IEICE TRANS. INF. & SYST., VOL.E95–D, NO.2 FEBRUARY 2012

PAPER

Global Mapping Analysis: Stochastic Gradient Algorithm in Multidimensional Scaling

Yoshitatsu MATSUDA[†a)], ***Nonmember and*** **Kazunori YAMAGUCHI**[††b)], ***Member***

SUMMARY In order to implement multidimensional scaling (MDS) efficiently, we propose a new method named "global mapping analysis" (GMA), which applies stochastic approximation to minimizing MDS criteria. GMA can solve MDS more efficiently in both the linear case (classical MDS) and non-linear one (e.g., ALSCAL) if only the MDS criteria are polynomial. GMA separates the polynomial criteria into the local factors and the global ones. Because the global factors need to be calculated only once in each iteration, GMA is of linear order in the number of objects. Numerical experiments on artificial data verify the efficiency of GMA. It is also shown that GMA can find out various interesting structures from massive document collections.

key words: *multidimensional scaling, stochastic gradient algorithm, text mining*

1. Introduction

Multidimensional scaling (MDS) is one of the well-known methods in multivariate data analysis, which is widely used for dimension reduction [1], [2]. It finds a mapping of objects in a low-dimensional space, which preserves the original disparities among all the objects in a high-dimensional space as "faithfully" as possible. Generally, MDS is carried out by minimizing a criterion (a cost function) which evaluates the "faithfulness." The criterion is often called *stress* (for example, the classical MDS stress [3], the Sammon's stress [4], and SSTRESS [5]). Since these criteria are based on pairwise disparities among all the objects, the computational cost of MDS increases at least quadratically according to the number of objects. Such cost is too high for large-scale problems. Because the classical MDS is equivalent to a matrix decomposition, there are some efficient algorithms such as power methods. However, because it needs the disparity matrix in advance, the essential computational cost increases quadratically. Moreover, it can not utilize non-linear criteria. Regarding non-linear dimension reduction, artificial neural network approaches such as the self-organizing map (SOM) [6], [7] have been known to be efficient. In the SOM [7], the objects are clustered and the mapping is formed among the clusters. Though the computational cost is reduced drastically, it still increases quadratically. Moreover, the relations among the objects in the same clusters are lost. In this paper, we propose a new stochastic gradient algorithm named "global mapping analysis" (GMA), which solves both the classical linear MDS [8], [9] and a non-linear MDS known as ALSCAL [5] efficiently. By using stochastic approximation [10], the computational cost of each update in the gradient descent is linear to the number of objects. So, GMA is expected to be efficient if the number of objects is large. This paper is an extension of [11] and [12] with elaboration of the algorithm and many additional results.

This paper is organized as follows. In Sect. 2, GMA is explained in both linear and non-linear cases. In Sect. 2.1, the linear GMA is described. In Sect. 2.2, the non-linear GMA is derived. Their relations to other methods are discussed in Sect. 2.3. In Sect. 3, experiments on artificial data verify that GMA is quite efficient. In Sect. 4, GMA and the SOM [7] are applied to massive document collections (the articles posted to a Usenet newsgroup and those to the whole Usenet) and the results are examined. Lastly, Sect. 5 concludes this paper.

2. Global Mapping Analysis

2.1 Linear GMA

Let $\boldsymbol{X} = (x_{ik})$ be a given $N \times M$ data matrix in the (M-dimensional) original space, where N is the number of objects and x_{ik} indicates the k-th coordinate of the i-th object. Moreover, let $\boldsymbol{Y} = (y_{il})$ be a mapping of objects in the L-dimensional ($L \ll M$) space. Then, the stress for the classical MDS [9] (denoted as CSTRESS in this paper) is given as follows (see [2], [3] and [11]):

$$\text{CSTRESS} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(b_{ij} - b_{ij}^{*} \right)^2, \tag{1}$$

where

$$b_{ij} = \sum_{l=1}^{L} y_{il} y_{jl}, \tag{2}$$

and

$$b_{ij}^{*} = \sum_{k=1}^{M} \Big(x_{ik} - \text{mean}_i \left(x_{ik} \right) \Big) \Big(x_{jk} - \text{mean}_i \left(x_{ik} \right) \Big). \tag{3}$$

Here, $\text{mean}_i \left(x_{ik} \right) = \frac{\sum_{i=1}^{N} x_{ik}}{N}$. By applying $x_{ik} := x_{ik} -$

Manuscript received May 20, 2010.
Manuscript revised April 19, 2011.

[†]The author is with Aoyama Gakuin University, Sagamihara-shi, 252–5258 Japan.
[††]The author is with The University of Tokyo, Tokyo, 153–8902 Japan.
a) E-mail: matsuda@it.aoyama.ac.jp
b) E-mail: yamaguch@graco.c.u-tokyo.ac.jp

DOI: 10.1587/transinf.E95.D.596

$\text{mean}_i(x_{ik})$, CSTRESS is simplified into

$$\text{CSTRESS} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(\sum_{l=1}^{L} y_{il}y_{jl} - \sum_{k=1}^{M} x_{ik}x_{jk}\right)^2. \tag{4}$$

Usually, the singular value decomposition (SVD) of the matrix $\boldsymbol{B}^* = \left(b^*_{ij}\right)$ is used for minimizing CSTRESS [2]. Though there are some efficient techniques for SVD such as the power iteration [13], the cost of calculating $\boldsymbol{B}^*$ is $O\left(N^2\right)$. In the following, an efficient algorithm minimizing CSTRESS is derived by using stochastic approximation. First, the simple gradient with respect to each y_{il} is given as

$$\frac{\partial \text{CSTRESS}}{\partial y_{il}} = \frac{4}{M}\sum_{k=1}^{M}\sum_{j=1}^{N}\left(\sum_{m=1}^{L} y_{im}y_{jm} - Mx_{ik}x_{jk}\right)y_{jl}. \tag{5}$$

Now, we interpret the operator $\frac{\sum_k}{M}$ as the expectation operator over k if k is given as a random integer which is distributed uniformly from 1 to M. Therefore, stochastic approximation [10] is applicable to the optimization of CSTRESS. Consequently, CSTRESS is minimized by repeating the following update equation:

$$y_{il} := y_{il} - c(T)\sum_{j=1}^{N}\left(\sum_{m=1}^{L} y_{im}y_{jm} - Mx_{ip}x_{jp}\right)y_{jl}, \tag{6}$$

where T means the time step, $c(T)$ is the stepsize which has to satisfy $\lim_{T\to\infty} c(T) = 0$ and $\sum_{T=1}^{\infty} c(T) = \infty$, and p is a random integer between 1 and M which is renewed at each time step T. Lastly, the update equation of GMA is given as

$$y_{il} := y_{il} - c(T)\left(\sum_{m=1}^{L} y_{im}\alpha_{lm} - Mx_{ip}\beta_{lp}\right), \tag{7}$$

where

$$\alpha_{lm} = \sum_{i=1}^{N} y_{il}y_{im} \text{ and } \beta_{lp} = \sum_{i=1}^{N} x_{ip}y_{il}. \tag{8}$$

Equation (7) is the update equation in the standard algorithm of GMA, where Eq. (7) is preceded by the calculation of Eq. (8). Note that α_{lm} and β_{lp} (which are called "global factors" in this paper) are "separated" from the calculation of Eq. (7). In other words, the global factors can be calculated independently of Eq. (7). The computational cost of all α_{lm} and β_{lp} is $O\left(NL^2\right)$. Then, the cost for the update of $\boldsymbol{Y}$ is $O\left(NL^2\right)$ because that for each y_{il} by Eq. (7) is only $O(L)$. Consequently, the total cost at each time step is $O\left(NL^2\right)$. This "separation" technique is crucial to the efficiency of GMA. Note that this "separation" is not applicable to the usual gradient descent where $\boldsymbol{B}^*$ is calculated in advance.

$\boldsymbol{Y}$ is given randomly at the initial step, which satisfies $\sum_{l=1}^{L} y_{il} = 0$ for each i and $\sum_{i=1}^{N}\sum_{k=1}^{M} x_{ik}^2 = \sum_{i=1}^{N}\sum_{l=1}^{L} y_{il}^2$. The appropriate determination of the stepsize $c(T)$ is an important problem. In this paper, the following heuristic is used:

$$c(T) = 0.5 \times \frac{50}{T+50}c_{init} \tag{9}$$

where c_{init} is given as

$$c_{init} = \sqrt{\frac{M\sum_{i=1}^{N}\sum_{l=1}^{L}\left(y_{il} - \frac{\sum_{m=1}^{L} y_{im}}{L}\right)^2}{\sum_{p=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{L}\left(\partial C(i,l,p)\right)^2}} \tag{10}$$

where $\partial C(i,l,p)$ is the estimation of the gradient of CSTRESS w.r.t. y_{il} for p in Eq. (7):

$$\partial C(i,l,p) = \sum_{m=1}^{L} y_{im}\alpha_{lm} - Mx_{ip}\beta_{lp}. \tag{11}$$

The computational cost of c_{init} by Eqs. (10) and (11) is $O\left(NML^2\right)$. c_{init} roughly matches the "total quantity" of the updates for $\boldsymbol{Y}$ (which are proportional to $\partial C(i,l,p)$) to that of $\boldsymbol{Y}$ itself at each time step. If $\boldsymbol{Y}$ diverges to infinity, c_{init} is halved and the learning process is restarted. Thus, c_{init} is set to a suitable value which achieves both fast convergence and global adjustments. Because $c(T)$ in Eq. (9) satisfies the necessary conditions for stochastic approximation, $\boldsymbol{Y}$ is guaranteed to converge to a local optimum. In addition, because $c(T)$ keeps a relatively large value at least for the first 50 time steps, $\boldsymbol{Y}$ is expected to be close to the global optimum.

2.2 Non-linear GMA

As shown in Sect. 2.1, the crucial point in GMA is the "separation" technique in the update equation. This separation technique is always available if the pairwise disparities in the MDS stress are estimated by some polynomial functions. In this section, the separation technique is applied to one of the well-known MDS named ALSCAL [5].

In ALSCAL, the following stress (named SSTRESS) is given as

$$\text{SSTRESS} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(\text{dist}_y(i,j) - \text{dist}_x(i,j)\right)^2 \tag{12}$$

where $\text{dist}_y(i,j) = \sum_{l=1}^{L}\left(y_{il} - y_{jl}\right)^2$ and $\text{dist}_x(i,j) = \sum_{k=1}^{M}\left(x_{ik} - x_{jk}\right)^2$. Then, the gradient w.r.t. y_{il} is given as

$$\frac{\partial \text{SSTRESS}}{\partial y_{il}} = 4\sum_{l=1}^{L}\sum_{j=1}^{N}\left(y_{il} - y_{jl}\right)^2\left(y_{il} - y_{jl}\right) - 4\sum_{k=1}^{M}\sum_{j=1}^{N}\left(x_{ik} - x_{jk}\right)^2\left(y_{il} - y_{jl}\right). \tag{13}$$

In the same way as in Sect. 2.1, the following update equation is derived:

$$
\begin{aligned}
y_{il} := y_{il} & \\
& - c(T) \sum_{m=1}^{L} \left(N y_{il} y_{im}^2 - 2 y_{il} y_{im} \gamma_m + y_{il} \alpha_{mm} \right) \\
& + c(T) \sum_{m=1}^{L} \left(y_{im}^2 \gamma_l - 2 y_{im} \alpha_{lm} + \delta_{lm} \right) \\
& + c(T) M \left(N x_{ip}^2 y_{il} - 2 x_{ip} y_{il} \epsilon_p + y_{il} \zeta_p \right) \\
& - c(T) M \left(x_{ip}^2 \gamma_l - 2 x_{ip} \beta_{lp} + \eta_{lp} \right), \qquad (14)
\end{aligned}
$$

where p, T, $c(T)$, α_{lm}, and β_{lp} are given in Sect. 2.1,

$$
\gamma_l = \sum_{i=1}^{N} y_{il}, \ \delta_{lm} = \sum_{i=1}^{N} y_{il} y_{im}^2, \ \epsilon_p = \sum_{i=1}^{N} x_{ip}, \qquad (15)
$$

$$
\zeta_p = \sum_{i=1}^{N} x_{ip}^2, \text{ and } \eta_{lp} = \sum_{i=1}^{N} x_{ip}^2 y_{il}. \qquad (16)
$$

By calculating the global factors α-η in advance, the total cost for the update of $\boldsymbol{Y}$ at each time step is only $O\left(NL^2\right)$.

To avoid converging to local minima, the result of the linear GMA (Sect. 2.1) is used as $\boldsymbol{Y}(0)$ of the non-linear GMA. Regarding the stepsize, Eq. (9) is used by replacing $\partial C(i,l,p)$ of Eq. (10) with $\partial S(i,l,p)$:

$$
\begin{aligned}
& \partial S(i,l,p) \\
& = - \sum_{m=1}^{L} \left(N y_{il} y_{im}^2 - 2 y_{il} y_{im} \gamma_m + y_{il} \alpha_{mm} \right) \\
& \quad + \sum_{m=1}^{L} \left(y_{im}^2 \gamma_l - 2 y_{im} \alpha_{lm} + \delta_{lm} \right) \\
& \quad + M \left(N x_{ip}^2 y_{il} - 2 x_{ip} y_{il} \epsilon_p + y_{il} \zeta_p \right) \\
& \quad - M \left(x_{ip}^2 \gamma_l - 2 x_{ip} \beta_{lp} + \eta_{lp} \right). \qquad (17)
\end{aligned}
$$

2.3 Relation to Other Methods

Other MDS methods: Though many other algorithms have been proposed in MDS [2], almost all of them need the calculation of the $N \times N$ disparity matrix. The most significant advantage of GMA is that it does not have to calculate the matrix. Its update cost at each step is linear to N. Therefore, GMA can solve dimension reduction problems of quite large N, which are intractable in the other methods.

Oja's PCA network rule: PCA (principal component analysis) is almost equivalent to the classical MDS with the squared Euclidean distance metric [14]. Consequently, the linear GMA can be easily extended to PCA by $x_{ik} := x_{ik} - \text{mean}_k(x_{ik})$ where $\text{mean}_k(x_{ik}) = \frac{\sum_{k=1}^{M} x_{ik}}{M}$. The data matrix $\boldsymbol{X}$ is centered along each column for MDS (by $\text{mean}_i(x_{ik})$) and along each row for PCA (by $\text{mean}_k(x_{ik})$). Moreover, it can be proven that the linear GMA (Eq. (7)) is equivalent to Oja's symmetrical PCA network rule [15], [16] in the following. The average of Eq. (7) over p is given as follows by the matrix notation:

$$
\boldsymbol{Y} := \boldsymbol{Y} + c(T) \left(\boldsymbol{B}^* \boldsymbol{Y} - \boldsymbol{Y} \boldsymbol{Y}^t \boldsymbol{Y} \right). \qquad (18)
$$

Because $\boldsymbol{B}^*$ is positive semi-definite, symmetric, and regular (by taking inevitable small noise into account), there exists $\boldsymbol{B}^{*-\frac{1}{2}}$. Therefore, by letting $\boldsymbol{Z}$ be $\boldsymbol{B}^{*-\frac{1}{2}} \boldsymbol{Y}$, Eq. (18) is rewritten as

$$
\boldsymbol{Z} := \boldsymbol{Z} + c(T) \left(\boldsymbol{B}^* \boldsymbol{Z} - \boldsymbol{Z} \boldsymbol{Z}^t \boldsymbol{B}^* \boldsymbol{Z} \right), \qquad (19)
$$

which is completely equivalent to Oja's symmetrical PCA network rule. This equivalence has been proven in [17] in a different context where they derived the cost function of Oja's rule. So, the linear GMA inherits the properties of Oja's rule. For example, GMA always converges to the global optimum like Oja's rule does under some weak conditions. The one difference between GMA and Oja's rule is that GMA solves MDS directly, while Oja's rule does it through PCA. The more significant difference is that GMA is applicable to non-linear cases. In Sect. 3, it is shown that GMA can find out better results in dimension reduction by a non-linear model than only by the linear one.

GIPSCAL: There has been another similar approach to the proposed method, which is named GIPSCAL [18]. In order to minimize a general criterion which can manage even an asymmetric disparity matrix, GIPSCAL derives dynamical differential equations by applying gradient descent to the criterion. Then, stepwise update equations are derived in the similar way as in GMA. Though GIPSCAL can efficiently utilize arbitrary (often asymmetric) disparity matrices, it needs to calculate all the elements of the disparity matrix in advance and its cost at each time step is of quadratic order as in many other MDS methods. On the other hand, because GMA can separate the criterion by "stochastic" gradient descent, the cost is of linear order. The efficiency is the most significant advantage of GMA over GIPSCAL.

Stochastic gradient methods in MDS: In the previous works of the authors [19], [20], the computational cost of MDS is reduced by using stochastic approximation in the same way as in this paper. The algorithm in [19] includes an additional term of $\sum_{i,j} \log\left(\sum_k \left(y_{ik} - y_{jk} \right)^2 \right)$ in the stress (named "LOGSTRESS"), so it has to employ the tree method for the optimization (additional computation). In the algorithm named "the stochastic MDS network" in [20], the mapping of the objects in the low-dimensional space is constrained to a discrete grid structure. Though this model is suitable for the SOM-like algorithms [6], [21], it needs additional computation. On the other hand, the computational cost of GMA at each time step is only $O\left(NL^2\right)$. In addition, GMA can directly solve the two well-known MDS models (the classical MDS and ALSCAL).

3. Experiments on Artificial Data

Here, experimental results on artificial data are shown. Each column of $\boldsymbol{X}$ is given according to an N-dimensional Gaussian distribution satisfying

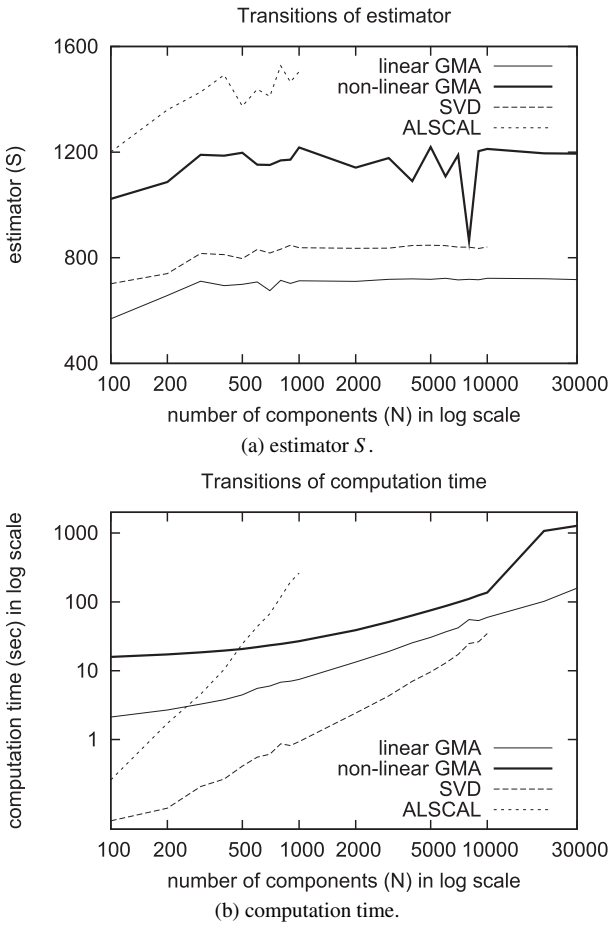
$$\mathrm{cov}\,(i,j) \simeq \begin{cases} 1.0, & \text{when } i = j, \\ 0.2, & \text{when } i \bmod 4 = j \bmod 4, \\ 0.0, & \text{otherwise,} \end{cases} \tag{20}$$

where $\mathrm{cov}\,(i,j)$ means the covariance between the i-th and j-th components. In this distribution, the N components are equally divided into four weak clusters. $M = 10000$ samples were generated by this distribution. So, $\boldsymbol{X}$ is an $N \times 10000$ matrix. The number of dimensions in the low-dimensional space L was set to 2. The linear GMA (Sect. 2.1) and the non-linear one (Sect. 2.2) were applied to find out the optimal mapping in the L-dimensional space. The update of $\boldsymbol{Y}$ in each GMA algorithm was repeated for 100000 time steps. For comparison, SVD was carried out by the ARPACK package using the efficient Lanczos method [13]. Moreover, ALSCAL with the Newton-Raphson method was also used for optimizing SSTRESS [5]. The initial $\boldsymbol{Y}$ is given randomly for linear GMA and ALSCAL. Non-linear GMA uses the result of linear GMA as the initial $\boldsymbol{Y}$. Therefore, the computation time of non-linear GMA includes the time of linear GMA for the initialization. All the computations were carried out by an Intel Xeon 2.66 GHz quad-core processor with 16 GB memory through matlab codings. In order to estimate the formed two-dimensional map of the components, the following estimator S was used:

$$S = \frac{\overline{\mathrm{dist}}_y[\mathrm{between}]}{\overline{\mathrm{dist}}_y[\mathrm{within}]}. \tag{21}$$

where $\overline{\mathrm{dist}}_y[\mathrm{between}]$ is the averaged square Euclidean distance between two components belonging to different clusters in Eq. (20) and $\overline{\mathrm{dist}}_y[\mathrm{within}]$ is the averaged distance among all the components within the same clusters. Larger S means that the clusters are extracted more distinctly in the map. S is defined in the similar way as in the Fisher's estimator in linear discriminant analysis. Because non-linear optimization often converges to a quite inferior local minimum, the medians over ten runs were employed in all the experiments in order to evaluate the results without the effects of a few quite inferior results.

Figure 1 shows the transitions of S (a) and computation time (b) along the number of components N for linear GMA (thin solid curve), non-linear GMA (thick solid), SVD with the Lanczos method (dashed), and ALSCAL with the Newton-Raphson method (dotted). N was from 100 to 30000. The results in ALSCAL beyond $N = 1000$ and those in SVD beyond $N = 10000$ could not be calculated because they exhausted the memory (described later). First, Fig. 1 (a) shows the performance of each method in this problem. Though there were some fluctuations, it shows that the best method was non-linear GMA for large N. Though



(a) estimator S.

(b) computation time.

Fig. 1 Transitions of (a) the estimator S and (b) the computation time along the number of components N: Linear GMA (thin solid curve), non-linear GMA (thick solid), SVD using the Lanczos method (dashed), and ALSCAL using the Newton-Raphson method (dotted) were applied to the artificial data. All the results were the medians over ten runs.

ALSCAL was superior to non-linear GMA, it was not available when N was large. So, those results verify the effectiveness of GMA for quite high-dimensional problems. Second, Fig. 1 (b) shows that SVD was more efficient than GMA if SVD was available. However, regarding the order of the time complexity (corresponding to the slope of each curve in the log-log plot), the order of GMA was clearly smaller than SVD. Therefore, it is expected that GMA is more efficient than SVD if N is much larger. Regarding the space complexity, GMA was much more efficient than ALSCAL and SVD. As described in the above, ALSCAL and SVD were not available when N was quite large. It was because they exhausted the 16 GB memory. Though it is difficult to estimate accurately the used memory size, the relatively complicated matrix manipulations in ALSCAL and SVD need huge memory generally. On the other hand, GMA needs relatively small memory for the data matrix $\boldsymbol{X}$ and the mapping $\boldsymbol{Y}$ only. So, those results also verify the efficiency and the applicability of GMA in quite high-dimensional problems.

In order to investigate the relation between the number of iterations in GMA and the accuracy of the results, Fig. 2

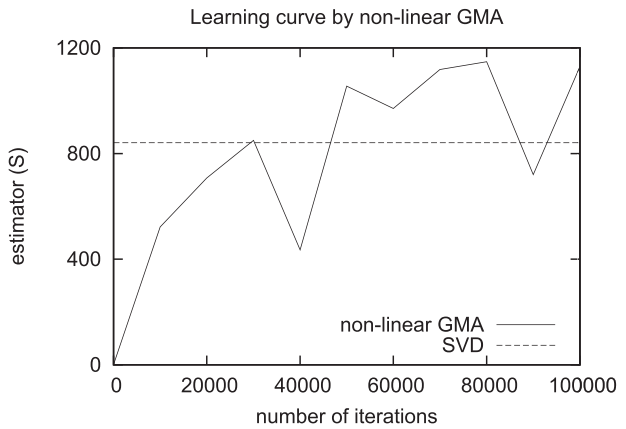


Fig. 2 Learning curve by non-linear GMA: It shows the transition of the estimator S by non-linear GMA along the number of iterations for the artificial data ($N = 10000$). It also shows the estimator by SVD as the dashed baseline. All the results were the medians over ten runs.

shows the learning curve of the estimator S by non-linear GMA. Though there were some fluctuations, it shows that non-linear GMA outperformed SVD roughly after 50000 iterations.

4. Analysis of Massive Document Collections

Here, the linear and non-linear GMA methods and the SOM [7] are applied to massive document collections (a set of about 80000 articles (posted to a Usenet newsgroup) and another of about 1500000 ones (posted to the whole Usenet)). In those cases, SVD and ALSCAL are not applicable because a quite large distance matrix such as 80000×80000 exhausts the memory. On the other hand, GMA and the SOM is applicable because they utilize stochastic gradient algorithms where the direct calculation of the distance matrix is not necessary.

4.1 Analysis of Articles Posted to a Group

4.1.1 Experimental Conditions

Raw data. 84817 articles posted to a Usenet newsgroup "`comp.os`" from Jan. 2001 to Feb. 2001 were used.

Extractions. Each article was decomposed into a set of words. Only the alphabetical words were extracted, and all the words were converted to their stems by WordNet [22]. In addition, the articles which consist of less than 5 words or more than 1000 words were removed. Consequently, 77280 distinct word stems and 84006 articles were extracted (namely, $N = 84006$).

Word selection. In order to select important words, the words which are included in more than 5000 articles and those which occur in less than 500 articles were removed. Then, only 1529 words were selected (Namely, $M = 1529$).

Data Matrix. The i-th row of $\boldsymbol{X}$ corresponds to the i-th article, where $x_{ik} = 1$ if the i-th article includes the k-th word, and $x_{ik} = 0$ otherwise.

Table 1 Computation Time on Massive Document Collections: This table shows the computation time (seconds) for the following three methods: linear GMA, non-linear GMA, and the SOM. Because non-linear GMA used the result of linear GMA as the initial $\boldsymbol{Y}$, it includes the additional time.

linear GMA	non-linear GMA	SOM
121	343	3590

GMA. L was set to 2. The linear and non-linear GMA methods were applied to $\boldsymbol{X}$. They were carried out for 10000 time steps, which was empirically enough for the convergence. Non-linear GMA uses the result of linear GMA as the initial $\boldsymbol{Y}$. The final $\boldsymbol{Y}$ was rotated by the usual PCA.

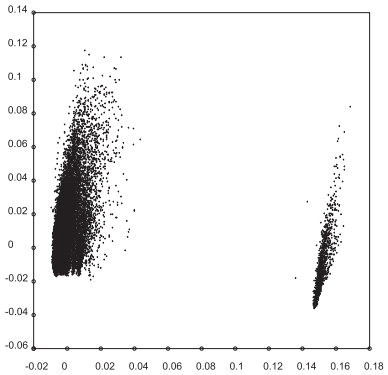
SOM. Because the SOM is not applicable if M is large ($M = 1529$), M was reduced to 500 by the random mapping method as in [7]. The reduced data matrix $\tilde{\boldsymbol{X}}$ is given by $\tilde{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{R}$. $\boldsymbol{R}$ is given as a 1529×500 random matrix, where only five randomly-selected components were set to 1 and the others were 0 in each row. It is easy to show that the square Euclidean distance between any two rows of $\tilde{\boldsymbol{X}}$ approximates that of $\boldsymbol{X}$. Each row of $\tilde{\boldsymbol{X}}$ was used as a training vector. A rectangular map of 100×100 units was used. The reference vectors of the units were trained by the simple SOM algorithm of SOM_PAK [23]. The training was terminated after 10000 steps during the first (ordering) phase. During the second (fine-tuning) phase, it was carried out for 100000 steps. Almost all the parameters and the preprocessing methods were based on [7].

Computation. All the computations were carried out by an Intel Xeon 2.66 GHz quad-core processor with 16 GB memory.

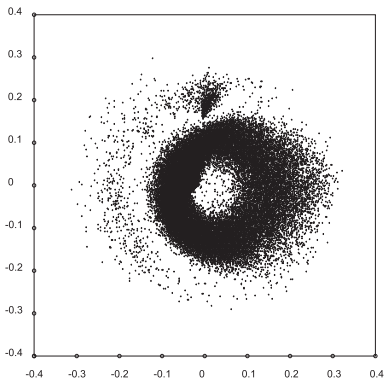
4.1.2 Results

Computation time. Table 1 shows the computation time taken by the linear and non-linear GMA methods and the SOM. These results show that GMA is much faster than the original SOM algorithm. It verifies the efficiency of GMA in these practical applications.

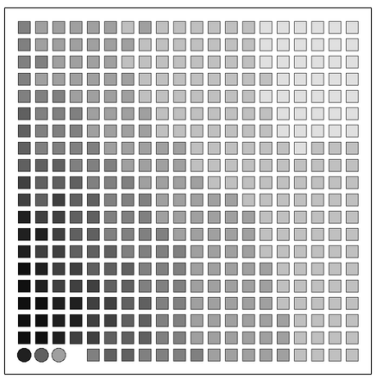
Formed Mapping. The formed mappings are shown in Fig. 3. Linear GMA (Fig. 3 (a)) found two (right and left) clusters quite clearly. Through close inspection, it was discovered that the right small cluster corresponds to the articles posted by a single author. In addition, Fig. 4 (a) shows that the vertical axis of Fig. 3 (a) corresponds to the number of words in an article. It is worth noting that the axis is common to the two clusters. In other words, the number of words (the vertical axis) and the distinction between the two clusters (the horizontal axis) were extracted independently. Regarding non-linear GMA (Fig. 3 (b)), the right and left clusters in Fig. 3 (a) correspond to the large circle around the center and the upper small one, respectively. Figure 4 (b) shows that the horizontal axis of



(a) linear GMA.



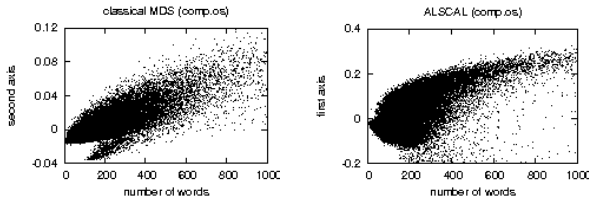
(b) non-linear GMA.



(c) SOM.

Fig. 3 Mappings for "comp.os": In (a) and (b) (mappings formed by GMA), the first and second axes of $\boldsymbol{Y}$ (which is rotated by the usual PCA) correspond to the horizontal and vertical ones. An article is represented by a dot. In (c) (a mapping formed by the SOM), the squares (■) and circles (●) correspond to the right and left clusters in (a). Here, the 100×100 SOM units were divided equally into 20×20 blocks. Each block (of 5×5 units) includes the articles nearest to a unit in the block. Then, the cluster of each block was determined by a majority vote from the articles which are included in the block. If there exists no article, the block is represented by a blank. In addition, the darkness of each block represents the average number of words over the included articles. In other words, a darker block includes longer articles.

Fig. 3 (b) roughly corresponds to the number of words in an article as well as linear GMA. Similarly, the SOM (Fig. 3 (c)) could distinguish the two clusters and could arrange the articles according to the number of words in *each* cluster. But, it failed to find the *common* axis. Even though there exists an axis representing the number of words in the intrinsic space of the articles, the SOM distorted it. Note that the effectiveness of applying GMA to this particular data is not discussed here. The characteristics of GMA are discussed and clarified by using this data just as an example.



(a) linear GMA. (b) non-linear GMA.

Fig. 4 The relation between the number of words and the axes in GMA for "comp.os": (a) The relation between the vertical axis of Fig. 3 (a) and the number of words of each article, where each dot corresponds to an article. (b) The relation between the horizontal axis of Fig. 3 (b) and the number of words.

4.2 Analysis of the Whole Usenet

4.2.1 Experimental Conditions

Raw data. 1518776 articles posted to the whole Usenet from Jan. 2001 to Feb. 2001 were used.

Extractions. The articles which consist of less than 5 words or more than 1000 words were removed. As a result, 1317864 word stems and 1396248 articles were extracted.

Word selection. By removing the words which are included in more than 50000 articles and those which occur in less than 5000 articles, 2419 words were selected.

GMA. Only linear GMA was applied for 10000 time steps. L was set to 2.

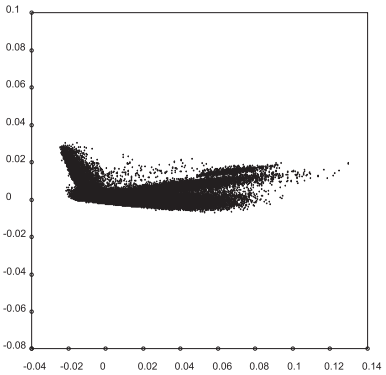
The other settings were the same as those in Sect. 4.1.1.

4.2.2 Results

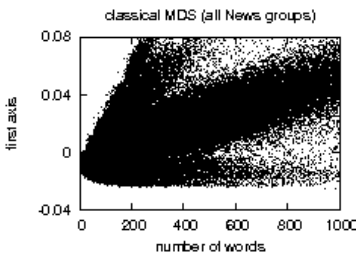
It took about 45 minutes to carry out linear GMA. The formed mapping is shown in Fig. 5 (a). Figure 5 (b) shows that the horizontal axis corresponds to the number of words of each article in the same way as for "comp.os." Figure 5 (a) shows that the articles are classified into three clusters (the main one on the center, the small ones on the upper-right and the upper-left of the main one.) These results verify that GMA is useful even for the analysis of quite massive document collections. The investigation of the "meanings" of the clusters found by GMA is beyond the scope of this paper.

5. Conclusion

In this paper, we proposed a new stochastic gradient algorithm (named "global mapping analysis" (GMA)), which minimizes the classical MDS stress efficiently. Extensions



(a) Mapping by GMA.



(b) Horizontal axis and the number of words.

Fig. 5 Results of GMA for the Whole Usenet: (a) The mapping formed by GMA. (b) The relation between the number of words and the horizontal axis of the mapping. Each dot corresponds to an article.

of GMA (GMA for PCA and GMA for SSTRESS) were also proposed. It was shown that GMA is equivalent to Oja's symmetrical PCA network rule. We showed the validity of GMA by applying it to artificial data and massive document collections. Moreover, we are planning to apply GMA to other real large-scale problems in the fields of data mining and knowledge discovery [24], [25]. In addition, we are planning to strengthen the theoretical foundations of GMA. Though the extensions of GMA for PCA and SSTRESS were proposed in this paper, the applicability of GMA to other stresses is still unclear. We are planning to construct the general framework of GMA and answer this applicability question. We are also planning to apply GMA to other related multivariate analysis methods (multiple correspondence analysis, factor analysis, and so on). On the other hand, because GMA has to separate the MDS stresses, their forms are limited to polynomial functions. So, GMA can not utilize many well-known stresses such as the Sammon's stress [4]. It also may be an interesting approach to approximate such non-polynomial stresses by polynomial functions and apply GMA to them.

Acknowledgments

This work is partially supported by Grant-in-Aid for Young Scientists (KAKENHI) 19700267.

References

[1] J.B. Kruskal and M. Wish, Multidimensional scaling, Sage Publications, Beverly Hills, Calif., 1978.

[2] T.F. Cox and M.A.A. Cox, Multidimensional Scaling, second ed., Chapman & Hall/CRC, 2001.

[3] K.V. Mardia, "Some properties of classical multidimensional scaling," Communications in Statistics: A, Theory and Method, vol.A7, no.13, pp.1233–1241, 1978.

[4] J.W. Sammon, "A nonlinear mapping for data structure analysis," IEEE Trans. Comput., vol.18, no.5, pp.401–409, 1969.

[5] Y. Takane, F.W. Young, and J. Deleeuw, "Nonmetric individual-differences multidimensional-scaling — Alternating least-squares method with optimal scaling features," Psychometrika, vol.42, no.1, pp.7–67, 1977.

[6] T. Kohonen, Self-organizing maps, Springer, Berlin, New York, 1995.

[7] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, J. Honkela, V. Paatero, and A. Saarela, "Self-organization of a massive document collection," IEEE Trans. Neural Netw., vol.11, no.3, pp.574–585, May 2000.

[8] G. Young and A.S. Householder, "Discussion of a set of points in terms of their mutual distances," Psychometrika, vol.3, no.1, pp.19–22, 1938.

[9] W.S. Torgerson, "Multidimensional scaling: 1. theory and method," Psychometrika, vol.17, no.4, pp.401–419, 1952.

[10] H. Robins and S. Monro, "A stochastic approximation method," Annals of Mathematical Statistics, vol.22, no.3, pp.400–407, 1951.

[11] Y. Matsuda and K. Yamaguchi, "Global mapping analysis: Stochastic gradient algorithm in SSTRESS and classical MDS stress," ICONIP2001 Proceedings, Shanghai, China, pp.102–107, 2001.

[12] Y. Matsuda and K. Yamaguchi, "An efficient MDS algorithm for the analysis of massive document collections," Knowledge-Based Intelligent Information and Engineering Systems: Proc. 9th International Conference, KES 2005, Part II, LNAI, vol.3682, Melbourne, Australia, pp.1015–1021, Springer-Verlag, Aug. 2005.

[13] G.H. Golub and C.F.V. Loan, Matrix Computations, Johns Hopkins University Press, 1989.

[14] J.C. Gower, "Some distance properties of latent root and vector methods used in multivariate analysis," Biometrika, vol.53, no.3/4, pp.325–338, 1966.

[15] E. Oja, "A simplified neuron model as a principal component analyzer," Journal of Mathematical Biology, vol.15, no.3, pp.267–273, 1982.

[16] E. Oja, "Neural networks, principal components, and subspaces," International Journal of Neural Systems, vol.1, no.1, pp.61–68, 1989.

[17] J.L. Wyatt and I.M. Elfadel, "Time-domain solutions of Oja's equations," Neural Computation, vol.7, no.5, pp.915–922, Sept. 1995.

[18] N.T. Trendafilov, "Gipscal revisited: A projected gradient approach," Statistics and Computing, vol.12, no.2, pp.135–145, 2002.

[19] Y. Matsuda and K. Yamaguchi, "Global mapping analysis: Stochastic approximation for multidimensional scaling," International Journal of Neural Systems, vol.11, no.5, pp.419–426, 2001.

[20] Y. Matsuda and K. Yamaguchi, "An efficient MDS-based topographic mapping algorithm," Neurocomputing, vol.64, pp.285–299, 2005.

[21] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten, "'Neural-gas' network for vector quantization and its application to time-series prediction," IEEE Trans. Neural Netw., vol.4, no.4, pp.558–569, July 1993.

[22] C. Fellbaum, ed., WordNet: An electronic lexical database, MIT Press, Cambridge, Massachusetts and London, England, 1998.

[23] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen, "SOM_PAK: The self-organizing map program package," Tech. Rep. A31, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[24] P. Adriaans and D. Zantinge, Data mining, Addison-Wesley, Harlow, Tokyo, 1996.

[25] P. Baldi, P. Frasconi, and P. Smyth, Modeling the Internet and the Web: Probabilistic Methods and Algorithms, John Wiley & Sons, New York, 2003.

Yoshitatsu Matsuda received his Ph.D. from the University of Tokyo, Japan in 2002. He is currently an assistant professor in Aoyama Gakuin University. His research interests include independent component analysis, massive data analysis, and self-organizing neural networks.

Kazunori Yamaguchi received the B.S., M.S., and Doctor of Science degrees in information science from the University of Tokyo, in 1979, 1981, and 1985, respectively. Currently, he is a professor of the University of Tokyo. His research interests are in data models and data analysis.