A Parallel Nonnegative Tensor Factorization Algorithm for Mining Global Climate Data

Qiang Zhang[1], Michael W. Berry[2], Brian T. Lamb[2], and Tabitha Samuel[2]

[1] Department of Biostatistical Sciences, Wake Forest University Health Sciences, Medical Center Blvd, Winston Salem, NC 27157
qizhang@wfubmc.edu
[2] Department of Electrical Engineering and Computer Science, University of Tennessee, 203 Claxton Complex, Knoxville, TN 37996-3450
{berry,blamb,tsamuel}@eecs.utk.edu

Abstract. Increasingly large datasets acquired by NASA for global climate studies demand larger computation memory and higher CPU speed to mine out useful and revealing information. While boosting the CPU frequency is getting harder, clustering multiple lower performance computers thus becomes increasingly popular. This prompts a trend of parallelizing the existing algorithms and methods by mathematicians and computer scientists. In this paper, we take on the task of parallelizing the Nonnegative Tensor Factorization (NTF) method, with the purposes of distributing large datasets into each cluster node and thus reducing the demand on a single node, blocking and localizing the computation at the maximal degree, and finally minimizing the memory use for storing matrices or tensors by exploiting their structural relationships. Numerical experiments were performed on a NASA global sea surface temperature dataset and result factors were analyzed and discussed.

Keywords: nonnegative tensor factorization, parallel computation, data mining, global climate.

1 Introduction

Data mining techniques are commonly used for the discovery of *interesting* patterns in earth science data. Such patterns can help to both understand and predict changes in climate and the global carbon cycle. Regions of the earth can be partitioned into land and ocean areas from which subregions described by an ensemble land- or sea-based parameters are possible. Patterns within these subregions are mined to reveal both spatial and temporal autocorrelation. In this study, we sought to identify regions (or clusters) of the earth which have similar short- or long-term characteristics. Earth scientists are particularly interested in patterns that reflect deviations from normal seasonal variations (e.g., El Niño and La Niña). Interpreting these patterns can facilitate a better understanding of biosphere processes and the effects human policy decisions at a global scale. Such effects include deforestation, air and water quality, urbanization, and global warming.

G. Allen et al. (Eds.): ICCS 2009, Part II, LNCS 5545, pp. 405–415, 2009.

Eigensystem-based analysis driven by principal component analysis (PCA) and the singular value decomposition (SVD) has been used to cluster climate indices [14]. Unfortunately, the orthogonal matrix factors (basis vectors) generated by the SVD are difficult to interpret and as discussed by Steinbach et al. in [13], stronger signals typically mask weaker signals. Among other data mining techniques, (approximate) Nonnegative Matrix Factorization (NMF) has attracted much attention since the early work of Paatero and Tapper [11] and Lee and Seung's seminal paper on learning the parts of objects [9]. In NMF, an $m \times n$ (nonnegative) mixed data matrix X is approximately factored into a product of two nonnegative rank-k matrices, with k small compared to m and n, $X \approx WH$. This factorization has the advantage that W and H can provide a physically realizable representation of the mixed data, due to the inherent nonnegativity constraint. Nonnegative Tensor Factorization (NTF) is a natural extension of NMF to higher dimensional data. In NTF, high-dimensional data, such as 3D or 4D global climate data, is factored directly and is approximated by a sum of rank-1 nonnegative tensors. See Figure 1 for an illustration of 3-D tensor factorization. Similar to NMF, we also see a quick development of NTF algorithms [12,15] and their applications in recent years. In this research, we exploit the nonnegative tensor factorization of multidimensional climate data in order to capture patterns/signals not possible with traditional 2-way factor analysis.

2 Parallel Nonnegative Tensor Factorization

In nonnegative tensor factorization (NTF), high-dimensional data, such as global sea surface temperature, is factored directly and is approximated by a sum of rank-1 nonnegative tensors. See Figure 1 for an illustration of a 3-D tensor factorization.

Definition 1. *Let* $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$ *be a nonnegative tensor and define*

$$\hat{\boldsymbol{\mathcal{T}}} = \sum_{i=1}^{k} \boldsymbol{x}^{(i)} \circ \boldsymbol{y}^{(i)} \circ \boldsymbol{z}^{(i)},$$

to be in a CANDECOMP (CP) canonical factored form, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^{D_1}, \boldsymbol{y}^{(i)} \in \mathbb{R}^{D_2}$, *and* $\boldsymbol{z}^{(i)} \in \mathbb{R}^{D_3}$ *are all nonnegative. Then, a rank-k nonnegative approximate tensor factorization problem is defined as*

$$\min_{\tilde{\boldsymbol{\mathcal{T}}}} ||\boldsymbol{\mathcal{T}} - \tilde{\boldsymbol{\mathcal{T}}}||_F^2, \text{ subject to } \tilde{\boldsymbol{\mathcal{T}}} \geq 0. \tag{1}$$

Given the large datasets we encounter with global climate data, our interest in this study is to parallelize the problem posed above and distribute computations evenly to processors in a distributed computing environment. By Definition (1), the NTF problem is posed as a non-linear optimization problem, which is not easily parallelizable. In a naive approach, we may separate the original data cube into 3D blocks and fit factors for each block in parallel. However, the factors from

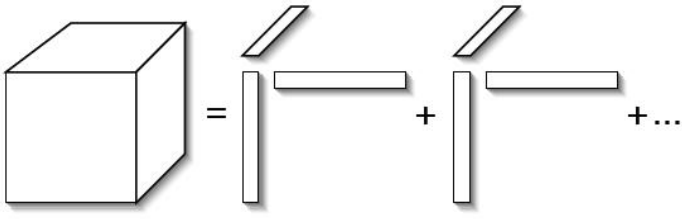

Fig. 1. An illustration of 3-D tensor approximate factorization using a sum of rank one tensors

each block may not match together, which is to say, blocks along each dimension should have an identical factor in either $\boldsymbol{X}$, $\boldsymbol{Y}$ or $\boldsymbol{Z}$. This almost certainly would not be the case when we optimize for three factors individually for each block.

Nevertheless, our hope lies in the fact that a common approach in solving (1) is the Alternating Least Squares (ALS) method [3,6], which is a special case of the block coordinate descent method, also known as the Block Gauss-Seidel (BGS) method [7]. At each iteration step, the BGS method (in alternating fashion) optimizes only a subset of the variables, while keeping the rest fixed, and turns the original non-convex problem into a sequence of convex least squares subproblems. In NTF, this means holding two matrix factors fixed while fitting for the other one. Thus, the original NTF problem is transformed into three semi-NMF (nonnegative matrix factorization) sub-problems in each iteration. Here we use the term "semi" to represent the optimization only for one of the two factor matrices, while assuming the other is given.

Definition 2. *Given* $\boldsymbol{A} \in \mathbb{R}^{m \times n} \geq 0$ *and* $\boldsymbol{W} \in \mathbb{R}^{m \times k} \geq 0$*, a semi-NMF problem is defined as*

$$\min_{\boldsymbol{H}} \Phi(\boldsymbol{H}) = ||\boldsymbol{A} - \boldsymbol{W}\boldsymbol{H}||_F^2, \ \textit{subject to } \boldsymbol{H} \geq 0. \tag{2}$$

One important observation on (2) is that solving (2) is equivalent to solving for each column of $\boldsymbol{H}$ independently, i.e.

$$\min_{\boldsymbol{h}_i} \Phi(\boldsymbol{h}_i) = \|\boldsymbol{a}_i - \boldsymbol{W}\boldsymbol{h}_i\|_F^2, , \tag{3}$$

where $\boldsymbol{a}_i$ and $\boldsymbol{h}_i$ are the column vectors of $\boldsymbol{A}$ and $\boldsymbol{H}$. This provides a great opportunity for parallelization of each semi-NMF subproblem, even though the original NTF problem is not defined for easy parallelization.

The ALS approach splits the NTF problem (1) into three semi-NMF subproblems, i.e. given $\boldsymbol{X}$ and $\boldsymbol{Y}$, we solve for $\boldsymbol{Z}$ by

$$\min_{\boldsymbol{Z}} \Phi(\boldsymbol{Z}) = \|\boldsymbol{T}_z - (\boldsymbol{X} \odot \boldsymbol{Y})\boldsymbol{Z}\|_F^2, \tag{4}$$

where $\boldsymbol{T}_z \in \mathbb{R}^{D_1 D_2 \times D_3}$ is the unfolded tensor $\boldsymbol{\mathcal{T}}$ along the z dimension and $(\boldsymbol{T}_z)_{(j-1)*D_2+i,k} = t_{ijk}$. $\boldsymbol{X} \odot \boldsymbol{Y}$ is the Khatri-Rao product of the two matrices. Next we fix $\boldsymbol{X}$ and $\boldsymbol{Z}$, and solve for $\boldsymbol{Y}$ by

$$\min_{\boldsymbol{Y}} \Phi(\boldsymbol{Y}) = \|\boldsymbol{T}_y - (\boldsymbol{X} \odot \boldsymbol{Z})\boldsymbol{Y}\|_F^2, \tag{5}$$

where $\boldsymbol{T}_y \in \mathbb{R}^{D_1 D_3 \times D_2}$ is the unfolded tensor $\boldsymbol{\mathcal{T}}$ along the y dimension and $(\boldsymbol{T}_y)_{(k-1)*D_3+i,j} = t_{ijk}$. Finally, we fix $\boldsymbol{Z}$ and $\boldsymbol{Y}$, and solve for $\boldsymbol{X}$ by

$$\min_{\boldsymbol{X}} \Phi(\boldsymbol{X}) = \|\boldsymbol{T}_x - (\boldsymbol{Z} \odot \boldsymbol{Y})\boldsymbol{X}\|_F^2, \tag{6}$$

where $\boldsymbol{T}_x \in \mathbb{R}^{D_2 D_3 \times D_1}$ is the unfolded tensor $\boldsymbol{\mathcal{T}}$ along the x dimension and $(\boldsymbol{T}_x)_{(k-1)*D_3+j,i} = t_{ijk}$.

Here we use a modified version of a Projected Gradient Descent (PGD) method developed by Lin [10] to solve the semi-NMF problem (2). The projected gradient descent method is basically adding a projection function to the regular gradient descent method.

$$\boldsymbol{H}^{(p+1)} = P_+[\boldsymbol{H}^{(p)} - \alpha_p \nabla \Phi(\boldsymbol{H}^{(p)})], \tag{7}$$

where the gradient is $\nabla \Phi(\boldsymbol{H}) = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H} - \boldsymbol{W}^T\boldsymbol{A}$, and P_+ is the projection function onto the nonnegative domain. Lin [10] enhanced the performance of the PGD method by improving the search for the optimal step size using the Armijo rule.

Two observations can be made about the PGD method. First, to solve for $\boldsymbol{H}$, we only need to use two quadratic forms of $\boldsymbol{W}$ and $\boldsymbol{A}$, i.e. $\boldsymbol{W}^T\boldsymbol{W}$ and $\boldsymbol{W}^T\boldsymbol{A}$ and by comparing the sizes of two quadratic forms, i.e. $k \times k$ and $k \times n$, with the sizes of $\boldsymbol{W}$ and $\boldsymbol{A}$, i.e. $m \times k$ and $m \times n$, and knowing $m, n \gg k$, we can save considerable memory required to store these matrices. Second, we can also split $\boldsymbol{W}$ and $\boldsymbol{A}$ and compute $\boldsymbol{W}^T\boldsymbol{W}$ and $\boldsymbol{W}^T\boldsymbol{A}$ in parallel, i.e.

$$\boldsymbol{W}^T\boldsymbol{W} = \sum_{i=1}^{p} \boldsymbol{W}_i^T\boldsymbol{W}_i \text{ and } \boldsymbol{W}^T\boldsymbol{A} = \sum_{i=1}^{p} \boldsymbol{W}_i^T\boldsymbol{A}_i.$$

Here, $W_i \in \mathbb{R}^{m/p \times k}$ is a block sub-matrix of $\boldsymbol{W}$, $\boldsymbol{A}_i \in \mathbb{R}^{m/p \times n}$ is a block sub-matrix of $\boldsymbol{A}$, p is the number of processors, and $i = 1, \ldots, p$.

2.1 Distribution of Data

We distribute four matrices to independent processors in the following ways to facilitate parallel I/O and computation. We divide $\boldsymbol{T}_z$ by row blocks, $\boldsymbol{T}_{zi}$, each block having a size of $D_1 D_2/p \times D_3$. This allows for parallel loading of data. One important observation is that we do not need to save $\boldsymbol{T}_y$ and $\boldsymbol{T}_x$ in the memory due to their relationships with $\boldsymbol{T}_z$, stated in the following proposition (proof is straightforward and thus omitted). These relationships will be used in computing the quadratic forms.

Proposition 1. *The relationships between $\boldsymbol{T}_x$, $\boldsymbol{T}_y$ and $\boldsymbol{T}_z$ are:*

1. *Each column of $\boldsymbol{T}_y$ is a vectorized row block matrix of $\boldsymbol{T}_z$.*

2. *Each row block matrix of* $\boldsymbol{T}_x$ *is a transpose of the corresponding row block matrix of* $\boldsymbol{T}_z$.

We next divide $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ by row blocks, each block having a size of $D_i/p \times k, i = 1, 2, 3$. This allows for parallel initialization and writing of these matrices to output. Note that if D_1, D_2 or D_3 cannot be divided by p, the last block will have a remainder number of rows.

2.2 Parallelization of the First Semi-NMF (4)

For convenience, we represent $\boldsymbol{X} \odot \boldsymbol{Y}$ by $\boldsymbol{W}$. We first collect $\boldsymbol{X}_i$ from each process(or) to form a full $\boldsymbol{X}$, and compute the local $\boldsymbol{W}_i = \boldsymbol{X} \odot \boldsymbol{Y}_i$. We then locally compute $\boldsymbol{W}_i^T \boldsymbol{W}_i$ and $\boldsymbol{W}_i^T \boldsymbol{T}_{zi}$ and compute their sums ($\boldsymbol{W}^T \boldsymbol{W}$ and $\boldsymbol{W}^T \boldsymbol{T}_z$, respectively) using the DSESUM2D subroutine provided by BLACS [4]. $\boldsymbol{W}^T \boldsymbol{T}_z$ is then partitioned into column blocks of a size $k \times D_3/p$ for input into the PGD subroutine. Thus, instances of the PGD subroutine run in parallel to solve for each block $\boldsymbol{Z}_i$. This process is illustrated in Figure 2.

2.3 Parallelization of the Second Semi-NMF (5)

Here, we represent $\boldsymbol{X} \odot \boldsymbol{Z}$ by $\boldsymbol{W}$. We first collect $\boldsymbol{Z}_i$ from each process(or) to form a complete $\boldsymbol{Z}$, and since we already have the complete $\boldsymbol{X}$ within each process(or), we can compute the complete $\boldsymbol{W} = \boldsymbol{X} \odot \boldsymbol{Z}$, and thereby compute $\boldsymbol{W}^T \boldsymbol{W}$. To compute $\boldsymbol{W}^T \boldsymbol{T}_{yi}$, notice that each column of $\boldsymbol{T}_y$ is a row block of $\boldsymbol{T}_z$,

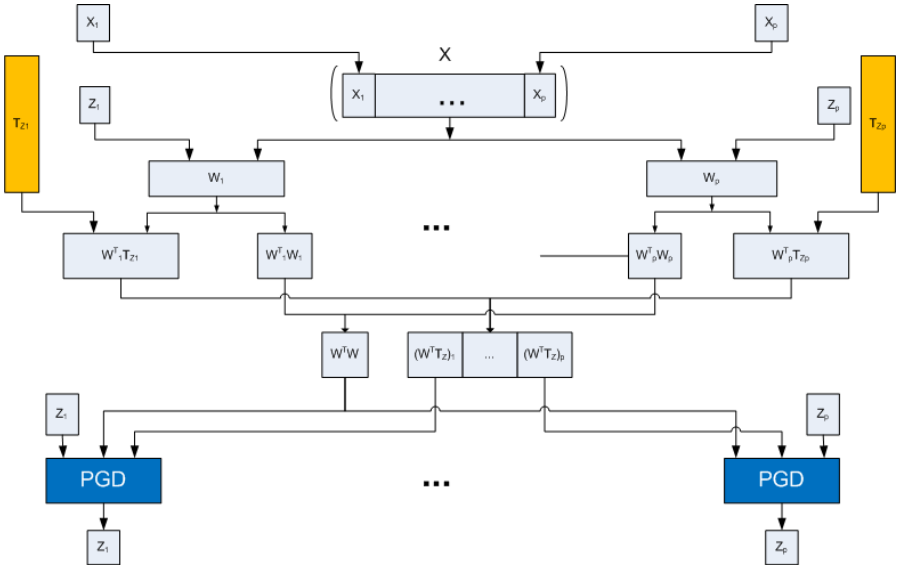

Fig. 2. Flowchart for the first semi-NMF subproblem (4) within NTF

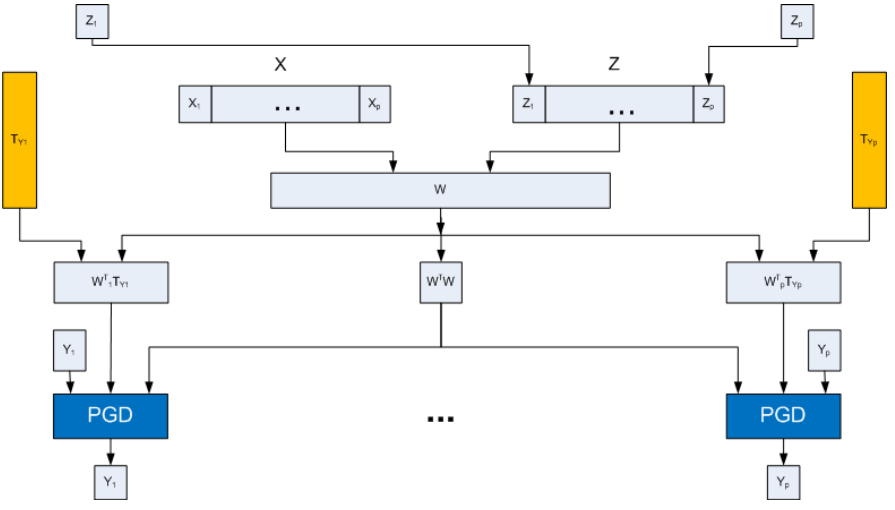

Fig. 3. Flowchart for the second semi-NMF subproblem (5) within NTF

and thus we can avoid saving $\boldsymbol{T}_y$ in memory and use the vectorized blocks of the local $\boldsymbol{T}_z$ to multiply with $\boldsymbol{W}$. Because $\boldsymbol{W}^T\boldsymbol{T}_{yi}$ is locally computed, it can used by a call to the PGD subroutine. Thus, independent calls to the PGD subroutine solve for each block $\boldsymbol{Y}_i$ in parallel. This process is illustrated in Figure 3.

2.4 Parallelization of the Third Semi-NMF (6)

Here, we represent $\boldsymbol{Z} \odot \boldsymbol{Y}$ by $\boldsymbol{W}$. We use the already collected complete $\boldsymbol{Z}$ and compute $\boldsymbol{Y}_i$ in order to formulate $\boldsymbol{W}_i = \boldsymbol{Z} \odot \boldsymbol{Y}_i$, which would then be used to compute $\boldsymbol{W}^T\boldsymbol{W}$. To compute $\boldsymbol{W}_i^T\boldsymbol{T}_{xi}$, notice that each row block of $\boldsymbol{T}_x$ is the transpose of the corresponding row block of $\boldsymbol{T}_z$, and thus we can avoid saving $\boldsymbol{T}_x$ in memory and use the row blocks of local $\boldsymbol{T}_z$ to multiply with $\boldsymbol{W}_i$. We deploy the LAPACK [1] subroutine DGEMM to avoid transposing $\boldsymbol{T}_{zi}$. To sum up $\boldsymbol{W}_i^T\boldsymbol{W}_i$ and $\boldsymbol{W}^T\boldsymbol{T}_{xi}$, we divide $\boldsymbol{W}^T\boldsymbol{T}_x$ into column blocks of a size $k \times D_1/p$ and make separate calls to the PGD subroutine. Again, these calls to PGD execute in parallel to solve for each block $\boldsymbol{X}_i$. The flowchart for this process is very similar to the one in Figure 2.

3 Data and Experimental Results

The six climate-based indices used for this study (see Table 1) were provided by researchers at the NASA Ames Research Center (ARC) in Moffett Field, CA. Pre-processing was performed to guarantee that the six variables matched the same coordinate system and time span. Most of the values are interpolated

Table 1. Climate variables considered in this study with adjustments (or shifts) to enforce nonnegativity (if needed)

Name	Description	Adjustment
sst	sea surface temperature	+273.15
ndvi	normalized difference vegetation index	+0.2
tem	land surface temperature	+273.15
pre	precipitation	
hg500	geopotential height (elevation) for barometric pressure of 500 millibars	+300
hg1000	geopotential height (elevation) for barometric pressure of 1000 millibars	+300

monthly averages on a uniform grid (with a slight distortion at the poles). Interpolation for some of the variables (such as geopotential height) necessarily produced negative values in some of the extreme regions (where it is difficult to sample or when surface pressure[1] is below 1000 mbar). It is not uncommon for many of the array values to be interpolated due to the sparsity of the original samples. The Arctic region, in particular, has few weather stations so that data values for many of the corresponding (lattitude, longitude) coordinates are interpolated from readings taken hundreds of miles away. Simple shifts (scalar increments) to these interpolated values are applied to all negative array elements to insure that all NTF input arrays are nonnegative.

Each parameter (from Table 1) is defined by a 3-way array or datacube of dimension $720 \times 360 \times 252$. The first two dimensions correspond to longitude and latitude coordinates, respectively, and the third dimension represents the month of reading. The time dimension spans from January 1982 to December 2002 for a total of 252 months (i.e., 21 years).

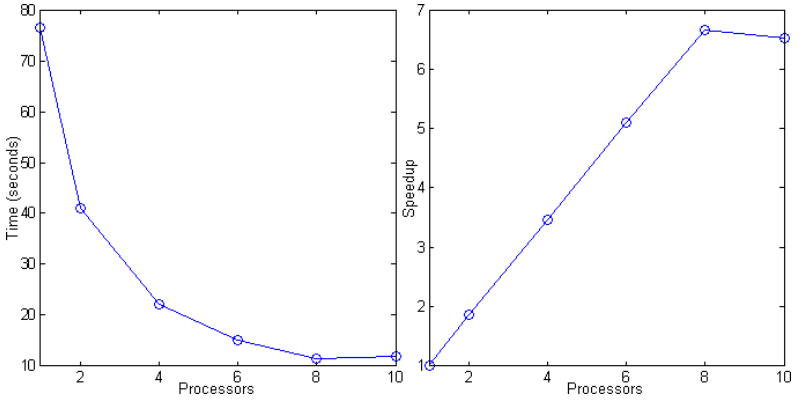

Fig. 4. Computation speedup of the parallel NTF algorithm

[1] Such was the case for the New Orleans area during hurricane Katrina.

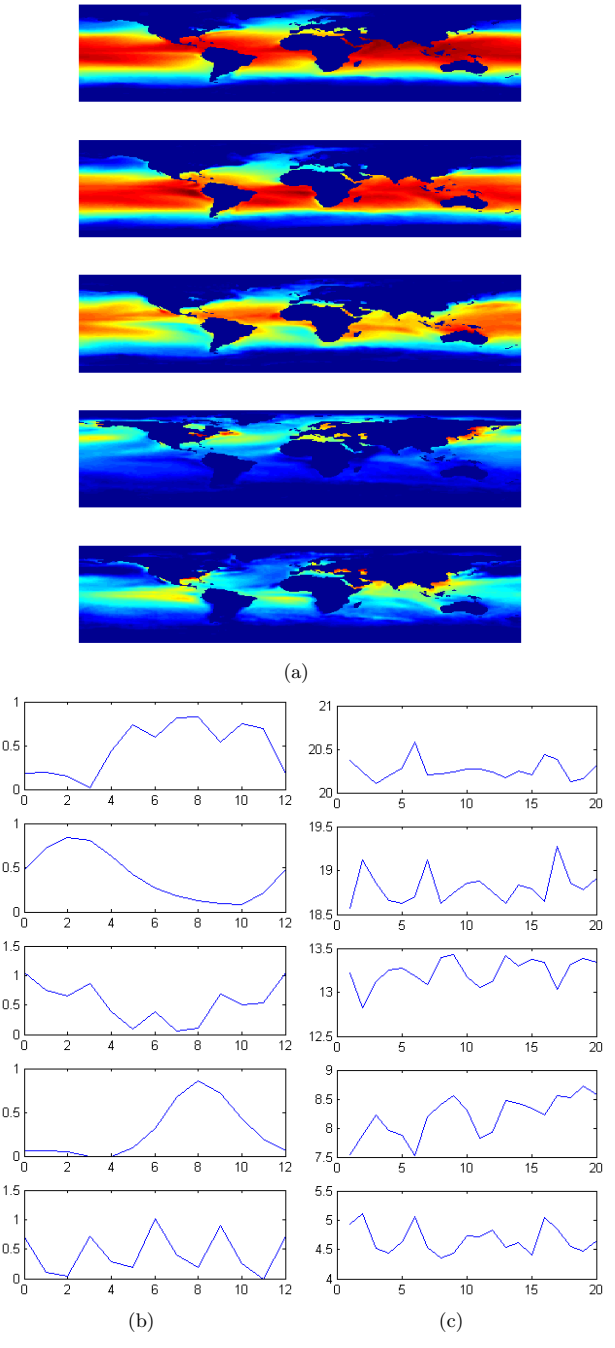

Fig. 5. (a) Global map of sea surface temperature patterns, (b) monthly variations of sea surface temperature patterns, (c) yearly variations of sea surface temperature patterns

All computations were performed on a Sun Fire X4600 M2 with 16 AMD 2.8GHz cores and 32GB of RAM. The original MATLAB®NTF codes were rewritten in C++ and compiled with several libraries including LAPACK [1], ScaLAPACK [2], BLACS [4], BLAS [5] and MPICH [8].

3.1 Speedup of the Parallel Computation

We use a simulated datacube in the size of $600 \times 400 \times 200$ to evaluate the speedup of the parallel NTF computation, by setting the number of processors at 1, 2, 4, 8 and 10. The size was deliberately chosen to be multiples of the number of processors to avoid any inconvenience in data distribution. The leftmost graph of Figure 4 shows the total computation time used for 100 iterations of the parallel NTF algorithm, taken from the processor that finishes last. The rightmost graph of Figure 4 shows the corresponding speedup. A sublinear speedup was achieved for 2 to 8 processors with an approximate peak speedup of 6.8 (among all runs with up to 10 processors).

3.2 Clustering Global Climate Data

The sea surface temperature parameter, originally in MATLAB®format, was partitioned into four sections and written in ASCII format for parallel reads by four different processes. The original data cube was first reshaped into a $259200 \times 12 \times 21$ array, and after removing sections corresponding to land-based coordinates the resulting data cube was $176876 \times 12 \times 20$. We also removed the last year data of data to make the time dimension a multiple of 4 for convenience.

Our intent was to extract typical monthly variation patterns in the second (tensor) factor, typical yearly variation patterns in the third factor and their corresponding global maps in the first factor. All three factors are shown in Figure 5, and they are sorted by the norm of the CP tensor from the greatest to the smallest in order to rank significance. We note that in Figure 5.b, the results in the last month are replicated at month 0 to reflect a full cycle.

The second factor represents El Niño, which has a characteristic peak in the winter and its global map shows a dark red tongue-shaped area off the coast of Ecuador. A yearly warming trend is observed in the fourth factor, mostly in the northern hemisphere and around the northern Pacific coastal area of China and Russia, and also to the northern Atlantic coastal area of the United States and Canada. It is also of interest to note that the last pattern shows some clear seasonal variations mostly along coastal areas (see the isolated red regions).

4 Conclusions

In this study, the nonnegative tensor factorization (NTF) method as a data mining tool is parallelized with the purpose of efficiently processing large datasets encountered in earth science. The parallel algorithm exploits the structural

relationships between matrices used in the original NTF algorithm for data distribution, memory savings and even computation task distribution. It is specifically applied to NASA-provided global land- and sea-based climate data with interesting results presented and analyzed for global sea surface temperature, in particular. Although not reported in this work, additional parallel NTF experiments using different combinations of the climate variables listed in Table 1 have been conducted. We expect to report on the results of clustering multiple land- and sea-based parameters in the near future.

Acknowledgement

This research was sponsored in part by the National Aeronautics and Space Administration (NASA) Ames Research Center under contract No. 07024004.

References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
2. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. SIAM, Philadelphia (1997)
3. Cichocki, A., Zdunek, R., Amari, S.: Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization. In: Davies, M.E., James, C.J., Abdallah, S.A., Plumbley, M.D. (eds.) ICA 2007. LNCS, vol. 4666, pp. 169–176. Springer, Heidelberg (2007)
4. Dongarra, J., Whaley, R.C.: LAPACK Working Note 94: A User"s Guide to the BLACS v1.0, Technical Report: UT-CS-95-281 (1995)
5. Dongarra, J., Du Croz, J., Duff, I.S., Hammarling, S.: A set of Level 3 Basic Linear Algebra Subprograms. ACM Trans. Math. Soft. 16, 1–17 (1990)
6. Faber, N.K.M., Bro, R., Hopke, P.K.: Recent developments in CANDECOMP/PARAFAC algorithms: a critical review. Chemometr. Intell. Lab. 65, 119–137 (2003)
7. Grippo, L., Sciandrone, M.: On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. Operations Research Letters 26, 127–136 (2000)
8. Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press, Cambridge (1994)
9. Lee, D., Seung, H.: Learning the Parts of Objects by Non-Negative Matrix Factorization. Nature 401, 788–791 (1999)
10. Lin, C.: Projected gradient methods for non-negative matrix factorization. Neural Computation 19, 2756–2779 (2007)
11. Paatero, P., Tapper, U.: Positive matrix factorization a nonnegative factor model with optimal utilization of error-estimates of data value. Environmetrics 5, 111–126 (1994)
12. Shashua, A., Hazan, T.: Non-negative tensor factorization with applications to statistics and computer vision. In: Proceedings of the 22^{nd} International Conference on Machine Learning, pp. 792–799 (2005)

13. Steinbach, M., Tan, P.-N., Kumar, V., Klooster, S., Potter, C.: Discovery of climate indices using clustering. In: Proceedings of the Ninth ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2003), Washington, DC, August 24-27, pp. 446–455 (2003)
14. Storch, H.V., Zwiers, F.W.: Statistical Analysis in Climate Research. Cambridge University Press, Cambridge (1999)
15. Zhang, Q., Wang, H., Plemmons, R., Pauca, P.: Tensor methods for hyperspectral data analysis: a space object material identification study. Journal of Optical Society of America A 12, 3001–3012 (2008)