Fast Stochastic Neighbor Embedding: A Trust-Region Algorithm

Kijoeng Nam, Hongmo Je, Seungjin Choi
Department of Computer Science
POSTECH
San 31 Hyoja-dong, Nam-gu
Pohang 790-784, Korea
Email: {lovegod, invu71, seungjin}@postech.ac.kr

***Abstract*— Stochastic neighbor embedding (SNE) is a probabilistic method of embedding objects, described by high-dimensional vectors or by pairwise dissimilarities, into a lower-dimensional space in a way that neighbor identities are preserved [5]. Despite of the useful behavior of SNE, it suffers from its slow convergence due to a gradient-based implementation. In this paper we present a fast SNE algorithm which is approximately 4-6 times faster than the gradient-based SNE algorithm. Our fast SNE algorithm, named *TR-SNE* employs a trust-region (TR) method which finds a direction and a step size in an efficient and reliable manner with the help of a quadratic model of the objective function. We confirm the high performance and the fast convergence of our our TR-SNR through numerical experiments.**

I. Introduction

Dimensionality reduction is a fundamental problem in a variety of areas such as machine learning, pattern recognition, exploratory data analysis, data visualization, and so on. Various methods of dimensionality reduction have been studied so far. Among them, the most representative method might be principle component analysis (PCA), the goal of which is to find a linear projection such that transformed variables retain the maximum variance. PCA is the simplest and the most powerful linear dimensionality reduction method, however, it is confined to be a linear mapping. Multidimensional scaling (MDS) [4] aims at searching for a lower-dimensional space, usually Euclidean, in which points in the space represent objects such that the distances between the points in the space, match, as well as possible, the original dissimilarities between the objects described by high-dimensional vectors or by pairwise dissimilarities. In fact MDS attempts to find a lower-dimensional embedding which preserve dissimilarities, measured by Euclidean distance. Classical scaling is closely related with PCA. Isomap [10] generalizes MDS by taking geodesic distance instead of Euclidean distance into account, so that objects in a curved manifold are embedded into a lower-dimensional space with preserving neighborhood relation. Other nonlinear dimensionality reduction methods include locally linear embedding (LLE) [7], [8], locally linear coordination (LLC) [9], charting a manifold [3], and Laplacian eigenmap [1], [2].

Stochastic neighbor embedding (SNE) was recently proposed, which is as probabilistic lower-dimensional embedding method [5]. In contrast to other nonlinear dimensionality reduction methods, SNE is a probabilistic approach that preserves the *distribution of neighbor identities*. The probabilistic framework on dimensionality reduction makes it easy to embed without any constraints. We recognized that LLE and LLC required that the manifold of data points are convex and Isomap and Laplacian eigenmap assume that the manifold of data points is compact and dense. In SNE, the densities under a Gaussian, placed on each object in the high-dimensional space, are used to define a probability distribution over all the potential neighbors of the object. The SNE searches for a set of lower-dimensional images of the objects such that probability distributions over potential neighbors of these images, match as well as possible, the original distributions over the neighbors of objects. It was formulated as a minimization of a sum of Kullback-Leibler divergences and was implemented using a gradient descent method. A major obstacle in the the original SNE algorithm [5] for practical applications, is its slow convergence, resulting from the gradient descent implementation.

In this paper, we present a fast SNE algorithm, *TR-SNE*, where we employ the trust-region method which finds a direction and a step size in an efficient and reliable manner with the help of a quadratic model of the objective function [6]. The trust-region method defines a region around the current iterate within which it trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this trust region. It is, in general, faster than the steepest descent method and is free of a learning rate unlike the gradient-based methods. Its convergence is between linear and quadratic rate and its stability is always guaranteed, in contrast to the Newton method. The TR-SNE algorithm inherits several useful properties of trust-region methods, such as fast convergence and stability. The TR-SNE algorithm is approximately 4-6 times faster than the original gradient-based SNE algorithm. Its useful behavior is confirmed through several numerical experiments.

II. Stochastic Neighbor Embedding

Denote by $\boldsymbol{x}_t \in \mathbb{R}^D$ an object described by a D-dimensional vector. The vector $\vec{\boldsymbol{x}} \in \mathbb{R}^{DN}$ is a long vector that is con-

structed by stacking $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ in a single column. The image of $\boldsymbol{x}_t$ is denoted by $\boldsymbol{y}_t \in \mathbb{R}^d$ ($d \ll D$) and the vector $\vec{\boldsymbol{y}} \in \mathbb{R}^{dN}$ is constructed in a similar manner. The original SNE algorithm [5] is described below.

Step 1 **Neighbors Selection** Select neighbors by ϵ neighborhoods or k nearest neighbors.

Step 2 **Computing p_{ij} and q_{ij}** Compute the probability, p_{ij}, that $\boldsymbol{x}_i$ would pick $\boldsymbol{x}_j$ as its neighbor:

$$p_{ij} = \frac{\exp\left(-d_{ij}^2\right)}{\sum_{k \neq i} \exp\left(-d_{ik}^2\right)}, \tag{1}$$

where d_{ij}^2 are dissimilarities between two objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the high-dimensional space and σ_i is a Gaussian kernel width usually set by hand. The dissimilarities are computed by the scaled Euclidean distance

$$d_{ij}^2 = \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma_i^2}. \tag{2}$$

In the lower-dimensional space, the *induced* probability q_{ij} (with a fixed variance) that the image $\boldsymbol{y}_i$ pick $\boldsymbol{y}_j$ as its neighbor, is described by

$$q_{ij} = \frac{\exp\left(-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|\boldsymbol{y}_i - \boldsymbol{y}_k\|^2\right)}. \tag{3}$$

Step3 **A Cost Function** The aim of the embedding is to match p_{ij} and q_{ij} as well as possible. This is achieved by minimizing a cost function which is a sum of Kullback-Leibler divergences between p_{ij} and q_{ij} for each object. The cost function is given by

$$\mathcal{J} = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{4}$$

Step4 **Embedding through Steepest Descent** The set of images, $\vec{\boldsymbol{y}}$ in the lower-dimensional space, are updated by a gradient-descent method which has the form

$$\vec{\boldsymbol{y}}^{(k+1)} = \vec{\boldsymbol{y}}^{(k)} - \eta^{(k)} \nabla \mathcal{J}^{(k)}, \tag{5}$$

where $\eta^{(k)}$ is a learning rate and the gradient $\nabla \mathcal{J}$ is given by

$$\nabla \mathcal{J} = \left[\left(\frac{\partial \mathcal{J}}{\partial \boldsymbol{y}_1} \right)^T, \ldots, \left(\frac{\partial \mathcal{J}}{\partial \boldsymbol{y}_N} \right)^T \right]^T, \tag{6}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{y}_i} = 2 \sum_j (\boldsymbol{y}_i - \boldsymbol{y}_j)(p_{ij} - q_{ij} + p_{ji} - q_{ji}). \tag{7}$$

III. SNE through Trust-Region Methods

Here we first briefly review trust-region methods and present a fast SNE algorithm based on a trust-region method, which speeds up the convergence of the original SNE algorithm dramatically.

A. Trust-Region Methods

Trust-region methods [6] define a region around the current iterate within which they trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this trust region. In effect, they choose the direction and length of the step simultaneously. If a step is not acceptable, they reduce the size of the region and find a new minimizer. In general, the step direction changes whenever the size of the trust region is altered.

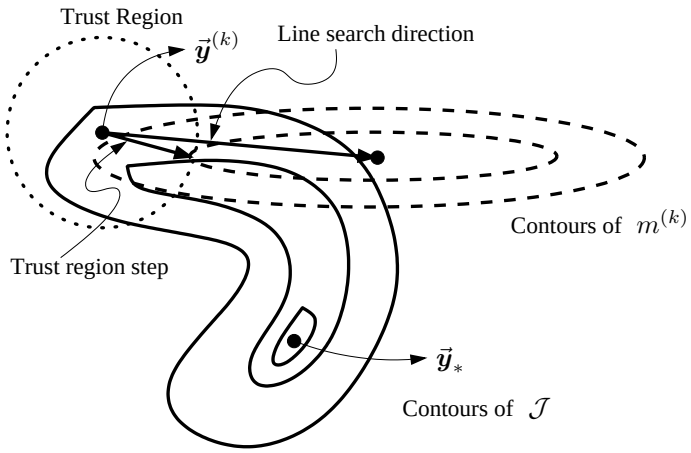


Fig. 1. An illustration of the trust-region method in determining a direction and a step size with the help of a quadratic model of the objective function.

Fig. 1 illustrates a trust-region approach for the minimization of an objective function $\mathcal{J}$ in which the current point lies at one end of a curved valley while the minimizer $\vec{\boldsymbol{y}}_*$ lies at the other end. A quadratic model function $m^{(k)}$, whose elliptical contours are shown as dashed lines, is based on function and derivative information at $\vec{\boldsymbol{y}}^{(k)}$ and possibly also on information accumulated from previous iterations and steps. A line search method based on this model searches along the step to the minimizer of $m^{(k)}$, but this direction allows only a small reduction in $\mathcal{J}$ even if an optimal step is taken. A trust-region method, on the other hand, steps to the minimizer of $m^{(k)}$ within the dotted circle, which yields a more significant reduction in $\mathcal{J}$ and a better step.

1) Outline of the Algorithm: The step $\vec{\boldsymbol{p}}$ is obtained by solving the following subproblem:

$$\min_{\|\vec{\boldsymbol{p}}\| \leq \Delta^{(k)}} m^{(k)}(\vec{\boldsymbol{p}}) = \mathcal{J}^{(k)} + \left[\nabla \mathcal{J}^{(k)}\right]^T \vec{\boldsymbol{p}} + \frac{1}{2} \vec{\boldsymbol{p}}^T \boldsymbol{B}^{(k)} \vec{\boldsymbol{p}}, \tag{8}$$

where $\boldsymbol{B}^{(k)} \in \mathbb{R}^{dN \times dN}$ is some symmetric matrix and $\Delta^{(k)} > 0$ is the trust-region radius and

$$\mathcal{J}^{(k)} = \mathcal{J}(\vec{\boldsymbol{y}}^{(k)}),$$
$$\nabla \mathcal{J}^{(k)} = \left. \frac{\partial \mathcal{J}}{\partial \vec{\boldsymbol{y}}} \right|_{\vec{\boldsymbol{y}} = \vec{\boldsymbol{y}}^{(k)}}. \tag{9}$$

The first issue to arise in defining a trust-region method is the strategy for choosing the trust-region radius $\Delta^{(k)}$ at

each iteration. Our choice of $\Delta^{(k)}$ is based on the agreement between the model function $m^{(k)}$ and the objective function $\mathcal{J}$ at previous iterations. Given a step $\vec{\boldsymbol{p}}^{(k)}$, this agreement measure $\rho^{(k)}$ is defined as the ratio of *actual reduction* to *predicted reduction*, i.e.,

$$\rho_k = \frac{\mathcal{J}(\vec{\boldsymbol{y}}^{(k)}) - \mathcal{J}(\vec{\boldsymbol{y}}^{(k)} + \vec{\boldsymbol{p}}^{(k)})}{m^{(k)}(\mathbf{0}) - m^{(k)}(\vec{\boldsymbol{p}}^{(k)})}. \tag{10}$$

Note that the predicted reduction, $m^{(k)}(\mathbf{0}) - m^{(k)}\left(\vec{\boldsymbol{p}}^{(k)}\right)$, is always nonnegative since the step $\vec{\boldsymbol{p}}^{(k)}$ is obtained by minimizing the model $m^{(k)}$ over a region that includes the step $\vec{\boldsymbol{p}} = \mathbf{0}$. Thus if $\rho^{(k)}$ is negative, the new objective value $\mathcal{J}\left(\vec{\boldsymbol{y}}^{(k)} + \vec{\boldsymbol{p}}^{(k)}\right)$ is greater than the current value $\mathcal{J}\left(\vec{\boldsymbol{y}}^{(k)}\right)$, so the step must be rejected. On the other hand, if $\rho^{(k)}$ is close to 1, there is good agreement between the model $m^{(k)}$ and the function $\mathcal{J}$ over this step, so it is safe to expand the trust region for the next iteration. If $\rho^{(k)}$ is positive but not close to 1, we do not alter the trust region, but if it is close to zero or negative, we shrink the trust region.

Trust-Region Algorithm

Given $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, and $\eta \in [0, \frac{1}{4})$;

for $k = 0, 1, 2, \ldots$

Obtain $\vec{\boldsymbol{p}}^{(k)}$ by (approximately) solving Eq. (8);

Update RTN:

Evaluate $\rho^{(k)}$ in Eq. (10);

if $\rho^{(k)} < \frac{1}{4}$

$\Delta^{(k+1)} = \frac{1}{4}\|\vec{\boldsymbol{p}}^{(k)}\|$

else

if $\rho^{(k)} > \frac{1}{4}$ and $\|\vec{\boldsymbol{p}}^{(k)}\| = \Delta^{(k)}$

$\Delta^{(k+1)} = \min\,(2\Delta^{(k)}\,,\,\bar{\Delta})$

else

$\Delta^{(k+1)} = \Delta^{(k)}$;

if $\rho^{(k)} > \eta$

$\vec{\boldsymbol{y}}^{(k+1)} = \vec{\boldsymbol{y}}^{(k)} + \vec{\boldsymbol{p}}^{(k)}$

else

$\vec{\boldsymbol{y}}^{(k+1)} = \vec{\boldsymbol{y}}^{(k)}$;

end (for)

Here $\bar{\Delta}$ is an overall bound on the step lengths. There are three strategies for finding approximate solution of Eq. (8). The first strategy is the *dogleg method* which is appropriate when the model Hessian $\boldsymbol{B}^{(k)}$ is positive definite. The second strategy is the *two-dimensional subspace minimization* which can be applied when $\boldsymbol{B}^{(k)}$ is indefinite. The third strategy is the *Steighaug's approach* which is the most appropriate when $\boldsymbol{B}^{(k)}$ is the exact Hessian $\nabla^2\mathcal{J}(\boldsymbol{y}^{(k)})$ (TR Newton-CG method) and when this matrix is large and sparse. (See Ch. 4 and 6 in [6] for further details)

2) Stability of the Algorithm: Trust-region methods guarantee the global convergence, which is stated in the following theorem (See [6] for the proof).

Theorem 1: Suppose that $\|\boldsymbol{B}^{(k)}\| \leq \beta$ for some constant β, that $\mathcal{J}$ is bounded below on the level set $\{\boldsymbol{y}|\mathcal{J}(\boldsymbol{y}) \leq \mathcal{J}\left(\boldsymbol{y}^{(0)}\right)\}$, and that all approximate solutions of Eq. (8) satisfy the inequalities

$$m^{(k)}(\mathbf{0}) - m^{(k)}(\vec{\boldsymbol{p}}^{(k)}) \geq c_1\|\nabla C^{(k)}\| \min\left(\Delta^{(k)}, \frac{\|\nabla\mathcal{J}^{(k)}\|}{\|\nabla\boldsymbol{B}^{(k)}\|}\right),$$

where $0 < c_1 \leq 1$ and $\|\vec{\boldsymbol{p}}^{(k)}\| \leq \gamma\Delta^{(k)}$ for some $\gamma \geq 1$.

(a) If $\eta = 0$ in the trust-region algorithm and $\mathcal{J}$ is continuously differentiable, then we have

$$\lim_{k\to\infty} \inf \|\nabla\mathcal{J}^{(k)}\| = 0.$$

(b) If $\eta = (0, \frac{1}{4})$ in the trust-region algorithm and $\mathcal{J}$ is Liptchitz continuously differentiable, then we have

$$\lim_{k\to\infty} \nabla\mathcal{J}^{(k)} = 0.$$

B. TR-SNE

1) Algorithm Outline: The trust-region method require a model Hessian matrix $\boldsymbol{B}$. Since it is possible to compute the exact Hessian $\nabla^2\mathcal{J}(\vec{\boldsymbol{y}})$ in SNE, we replace $\boldsymbol{B}$ in Eq. (8) by $\nabla^2\mathcal{J}(\vec{\boldsymbol{y}})$. The Hessian matrix $\nabla^2\mathcal{J}(\vec{\boldsymbol{y}}) \in \mathbb{R}^{dN\times dN}$ is computed as

$$\nabla^2\mathcal{J}(\vec{\boldsymbol{y}}) = \begin{bmatrix} \frac{\partial^2\mathcal{J}}{\partial\boldsymbol{y}_1\partial\boldsymbol{y}_1} & \cdots & \frac{\partial^2\mathcal{J}}{\partial\boldsymbol{y}_1\partial\boldsymbol{y}_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2\mathcal{J}}{\partial\boldsymbol{y}_N\partial\boldsymbol{y}_1} & \cdots & \frac{\partial^2\mathcal{J}}{\partial\boldsymbol{y}_N\partial\boldsymbol{y}_N} \end{bmatrix}, \tag{11}$$

where

$$\frac{\partial^2\mathcal{J}}{\partial\boldsymbol{y}_i\partial\boldsymbol{y}_j} = -2(p_{ji} - q_{ji} + p_{ij} - q_{ij})\text{diag}(y_{1i}, \ldots, y_{di}). \tag{12}$$

Note that p_{ij} is asymmetric, so $\boldsymbol{B}$ is also asymmetric. Therefore we use a symmetric form defined by $\frac{\boldsymbol{B}^T+\boldsymbol{B}}{2}$ instead of $\boldsymbol{B}$.

We use the Steihaug method which employs the conjugate-gradient (CG) method with a Steihaug's termination test. Compared to the dogleg and the subspace method where solving the linear system of involving $\boldsymbol{B}$ or $(\boldsymbol{B} + \alpha\boldsymbol{I})$ (for some $\alpha \in \mathbb{R}$) is costly, the Steihaug method is a TR-Newton-CG when $\boldsymbol{B}$ is an exact Hessian of the objective function. The Steihaug method has several attractive properties: (1) It requires no matrix factorization, so we can exploit the sparse structure of the Hessian $\nabla^2\mathcal{J}$ without worrying about fill-in during a direct factorization; (2) When the Hessian matrix is positive definite, the Newton-CG method approximates the pure Newton step more and more closely as the solution $\vec{\boldsymbol{y}}_*$ is approached, so rapid convergence is also possible. When Hessian matrix is not positive definite, we can make it positive definite by adding $\lambda\boldsymbol{I}$; (3) The CG iteration -the most computationally intensive part of the algorithm- may be executed in parallel, since the key operation is a matrix-vector product. In this paper we use

TR Newton-CG method implemented through the *fminunc* in Matlab Toolbox. The pseudo code of the TR-SNE algorithm is described below.

TR-SNE Algorithm

Given N data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \in \mathbb{R}^D$;
Initialize the lower-dimensional images
$\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\} \in \mathbb{R}^d$ with random numbers;
Set $k = 0, \varepsilon = 10^{-5}, \delta = 10^{-5}$;
do
Compute p_{ij} and q_{ij};
Compute the cost function $\mathcal{J}$ by solving Eq. (4);
if $\mathcal{J} < \varepsilon$ stop condition is satisfied;
Find $\vec{\boldsymbol{p}}^{(k+1)}$ by using TR Newton-CG $[\nabla^2 \mathcal{J}(\vec{\boldsymbol{y}}^{(k)})]$;
Update $\vec{\boldsymbol{y}}^{(k+1)}$ by using **Update-RTN** in TR Algorithm;
Set $k = k + 1$;
if $\|\vec{\boldsymbol{y}}^{(k+1)} - \vec{\boldsymbol{y}}^{(k)}\| < \delta$, stop condition is satisfied;
end (do) until any stop conditions are satisfied

TR Newton-CG [parameter B]

Modify $\boldsymbol{B}$ as $\frac{\boldsymbol{B}^T + \boldsymbol{B}}{2}$
Given $\epsilon > 0$;
Set $\vec{\boldsymbol{p}}^{(0)} = \boldsymbol{0}, \boldsymbol{r}^{(0)} = \nabla \mathcal{J}^{(0)}, \boldsymbol{d}^{(0)} = -\boldsymbol{r}^{(0)}$;
if $\|\boldsymbol{r}^{(0)}\| < \epsilon$
return $\vec{\boldsymbol{p}} = \vec{\boldsymbol{p}}^{(0)}$;
for $j = 0, 1, 2, \ldots$
if $[\boldsymbol{d}^{(j)}]^T \boldsymbol{B} \boldsymbol{d}^{(j)} \leq 0$
Find τ such that $\vec{\boldsymbol{p}} = \vec{\boldsymbol{p}}^{(j)} + \tau \boldsymbol{d}^{(j)}$
minimizes $m(p)$ in Eq. (8), satisfies $\|\vec{\boldsymbol{p}}\| = \Delta$;
return
Set $\alpha^{(j)} = [\boldsymbol{r}^{(j)}]^T \boldsymbol{r}^{(j)} / [\boldsymbol{d}^{(j)}]^T \boldsymbol{B} \boldsymbol{d}^{(j)}$;
Set $\vec{\boldsymbol{p}}^{(j+1)} = \vec{\boldsymbol{p}}^{(j)} + \alpha^{(j)} \boldsymbol{d}^{(j)}$;
if $\|\vec{\boldsymbol{p}}^{(j+1)}\| \geq \Delta$
Find $\tau \geq 0$ such that $\vec{\boldsymbol{p}} = \vec{\boldsymbol{p}}^{(j)} + \tau \boldsymbol{d}^{(j)}$
satisfies $\|\vec{\boldsymbol{p}}\| = \Delta$;
return $\vec{\boldsymbol{p}}$;
Set $\boldsymbol{r}^{(j+1)} = \boldsymbol{r}^{(j)} + \alpha^{(j)} \boldsymbol{B} \boldsymbol{d}^{(j)}$;
if $\|\boldsymbol{r}^{(j+1)}\| < \epsilon \|\boldsymbol{r}^{(0)}\|$
return $\vec{\boldsymbol{p}} = \vec{\boldsymbol{p}}^{(j+1)}$;
Set $\beta^{(j+1)} = [\boldsymbol{r}^{(j)}]^T \boldsymbol{r}^{(j+1)} / [\boldsymbol{r}^{(j)}]^T \boldsymbol{r}^{(j)}$;
Set $\boldsymbol{d}^{(j+1)} = \boldsymbol{r}^{(j+1)} + \beta^{(j+1)} \boldsymbol{d}^{(j)}$;
end (for)

2) Convergence Rate [SNE vs. TR-SNE] : The convergence rate of the steepest descent method is linear, i.e., the objective value $\mathcal{J}^{(k)}$ converge to a minimum at a linear rate. On the other hand, the convergence rate of TR Newton-CG method becomes closer and closer to the pure Newton step, i.e, the function value $\mathcal{J}^{(k)}$ converge to a minimum at a quadratic rate in sufficiently local region by following theorem. Therefore the TR-SNE algorithm based on the TR Newton-CG method is much faster than the original SNE algorithm based on the steepest descent method.

Theorem 2: Let $\mathcal{J}$ be twice Lipschitz continuously differentiable. Suppose the sequence $\{\vec{\boldsymbol{y}}^{(k)}\}$ converges to a point $\vec{\boldsymbol{y}}_*$ that satisfies the second-order sufficient conditions and that for all k sufficiently large, the trust-region algorithm based on Eq. (8) with $\boldsymbol{B}^{(k)} = \nabla^2 \mathcal{J}(\vec{\boldsymbol{y}}^{(k)})$ choose steps $\vec{\boldsymbol{p}}^{(k)}$ that achieve at least the same reduction as the Cauchy point (that is, $m^{(k)}(\vec{\boldsymbol{p}}^{(k)}) \leq m^{(k)}([\vec{\boldsymbol{p}}^{(k)}]^c)$) and are asymptotically exact whenever $\|[\vec{\boldsymbol{p}}^{(k)}]^N\| \leq \frac{1}{2}\Delta^{(k)}$, that is ,

$$\|\vec{\boldsymbol{p}}^{(k)} - [\vec{\boldsymbol{p}}^{(k)}]^N\| = o(\|[\vec{\boldsymbol{p}}^{(k)}]^N\|).$$

Then the trust-region bound Δ_k becomes inactive for all k sufficiently large.

IV. Numerical Experimental Results

We used USPS digit data and Frey face DB in the task of nonlinear dimensionality reduction, in order to evaluate the performance of the TR-SNE algorithm and to compare it to the original SNE algorithm where the learning rate $\eta^{(k)}$ in Eq. (5) was fixed as 0.1.

A. Comparison of the convergence rate

The first experiment was carried out with USPS digit data. In the experiment, we chose 1000 digits among the overall set of data. To avoid very similar pairs like 3 to 5 or 7 to 9, two hundred samples from each five classes (0,1,2,3, and 4) are randomly selected. The comparison of the run-time as the number of neighbors grow, is summarized in Table I. Fig. 2 shows the plot of objective function values versus the number of iterations when the size neighbor, k is 10. We can see the convergence rate of TR-SNE is superior to the original SNE. In the experiment, the TR-SNE achieved the convergence at 23 iterations, whereas it took 692 iterations for the original SNE to achieve the convergence. Fig. 3 illustrates the average of execution times as the number of neighbor increase. As the result says, TR-SNE is about three to six times as fast as SNE.

TABLE I
Comparison of the execution time as the number of neighbors grow.

k	10	30	50	100	200	300	500
SNE	1113	2086	3025	6087	12903	34140	53200
TR-SNE	220	470	1051	1281	2219	6401	15955

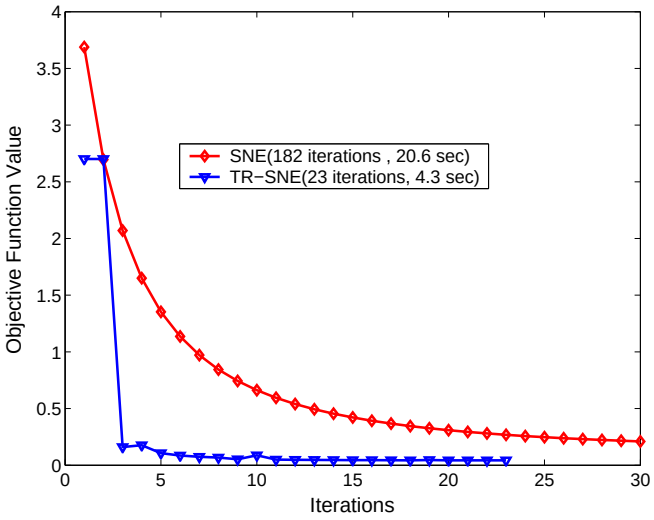


Fig. 2. The comparison of convergence speed in terms of the objective value vs. the number of iterations.

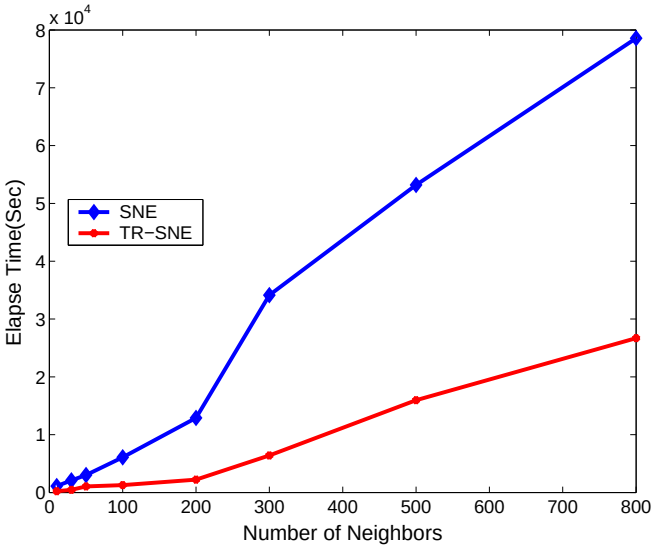


Fig. 3. The comparison of run-time as the number of neighbor grows.

B. Data Visualization

As mentioned pervious section, SNE can be applied in Data Visualization that is a reduction problem to visualize high dimensional data using only 2 or 3 dimensional data. Now, we are focusing on the embedding results to preserve the identity of neighbors of the original data in as well as possible.

Fig. 4 indicates a result of embedding on USPS data using SNE and TR-SNE, respectively. From this result, we note that both SNE and TR-SNE translate 256 dimensional space into two dimensional space with most closely distribution. Even though all embedded data points corresponding the original data have different absolute-coordinator between those methods, we can aware almost same distributed form to each other. The only difference between SNE and TR-SNE is nothing except the convergence rate.

In addition, we attempt to embedding on FREY data which has about 2000 face samples with a variety of pose and expression. We chose randomly 530 images from whole data as test images. In Fig. 5, we shows the result of TR-SNE on FREY data that has been embedded 560 dimension into two dimensional space. From the result, we can find a intrinsic property of the data. For example, the same pose images are locally nearby each other or similar expression images are closely distributed. In this experiment, the convergence time of TR-SNE is approximately 4.2 times as fast as SNE.

V. Conclusions

The original SNE algorithm based on the steepest descent method suffered from its slow convergence. To cope with this limitation, a simple idea was suggested in [5]. By adding a jitter (noise) for each iteration, it could accelerate the convergence rate slightly. However, it is still at a linear convergence rate.

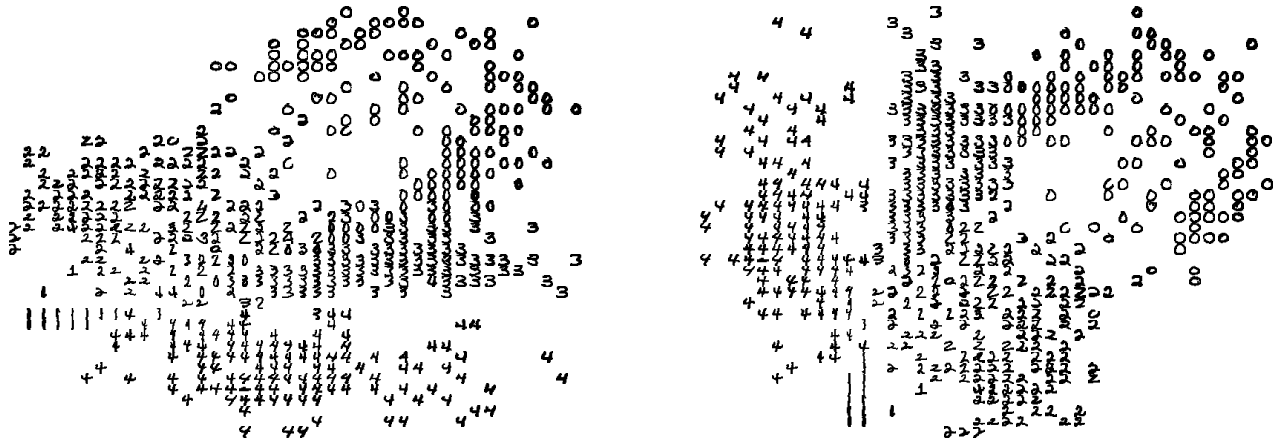
Trust-region methods guarantee the globally linear and locally quadratic convergence rate. We have presented a fast SNE algorithm, TR-SNE, which employed a trust-region method. We have confirmed the high performance and fast convergence of our TR-SNE algorithm through several numerical experiments.

Acknowledgment

This work was supported by Korea Ministry of Science and Technology under Brain Science and Engineering Research Program and under International Cooperative Research Program, by KOSEF 2000-2-20500-009-5, and by BK 21 in POSTECH.

References

[1] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *Advances in Neural Information Processing Systems*, vol. 14. MIT Press, 2002.
[2] ——, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural Computation*, vol. 15, pp. 1373–1396, 2003.
[3] M. Brand, “Charting a manifold,” in *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2003.
[4] T. Cox and M. Cox, *Multidimensional Scaling, 2nd Edition*. Chapman & Hall, 2001.
[5] G. Hinton and S. Roweis, “Stochastic neighbor embedding,” in *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2003.
[6] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.
[7] S. T. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, pp. 2323–2326, 2000.
[8] L. Saul and S. T. Roweis, “Think globally, fit locally: Unsupervised learning of low dimensional manifolds,” *Journal of Machine Learning Research*, vol. 4, pp. 119–155, June 2003.
[9] Y. W. Teh and S. Roweis, “Automatic allignment of local representations,” in *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2003.
[10] J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, pp. 2319–2323, 2000.



(a) (b)

Fig. 4. The embedding of USPS data (0-4) in two dimensional space: (a) the gradient-based SNE [5]; (b) our TR-SNE.

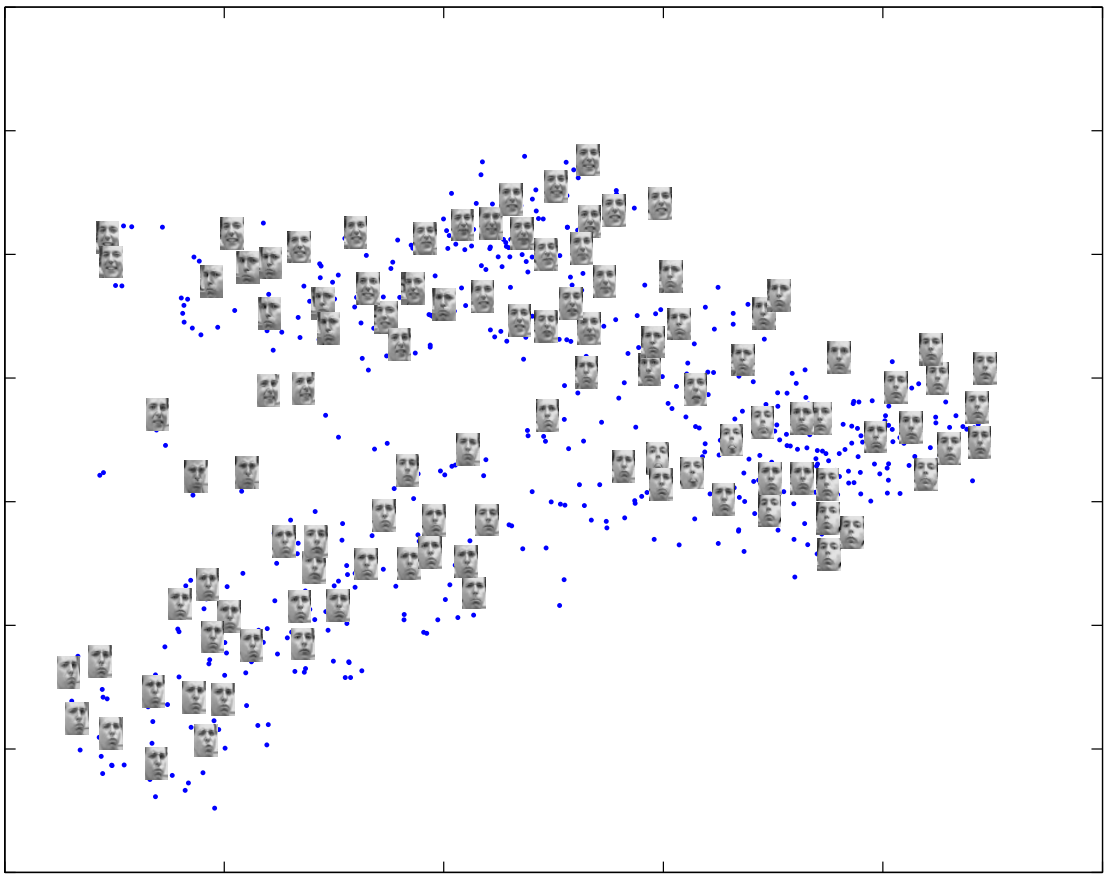


Fig. 5. The result of TR-SNE on face data