, `verb`) appearing in document $1, \ldots, T$. $\mathcal{X}$ is thus written as $\mathcal{X} = \sum_{t=1}^{T} \mathcal{X}^{(t)}$ where $\mathcal{X}^{(t)}$ is the co-occurrence tensor of document t. Another example appears in time series analysis. Suppose $\mathcal{X}^{(t)}$ represents the climate data at time t. Because the data often contain random fluctuations, averaging past T observations $\mathcal{X} = \frac{1}{T} \sum_{s=t-T}^{t} \mathcal{X}^{(s)}$ before decomposition is more reliable than decomposing $\mathcal{X}^{(t)}$ directly.

When $\mathcal{X}^{(t)}$ is independently and identically distributed, target tensors in such examples are generally written as expectation $\mathbb{E}[\mathcal{X}]$.[2] We term the CP decomposition of $\mathbb{E}[\mathcal{X}]$ *expected CP decomposition*. A straightforward approach to expected CP decomposition is to first construct $\mathbb{E}[\mathcal{X}]$ and then decompose it. However, this naive approach has several limitations. The first is scalability. In some applications (typically in natural language processing), $\mathcal{X}^{(t)}$ is very high-dimensional but sparse and is manageable with moderate memory space. However, $\mathbb{E}[\mathcal{X}]$ may be dense and may not fit in memory. Moreover, whenever $X^{(t+1)}$ is additionally observed, the naive approach needs to compute from scratch: by reconstructing $\mathbb{E}[\mathcal{X}]$, initializing the algorithm, and updating U, V, W, which is time consuming.

Arora et al. (Arora et al. 2012), in their seminal work, addressed the case of matrix decomposition and proposed a stochastic optimization algorithm to overcome these limitations. However, the stochastic optimization of expected CP decomposition is non-trivial because of its unstable convergence property. Arora et al. focused on a convex optimization problem to which the standard theory for stochastic gradient descent can be applied directly. However, CP decomposition is non-convex and ill-conditioned. This means that the initial solution and step size must be carefully selected to ensure that existing standard algorithms including stochastic gradient descent do not diverge[3]. Another issue is the slow convergence rate. In the case of non-stochastic CP decomposition, the gradient descent method has poor convergence properties (Comon, Luciani, and De Almeida 2009).

[1]Note that all the ideas of this study written for three-mode tensors also works for higher-order tensors.

[2]In the `subject`×`object`×`verb` case, rescaling by T is necessary.

[3]We say diverge for "diverge to infinity".

1.2 Contributions

As the first attempt to solve the expected CP decomposition problem, we analyzed its stochastic formulation and optimization. The key idea is that the CP decomposition of $\mathbb{E}[\mathcal{X}]$ can be equivalently formulated as a stochastic optimization problem (Section 3). From this notion, we derive three stochastic gradient algorithms (Section 4). First, we derive *stochastic gradient descent* (*SGD*) as a baseline. Next, we extend *SGD* to *second order stochastic gradient descent* (*2SGD*) and *stochastic alternating least squares* (*SALS*). Our theoretical analysis shows that *2SGD* and *SALS* have convergence rates faster than that of *SGD* while the computational complexity remains roughly the same. Our algorithms have the following attractive features:

(1) A good space-time tradeoff. Only the memory space for storing $\mathcal{X}^{(t)}$, U,V, and W is required at time t. This enables us to decompose $\mathbb{E}[\mathcal{X}]$ with sparse $\mathcal{X}^{(t)}$ even if $\mathbb{E}[\mathcal{X}]$ exceeds the memory space. Therefore, our approach is advantageous over parallel/distributed methods such as (Kang et al. 2012; Jeon et al. 2015) since our algorithms can work in a single machine and hence there is no additional overhead such as communication cost.

(2) Online learning. Decomposition for a new sample $X^{(t+1)}$ is updated in a single step. This is particularly useful in an online setting, i.e., when $\mathcal{X}^{(t)}$ is observed and decomposes $\mathbb{E}[\mathcal{X}]$ for every time t.

(3) Robustness against parameter tuning (for *2SGD* and *SALS*). The use of second-order information ensures that the solution does not diverge for any initial solution or step size. Moreover, the performance is robust to the step size parameter and careful parameter tuning is not necessary.

(4) Simplicity. The update equations are quite simple and easy to implement. This may be beneficial for further extensions such as generalizing the loss function (e.g., to the absolute error or Kullback–Leibler divergence) and computing in a distributed way.

The experimental results demonstrate that *2SGD* and *SALS* significantly outperform existing SGD algorithms. Also, our algorithms successfully decomposed the real tensor having 1.09 billion nonzero entries, whereas the standard algorithm failed because it exceeded the available memory.

Our contributions are highlighted as follows:

- Proposing expected tensor decomposition, a new tensor decomposition problem.
- Developing three stochastic algorithms.
- Analyzing convergence properties of our algorithms.

The second contribution is a reasonable extension of Hayashi et al. who addressed the case of matrix decomposition with nonnegative constraints for U and V (Hayashi et al. 2015). In contrast, the first and third contributions are new concepts. In particular, the idea of expected tensor decomposition substantially improves the space complexity of CP decomposition.

2 Preliminaries and Related work

2.1 CP decomposition

As mentioned above, the CP decomposition of the three-mode tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is given by (1). For any tensor, we can determine the exact decomposition of (1) for a sufficiently large R (e.g., $R = IJK$), but we are interested in a decomposition where R is small (typically $R \approx 10$). The notion of rank-R CP decomposition is then considered. Let

$$L(\mathcal{X}; U, V, W) = \frac{1}{2} \sum_{ijk} \left(\mathcal{X}_{ijk} - \sum_{r=1}^{R} U_{ir} V_{jr} W_{kr} \right)^2$$

be the squared loss function. The CP decomposition of rank R is obtained by solving the non-convex optimization problem:

$$\min_{U,V,W} L(\mathcal{X}; U, V, W).$$

However, this problem is "ill-conditioned" in the sense as follows:

1. The loss function L has continuously many local minima because of the indeterminacy of scaling. For example, $L(\mathcal{X}; U, V, W) = L(\mathcal{X}; aU, bV, W/ab)$.
2. The loss function L may not have a global minimum because the domain is non-compact (de Silva and Lim 2008; Chen and Saad 2009).

To solve these issues, Paatero (Paatero 2000) suggested the use of Tikhonov regularized loss function:

$$L_\rho(\mathcal{X}; U, V, W) = L(\mathcal{X}; U, V, W) + (\rho/2) \left(\|U\|^2 + \|V\|^2 + \|W\|^2 \right), \quad (2)$$

where $\rho > 0$ is a regularization parameter, and $\|\cdot\|$ denotes the Frobenius norm. The corresponding optimization problem is given by

$$\min_{U,V,W} L_\rho(\mathcal{X}; U, V, W). \quad (3)$$

This regularization resolves the abovementioned two issues as follows:

1. The regularized loss function L_ρ has isolated local optimal solutions, because each local optimal solution must satisfy $\|U\| = \|V\| = \|W\|$.
2. The regularized loss function L_ρ has a global optimal solution, because we can assume $\rho(\|U\|^2 + \|V\|^2 + \|W\|^2) \leq \|X\|^2$, which is a compact region.

Therefore, in the remainder of this paper, we address only regularized problem (3).

Several algorithms have been proposed to obtain CP decomposition. Among these algorithms, *alternating least squares* (ALS) is a widely used algorithm, which solves the least-square problem cyclically for components U, V, and W while keeping other components constant (Carroll and Chang 1970; Harshman 1970). Another approach is based on nonlinear optimization, which directly solves (3) using a standard optimization approach such as nonlinear conjugate gradient method (Acar, Dunlavy, and Kolda 2011) or

Gauss–Newton method (Hayashi and Hayashi 1982; Phan, Tichavský, and Cichocki 2013). For symmetric tensors, the power method (Kolda and Mayo 2011; Anandkumar et al. 2014) has also been used.

2.2 Stochastic gradient descent

Here, we describe the stochastic gradient method (see (Bottou 2004; Bottou and Bousquet 2007) for more details).

Let $L(x; u)$ be a function to be minimized, where u is a variable and x is data. Suppose that we can sample data $x^{(1)}, x^{(2)}, \ldots$ from a distribution. We consider the stochastic optimization problem:

$$\min \mathbb{E}[L(x; u)]. \tag{4}$$

The *stochastic gradient descent method*, described below, solves this problem:

$$u^{(t+1)} = u^{(t)} - \eta^{(t)} \frac{\partial L}{\partial u}(u^{(t)}; x^{(t)}), \tag{5}$$

where $\eta^{(t)}$ is a step size, usually set to $\eta^{(t)} = O(1/t)$. If L is convex in u and $\|\partial L/\partial u\|$ is bounded, the stochastic gradient descent method converges at rate $O(1/t)$ (Bottou 2004).

Second-order stochastic gradient descent (Bordes, Bottou, and Gallinari 2009; Bordes et al. 2010) is an extension of the stochastic gradient descent and is defined as

$$u^{(t+1)} = u^{(t)} - \eta^{(t)} H^{(t)} \frac{\partial L}{\partial u}(u^{(t)}; x^{(t)}), \tag{6}$$

where $H^{(t)}$ is a positive definite matrix (e.g., an approximate inverse Hessian). This method also converges under same conditions as those of the standard stochastic gradient descent while reducing the number of iterations by a constant factor.

Previous studies have combined CP decomposition and stochastic optimization. Rendle and Thieme (Rendle and Thieme 2010) proposed the following randomized algorithm for (non-stochastic) tensor decomposition, which was originally used by Koren, Bell, and Volinsky (Koren, Bell, and Volinsky 2009) for matrix decomposition. The algorithm iteratively samples an index (i, j, k) and updates U, V, and W by the gradient depending on $\mathcal{X}_{ijk}$, which corresponds to the case that each element is the random sample of stochastic optimization. While this algorithm works well in practice, its theoretical properties have not yet been completely understood.

Ge et al. (Ge et al. 2015) applied the stochastic optimization to the orthogonal tensor decomposition. By introducing new loss function, they produced several good theoretical results such as global convergence guarantees when a tensor is exactly decomposable; however, the results are not applicable to our case.

To the best of our knowledge, this is the first study addressing the stochastic formulation and optimization of CP decomposition where random samples are given as tensors.

3 Expected CP Decomposition

Now we extend CP decomposition to a stochastic setting analogous to the formulation of the stochastic gradient descent. Suppose that we observe random tensors $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(T)}$ for which the expectation is $\mathbb{E}[\mathcal{X}]$. To determine a decomposition of the form

$$\mathbb{E}[\mathcal{X}] \approx [\![U, V, W]\!], \tag{7}$$

we introduce *expected CP decomposition* as a solution to the stochastic optimization problem as follows:

$$\min_{U,V,W} \mathbb{E}[L_\rho(\mathcal{X}; U, V, W)], \tag{8}$$

where L_ρ is the regularized squared loss function, defined by (2). This indicates clearly that the expected loss and the loss of the expected tensor only differ by a constant, i.e.,

$$\mathbb{E}[L_\rho(\mathcal{X}; U, V, W)] = L_\rho(\mathbb{E}[\mathcal{X}]; U, V, W) + \sum_{ijk} \mathrm{Var}[\mathcal{X}_{ijk}].$$

The CP decomposition of the expected tensor $\mathbb{E}[\mathcal{X}]$ is then obtained by computing expected CP decomposition (8). Note that Hayashi et al. show the similar relationship in the matrix case (Hayashi et al. 2015).

Non-stochastic approaches such as ALS also provide the solution of (7) when we have the expectation $\mathbb{E}[\mathcal{X}]$ or its estimation (e.g., sample mean $(1/T)\sum_t \mathcal{X}^{(t)}$). However, memory usage is a crucial issue in this approach. Consider the case in which we confront a large-scale problem (i.e., I, J, K are large) where $\mathbb{E}[\mathcal{X}]$ is dense but each $\mathcal{X}^{(t)}$ is sparse. This situation typically arises in text mining applications where $\mathcal{X}^{(t)}$ represents word cooccurrences of the t-th document and $\mathbb{E}[\mathcal{X}]$ is the joint probability.

Clearly, the non-stochastic approach must store the complete information of $\mathbb{E}[\mathcal{X}]$ while the algorithm is executing, which demands massive memory space. In contrast, the stochastic approach only needs to store the non-zero elements of $\mathcal{X}^{(t)}$ in each iteration, and in each iteration the previous information can be discarded.

4 Proposed algorithms

In this section, we propose three algorithms. First, we give *SGD* as a direct adaptation of the general stochastic gradient descent method (5). Second, we propose *2SGD* as an extension of *SGD* to the general second order stochastic gradient descent method (6) with block-diagonal approximation of the Hessian. Finally, by combining the alternating optimization with *2SGD*, we obtain *SALS*.

4.1 SGD

To apply a stochastic gradient descent (5) to our problem (8), we must compute the gradient of a loss function (2). The gradient of a (non-regularized) loss function is sometimes called *CP gradient* (Acar, Dunlavy, and Kolda 2011; Phan, Tichavský, and Cichocki 2012), and the explicit formula is obtained as follows. Let $\mathcal{X}(\cdot, V, W)$ be a matrix obtained from $\mathcal{X}$ by contracting V and W:

$$\mathcal{X}(\cdot, V, W)_{ir} := \sum_{jk} \mathcal{X}_{ijk} V_{jr} W_{kr}.$$

Note that the size of $\mathcal{X}(\cdot, V, W)$ is $I \times R$, which is same as that of U. $\mathcal{X}(U, \cdot, W)$ and $\mathcal{X}(U, V, \cdot)$ are defined similarly. The gradient of the (non-regularized) loss function is given as follows.

Theorem 1 (CP gradient (Acar, Dunlavy, and Kolda 2011; Phan, Tichavský, and Cichocki 2012))**.**

$$\frac{\partial L}{\partial U}(\mathcal{X}; U, V, W) = -\mathcal{X}(\cdot, V, W) + U\Gamma(V, W), \tag{9}$$

where $\Gamma(A, B)$ *is the Hadamard product of* $A^\top A$ *and* $B^\top B$*, i.e.,* $\Gamma(A, B)_{ij} = (A^\top A)_{ij}(B^\top B)_{ij}$*. Similar formulas hold for* $\partial L/\partial V$ *and* $\partial L/\partial W$*.*

This theorem immediately yields the gradient for the regularized loss function as follows.

Corollary 2 ((Acar, Dunlavy, and Kolda 2011))**.**

$$\frac{\partial L_\rho}{\partial U}(\mathcal{X}; U, V, W) = -\mathcal{X}(\cdot, V, W) + U\Gamma_\rho(V, W), \tag{10}$$

where $\Gamma_\rho(A, B) := \Gamma(A, B) + \rho I$*. Similar formulas hold for* $\partial L_\rho/\partial V$ *and* $\partial L_\rho/\partial W$*.*

By substituting (10) into (5), we obtain the stochastic gradient descent method for expected CP decomposition. We refer to this basic algorithm as *SGD*:

$$\begin{aligned} U^{(t+1)} &= U^{(t)} - \eta^{(t)}\frac{\partial L_\rho}{\partial U} \\ &= U^{(t)}\left(I - \eta^{(t)}\Gamma_\rho(V^{(t)}, W^{(t)})\right) \\ &\quad + \eta^{(t)}\mathcal{X}^{(t)}(\cdot, V^{(t)}, W^{(t)}). \end{aligned} \tag{11}$$

The updates for $V^{(t)}$ and $W^{(t)}$ are similar to the abovementioned ones.

By denoting $|\mathcal{X}|$ as the number of nonzero elements in $\mathcal{X}$, the time complexity of *SGD* is $O((I + J + K)R^2 + |\mathcal{X}^{(t)}|R)$ for each update, where $O((I + J + K)R^2$ is for computing $U\Gamma_\rho(V, W)$ (matrix-matrix multiplication) and $O(|\mathcal{X}^{(t)}|R)$ is for computing $\mathcal{X}^{(t)}(\cdot, V, W)$ (tensor-matrix multiplication). The space complexity is $O((I+J+K)R+R^2 + |\mathcal{X}^{(t)}|)$, where $O((I + J + K)R)$ is for storing U, V, and W, $O(R^2)$ is for storing $\Gamma(V, W)$, $\Gamma(U, W)$, and $\Gamma(U, V)$, and $O(|\mathcal{X}^{(t)}|)$ is for storing each sample. Note that in some applications, such as the bag-of-words topic model (Anandkumar et al. 2014), each $\mathcal{X}^{(t)}$ is expressed as a rank-one tensor, i.e.,

$$\mathcal{X}^{(t)}_{ijk} = u^{(t)}_i v^{(t)}_j w^{(t)}_k.$$

In this case, even when $\mathcal{X}^{(t)}$ is dense, we can compute the tensor-matrix multiplication efficiently ($O((I + J + K)R)$) as

$$\mathcal{X}^{(t)}(\cdot, V, W)_{ir} = u_i(v^{(t)\top}V)_r(w^{(t)\top}W)_r.$$

Note that the algorithm of (Rendle and Thieme 2010) is a special case of *SGD*. More specifically, their algorithm applies to the case in which each sampled tensor $\mathcal{X}^{(t)}$ is given by

$$\mathcal{X}^{(t)} := \begin{cases} IJK\mathcal{X}_{ijk}, & i = i^{(t)}, j = j^{(t)}, k = k^{(t)}, \\ 0, & \text{otherwise}, \end{cases}$$

for some fixed tensor $\mathcal{X}$ and randomly sampled indices $(i^{(t)}, j^{(t)}, k^{(t)})$.

4.2 2SGD

Analogous to (6), we next derive *2SGD*, which is an extension of *SGD* with second-order information. By taking a derivative with respect to U on both sides of (10), we obtain

$$\frac{\partial^2 L_\rho}{\partial U_{ir}\partial U_{i'r'}} = \delta_{ii'}\Gamma_\rho(V, W)_{rr'},$$

where $\delta_{ii'}$ denotes the Kronecker delta, i.e., $\delta_{ii'} = 1$ if $i = i'$, and 0 otherwise. This shows that $\Gamma_\rho(V, W)$ is a block-diagonal component of the Hessian of L_ρ. More precisely, if we consider U as vector $[U_{11}, \dots, U_{1R}, \dots, U_{I1}, \dots, U_{IR}]$, the submatrix of the Hessian corresponding to U is $\mathrm{diag}(\Gamma_\rho(V, W), \dots, \Gamma_\rho(V, W))$. Similarly, $\Gamma_\rho(U, W)$ and $\Gamma_\rho(U, V)$ are also block-diagonal components of the Hessian. We use these matrices as an approximate Hessian (Hayashi et al. 2015) and obtain

$$\begin{aligned} U^{(t+1)} =& U^{(t)} - \eta^{(t)}\frac{\partial L}{\partial U}\Gamma_\rho(V^{(t)}, W^{(t)})^{-1} \\ =& U^{(t)}\left(1 - \eta^{(t)}\right) \\ &+ \eta^{(t)}\mathcal{X}^{(t)}(\cdot, V^{(t)}, W^{(t)})\,\Gamma_\rho(V^{(t)}, W^{(t)})^{-1}. \end{aligned} \tag{12}$$

The updates for $V^{(t)}$ and $W^{(t)}$ are similar to the abovementioned ones.

2SGD (12) has the following intuitive interpretation. By letting $\partial L_\rho/\partial U = 0$ in (10), the matrix

$$U^*(\mathcal{X}; V, W) := \mathcal{X}(\cdot, V, W)\Gamma_\rho(V, W)^{-1}, \tag{13}$$

which appears in the second term of (12), is the least-square solution of $L_\rho(\mathcal{X}; U, V, W) = 0$ by fixing V and W. Let V^* and W^* be defined similarly. Then (12) is expressed as

$$\begin{aligned} U^{(t+1)} &= (1 - \eta^{(t)})U^{(t)} + \eta^{(t)}U^*(\mathcal{X}^{(t)}; V^{(t)}, W^{(t)}), \\ V^{(t+1)} &= (1 - \eta^{(t)})V^{(t)} + \eta^{(t)}V^*(\mathcal{X}^{(t)}; U^{(t)}, W^{(t)}), \\ W^{(t+1)} &= (1 - \eta^{(t)})W^{(t)} + \eta^{(t)}W^*(\mathcal{X}^{(t)}; U^{(t)}, V^{(t)}). \end{aligned} \tag{14}$$

An intuitive meaning of these update equations is: "the next solution is a weighted average of the current solution and the least-square solution."

Note that the complexity of one update of *2SGD* is almost the same as that of *SGD* (11) — it only requires additional $O(R^3)$ time to compute the inverse of $R \times R$ matrices $\Gamma_\rho(V, W)$, $\Gamma_\rho(U, W)$ and $\Gamma_\rho(V, W)$. This additional cost is usually negligible because R is much smaller than I, J, and K. Moreover, note that the inverses of these matrices always exist, because the Hadamard product of two positive semidefinite matrices is a positive semidefinite matrix (Bapat and Raghavan 1997) thus $\Gamma_\rho(V, W) := \Gamma(V, W) + \rho I$ is positive definite (i.e., invertible).

4.3 SALS

Our third algorithm is a cyclic variant of *2SGD*. By modifying (14) to be cyclically updated, we obtain

$$\begin{aligned} U^{(t+1)} &= (1 - \eta^{(t)})U^{(t)} + \eta^{(t)}U^*(\mathcal{X}^{(t)}; V^{(t)}, W^{(t)}), \\ V^{(t+1)} &= (1 - \eta^{(t)})V^{(t)} + \eta^{(t)}V^*(\mathcal{X}^{(t)}; U^{(t+1)}, W^{(t)}), \\ W^{(t+1)} &= (1 - \eta^{(t)})W^{(t)} + \eta^{(t)}W^*(\mathcal{X}^{(t)}; U^{(t+1)}, V^{(t+1)}), \end{aligned} \tag{15}$$

which can be interpreted as: "update each component as a weighted average of the current solution and the least-square solution, *alternatingly*." We refer to this algorithm as *SALS*. Because *SALS* is a rearranged version of *2SGD*, the time and space complexity of *SALS* and *2SGD* are the same.

As shown in our experiments, the performance of *2SGD* and *SALS* is basically the same, but we expect *SALS* to be more stable. This will be explained in the next section.

5 Analysis

First, we show that *SALS* has significant advantage over *SGD* in terms of performance stability.

Proposition 3. *Suppose that* $\sup_t \|\mathcal{X}^{(t)}\|^2 \leq C$. *For any initial solution and any step size,* SALS *and* 2SGD *do not diverge.*

Proof. We prove this only for *SALS*; notably, a similar proof works for *2SGD*. Consider the t-th update for $U^{(t)}$. Recall that $U^{(t+1)}$ is a convex combination of $U^{(t)}$ and U^*, where U^* is a solution of the least-square problem. Then, we have

$$L_\rho(\mathcal{X}^{(t)}; U^*, V^{(t)}, W^{(t)}) \leq L_\rho(\mathcal{X}^{(t)}; U, V^{(t)}, W^{(t)}),$$

for all U. By substituting $U = 0$, we obtain

$$2\rho\|U^*\| \leq \|\mathcal{X}^{(t)}\|^2 \leq C.$$

According to the definition of $U^{(t+1)}$, we have

$$\|U^{(t+1)}\| \leq \max\{\|U^{(t)}\|, \frac{C}{2\rho}\} \leq \max\{\|U^{(0)}\|, \frac{C}{2\rho}\}.$$

This shows that $U^{(t)}$ is contained in a compact set. The same results hold for $\|V^{(t)}\|$ and $\|W^{(t)}\|$. □

Let us emphasize that this proposition does not guarantee the convergence of the algorithm (i.e., solutions may oscillate forever). Nevertheless, this property is useful in practice. For a non-convex problem, stochastic gradient descent requires a careful selection of an initial solution and step size to obtain a solution. In contrast, based on this proposition, these settings are not critical for *SALS*. This is particularly useful when we have only limited knowledge about the distribution of $\mathcal{X}^{(t)}$.

By comparing *SALS* with *2SGD*, it can be seen that *SALS* is more stable, i.e., it monotonically decreases the loss function in each cycle.

Proposition 4. *For* SALS, *for each* n, *we have*

$$\begin{aligned} &L_\rho(\mathcal{X}^{(t)}; U^{(t+1)}, V^{(t+1)}, W^{(t+1)}) \\ \leq &L_\rho(\mathcal{X}^{(t)}; U^{(t)}, V^{(t)}, W^{(t)}). \end{aligned} \tag{16}$$

Proof. Since $L_\rho(\mathcal{X}^{(t)}; U, V^{(t)}, W^{(t)})$ is convex in U, $U^{(t+1)}$ is a convex combination of $U^{(t)}$ and the least square solution U^*, giving

$$L_\rho(\mathcal{X}^{(t)}; U^{(t+1)}, V^{(t)}, W^{(t)}) \leq L_\rho(\mathcal{X}^{(t)}; U^{(t)}, V^{(t)}, W^{(t)}).$$

By continuing the inequality by the same argument for V and W, we obtain (16). □

We remark that if $\mathcal{X}^{(1)} = \cdots = \mathcal{X}^{(T)}$, i.e., in the non-stochastic case, Proposition 4 shows the convergence of the loss function L_ρ. However, for the stochastic case, there may be no inequality between $L_\rho(\mathcal{X}^{(t+1)}; U^{(t)}, V^{(t)}, W^{(t)})$ and $L_\rho(\mathcal{X}^{(t)}; U^{(t)}, V^{(t)}, W^{(t)})$. Thus we cannot guarantee convergence from this proposition.

Finally, we show the convergence of *SALS* when the regularization parameter ρ is sufficiently large. While such a setting of ρ is generally impractical, we believe that the proposition given below is meaningful as the first step toward the convergence analysis.

Proposition 5. *If* ρ *is sufficiently large,* SALS *converges to the global optimal solution.*

Proof. The Hessian of a regularized loss function is

$$\nabla^2 \mathbb{E}[L_\rho(\mathcal{X}; U, V, W)] = \nabla^2 L(\mathbb{E}[\mathcal{X}]; U, V, W) + \rho I.$$

From Proposition 3, we can assume $(U, V, W) \in D(\rho)$ for some compact set $D(\rho)$, which is a monotone decreasing set with respect to ρ. Let

$$\lambda(\rho) := \min_{(U,V,W) \in D(\rho)} \lambda_{\min}(\nabla^2 L(\mathbb{E}[\mathcal{X}]; U, V, W)),$$

where $\lambda_{\min}$ denotes the minimum eigenvalue. Then we have

$$\nabla^2 \mathbb{E}[L_\rho(\mathcal{X}; U, V, W)] \succeq (\rho + \lambda(\rho))I$$

Because $\lambda(\rho)$ is monotone increasing, for a sufficiently large ρ, the right hand side is positive definite. Thus we can regard the regularized loss function as a strongly convex function.

The convergence of *SALS* now follows by the general theory of stochastic gradient descent method for strongly convex functions (Bottou 2004; Bottou and Le Cun 2005). □

6 Experiments

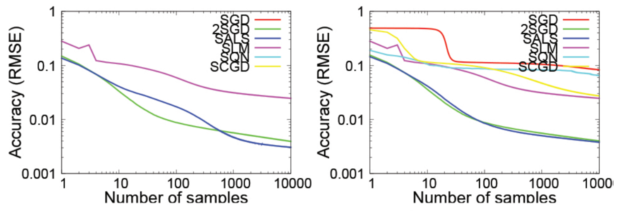
Throughout the experiments, the regularization parameter was fixed as $\rho = 0.0001$. All experiments were conducted using an Intel Xeon E5-2690 2.90GHz CPU with 256GB memory and Ubuntu 12.04. Our algorithm was implemented in C++ and compiled using g++v4.6 with -O3 option.

6.1 Comparison on synthetic data

We compared our three algorithms with three existing ones, which are used for a general stochastic optimization:

1. ***SGD*** (Subsection 4.1).
2. ***2SGD*** (Subsection 4.2).
3. ***SALS*** (Subsection 4.3).
4. ***SLM***: stochastic diagonal Levenberg Marquardt (LeCun et al. 1998).
5. ***SCGD***: stochastic conjugate gradient descent (Schraudolph and Graepel 2002).
6. ***SQN***: stochastic quasi-Newton method (Schraudolph, Yu, and Günter 2007).

All algorithms except SQN had the same complexity, i.e., $O(IR^2 + |\mathcal{X}|R)$ time and $O(IR + |\mathcal{X}|)$ space, where I is the size of tensors, R is the rank to be decomposed, and



(a) default (b) optimized

Figure 1: Comparison of algorithms. The step size is $\eta^{(t)} = 1/(1+t)$ for (a), and $\eta^{(t)} = \lambda_0/(t_0+t)$ for (b), where λ_0 and t_0 are optimized by grid search.

$|\mathcal{X}|$ is the number of nonzero elements of sample $\mathcal{X}$. SQN has complexity $O(IR^2H + |\mathcal{X}|R)$ time and $O(IRH + |\mathcal{X}|)$ space, where H is the history size.

We first generated random matrices $U^* \in \mathbb{R}^{30\times 5}$, $V^* \in \mathbb{R}^{40\times 5}$, and $W^* \in \mathbb{R}^{50\times 5}$ in which entries were drawn from the standard uniform distribution $U(0,1)$. Then we constructed a $30 \times 40 \times 50$ tensor $\mathcal{X} = [\![U^*, V^*, W^*]\!]$ of rank 5 and generate samples $\mathcal{X}^{(t)}$ by $\mathcal{X}^{(t)}_{ijk} = \mathcal{X}_{ijk} + \mathrm{U}(-0.5, 0.5)$, where $\mathrm{U}(-0.5, 0.5)$ denotes the uniform distribution over interval $(-0.5, 0.5)$. We evaluated the performance by the number of samples t versus the root mean square error with respect to $\mathcal{X}$. The results are shown in Figure 1. For Figure 1 (a), we used the fixed step size rule $\eta^{(t)} = 1/(1+t)$ and for Figure 1 (b), we used $\eta^{(t)} = \lambda_0/(t_0+t)$, where parameters λ_0 and t_0 were optimized by a grid search. Note that, we could not identify suitable parameters for SQN in this manner; thus, instead we used $\eta^{(t)} = \lambda_0/((t_0+t)\|\nabla L(\mathcal{X}^{(n)})\|)$ and searched λ_0 and t_0.

The results clearly show the faster convergence and the robustness of *2SGD* and *SALS* with respect to the step size. *SGD*, *SCGD*, and *SQN* diverged (Figure 1 (a)), and thus, these lines are not shown in the figure. To prevent divergence, the step size needed to be selected carefully (Figure 1 (b)). However, the optimization of the step size would be time consuming, and in any case, *2SGD* and *SALS* still outperformed other algorithms.

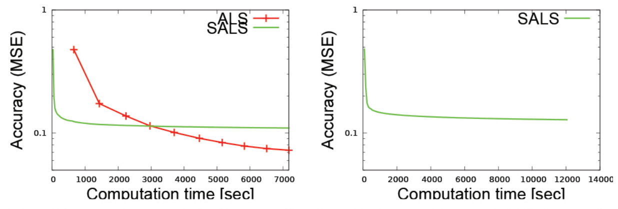
6.2 Scalability for large tensors

To evaluate the scalability, we employed the Amazon review dataset[4] (McAuley and Leskovec 2013), which contains 34 million user reviews. This dataset was originally used to construct a recommender system using matrix factorization, and here we extended the approach to tensor factorization.

We prepared a subset of reviews for T = 10M and 34M and constructed *word-word-word* tensors as data samples, where $\mathcal{X}^{(t)}_{ijk} = 1$ means that the t-th review title contains words i, j, and k simultaneously. For the T = 10M case, the tensor was of size 280K $\times$ 280K $\times$ 280K, and had 500M nonzero elements. For the T = 34M case, it was of size 520K $\times$ 520K $\times$ 520K, and had 1.09G nonzero elements.

We computed a rank 20 expected CP decomposition for these tensors by *SALS*. For efficient computation, we used

[4]http://snap.stanford.edu/data/web-Amazon.html



(a) 10M reviews, 280K words, and 500M nonzeros. (b) 34M reviews, 520K words and 1.09G nonzeros.

Figure 2: Convergence on the Amazon reviews dataset. For the case of 34M reviews, *ALS* did not work do to memory error.

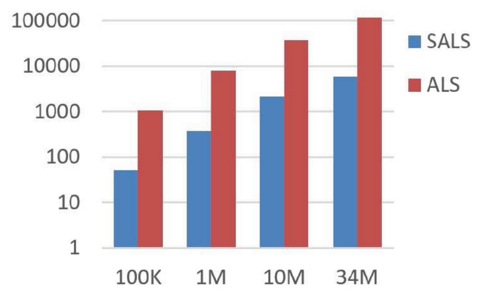


Figure 3: Memory usage on the Amazon review dataset; y-axis denotes the memory used in KB. For the 34M dataset, ALS cannot allocate the memory in our environment.

the mini-batch method, i.e., each sample was a sum of the tensors of 1000 reviews. Note that each tensor $\mathcal{X}^{(t)}$ was rank-one, and therefore, it was more efficiently computable. However, as our purpose was to examine the scalability of algorithms, we did not exploit this low-rank structure. For comparison, we also computed CP decomposition for the sample mean $(1/T)\sum_t \mathcal{X}^{(t)}$ using *ALS*.

The results are shown in Figure 2. *SALS* quickly converged around a near-optimal solution, and then slowly converged to an optimal solution (and loses the lead against *ALS*). This behavior is commonly observed in stochastic gradient methods and theoretically analyzed when a function is strongly convex (Richtárik and Schmidt 2015). The memory usages are shown in Figure 3. In all cases, *SALS* used approximately ten times lower memory than *ALS*. In the 34M case, *ALS* did not work.[5] In contrast, our *SALS* used only 6GB of memory, which can run in a standard computer.

7 Conclusion

This study investigated CP decomposition in which a given tensor is expressed by an expectation. We formalized expected CP decomposition, which is a stochastic optimization problem, and propose three algorithms based on the stochastic gradient descent method. Our *SALS* and *2SGD* algorithms displayed two desirable characteristics: more efficient memory usage and stable behavior of the solution. The

[5]Note that our program allocated additional space for error evaluation. Without this space, ALS used about 118GB of memory.

proposed algorithms outperformed all existing algorithms in terms of accuracy. In addition, *SALS* could decompose the tensor, which had one billion nonzero elements.

References

Acar, E.; Dunlavy, D. M.; and Kolda, T. G. 2011. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics* 25(2):67–86.

Anandkumar, A.; Ge, R.; Hsu, D.; Kakade, S. M.; and Telgarsky, M. 2014. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research* 15(1):2773–2832.

Arora, R.; Cotter, A.; Livescu, K.; and Srebro, N. 2012. Stochastic optimization for PCA and PLS. In *Allerton Conference*, 861–868.

Bahadori, M. T.; Yu, Q. R.; and Liu, Y. 2014. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *Advances in Neural Information Processing Systems 27*, 3491–3499.

Bapat, R. B., and Raghavan, T. E. S. 1997. *Nonnegative matrices and applications*, volume 64. Cambridge University Press.

Bordes, A.; Bottou, L.; Gallinari, P.; Chang, J.; and Smith, S. A. 2010. Erratum: SGDQN is less careful than expected. *Journal of Machine Learning Research* 11:2229–2240.

Bordes, A.; Bottou, L.; and Gallinari, P. 2009. Sgd-qn: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research* 10:1737–1754.

Bottou, L., and Bousquet, O. 2007. The tradeoffs of large scale learning. In *Proceedings of the 21th Annual Conference on Neural Information*, 161–168.

Bottou, L., and Le Cun, Y. 2005. On-line learning for very large data sets. *Applied Stochastic Models in Business and Industry* 21(2):137–151.

Bottou, L. 2004. Stochastic learning. In *Advanced lectures on machine learning*. Springer. 146–168.

Carroll, J. D., and Chang, J.-J. 1970. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika* 35(3):283–319.

Chen, J., and Saad, Y. 2009. On the tensor svd and the optimal low rank orthogonal approximation of tensors. *SIAM Journal on Matrix Analysis and Applications* 30(4):1709–1734.

Comon, P.; Luciani, X.; and De Almeida, A. L. 2009. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics* 23(7-8):393–405.

de Silva, V., and Lim, L.-H. 2008. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications* 30(3):1084–1127.

Fried, D.; Polajnar, T.; and Clark, S. 2015. Low-rank tensors for verbs in compositional distributional semantics. In *Proceedings of the Short Papers of the 53rd Annual Meeting of the Association for Computational Linguistics*, 731–736.

Ge, R.; Huang, F.; Jin, C.; and Yuan, Y. 2015. Escaping from saddle points - online stochastic gradient for tensor decomposition. In *Proceedings of The 28th Conference on Learning Theory*, 797–842.

Harshman, R. A. 1970. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics* 16(1):84.

Hayashi, C., and Hayashi, F. 1982. A new algorithm to solve PARAFAC-model. *Behaviormetrika* 11:49–60.

Hayashi, K.; Maehara, T.; Toyoda, M.; and Kawarabayashi, K. 2015. Real-time top-r topic detection on twitter with topic hijack filtering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Jeon, I.; Papalexakis, E. E.; Kang, U.; and Faloutsos, C. 2015. Haten2: Billion-scale tensor decompositions. In *Proceedings of the 31st IEEE International Conference on Data Engineering*, 1047–1058.

Kang, U.; Papalexakis, E. E.; Harpale, A.; and Faloutsos, C. 2012. Gigatensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge*, 316–324.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *SIAM Review* 51(3):455–500.

Kolda, T. G., and Mayo, J. R. 2011. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications* 32(4):1095–1124.

Koren, Y.; Bell, R. M.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

McAuley, J. J., and Leskovec, J. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, 165–172.

Paatero, P. 2000. Construction and analysis of degenerate parafac models. *Journal of Chemometrics* 14(3):285–299.

Phan, A. H.; Tichavský, P.; and Cichocki, A. 2012. On Fast Computation of Gradients for CANDECOMP/PARAFAC Algorithms. *CoRR* abs/1204.1586.

Phan, A.-H.; Tichavský, P.; and Cichocki, A. 2013. Low complexity damped gauss–newton algorithms for candecomp/parafac. *SIAM Journal on Matrix Analysis and Applications* 34(1):126–147.

Rendle, S., and Thieme, L. S. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the 3rd ACM international conference on Web search and data mining*, 81–90.

Richtárik, P., and Schmidt, M. 2015. Modern convex optimization methods for large-scale empirical risk minimization. Tutorial in ICML'15.

Schraudolph, N. N., and Graepel, T. 2002. Towards stochastic conjugate gradient methods. In *Proceedings of the 9th IEEE International Conference on Neural Information Processing*, 853–856.

Schraudolph, N. N.; Yu, J.; and Günter, S. 2007. A stochastic quasi-Newton method for online convex optimization. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2 of *JMLR Proceedings*, 436–443.