Scalable Optimization of Neighbor Embedding for Visualization

Zhirong Yang[1] ZHIRONG.YANG@AALTO.FI
Jaakko Peltonen[1,2] JAAKKO.PELTONEN@AALTO.FI
Samuel Kaski[1,2,3] SAMUEL.KASKI@AALTO.FI
[1]Department of Information and Computer Science, Aalto University, Finland
[2]Helsinki Institute for Information Technology HIIT, Aalto University and [3]University of Helsinki

Abstract

Neighbor embedding (NE) methods have found their use in data visualization but are limited in big data analysis tasks due to their $O(n^2)$ complexity for n data samples. We demonstrate that the obvious approach of subsampling produces inferior results and propose a generic approximated optimization technique that reduces the NE optimization cost to $O(n \log n)$. The technique is based on realizing that in visualization the embedding space is necessarily very low-dimensional (2D or 3D), and hence efficient approximations developed for n-body force calculations can be applied. In gradient-based NE algorithms the gradient for an individual point decomposes into "forces" exerted by the other points. The contributions of close-by points need to be computed individually but far-away points can be approximated by their "center of mass", rapidly computable by applying a recursive decomposition of the visualization space into quadrants. The new algorithm brings a significant speed-up for medium-size data, and brings "big data" within reach of visualization.

1. Introduction

Very simple techniques dominate in visual analytics and large-scale information visualization, as the computational complexity of advanced nonlinear dimensionality reduction (NLDR) methods developed in the machine learning community is too high. There is an immediate need for methods for initial inspection, "looking at the data", for the sets that typically currently range in tens of thousands or millions of data samples. The scatterplot-type displays produced with NLDR are one of the most common tools for smaller sets and would be useful for the "bigger data" as well.

Neighbor embedding (NE) methods work by optimizing the low-dimensional output coordinates of samples, and some other NLDR methods by optimizing the mapping function. The cost functions typically measure preservation of distances or neighborhood relationships from the original space to the output space. Unfortunately, current *advanced NLDR methods do not scale well* to large data sets, as the computational complexity of their optimization is typically at least quadratic ($O(n^2)$ for n data samples), and may be cubic or even worse. This is unfeasible for large data sets, and in some cases also the memory complexity may grow unmanageably large. The naive solution, of applying NLDR to a subset of data only, is unsatisfactory as it neglects the vast majority of relationships in data. We demonstrate experimentally that the naive solution can lead to poor generalization.

In this paper we present novel, fast and scalable versions of several recent state-of-the-art NE methods, by applying an efficient approximation to their gradient terms. The key insight is that in the methods we consider, interactions between far-away data samples do not contribute much to the gradient, and hence their contribution can be computed with stronger approximations. Inspired by the Barnes-Hut simulation approach in n-body dynamical systems (Barnes & Hut, 1986), we use a quad-tree to approximate far-away pairwise interactions: for each data sample, the effect of a far-away sample is approximated by the effect of the mean of a hierarchical cluster. The farther away the samples, the less detailed clusters are needed.

Our principle for generating the approximate methods is generic and can be applied to several NE methods. In this paper we apply the principle to develop novel fast versions of Stochastic Neighbor Embedding

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

(SNE; Hinton & Roweis, 2002), t-distributed Stochastic Neighbor Embedding (t-SNE; van der Maaten & Hinton, 2008), Elastic Embedding (EE; Carreira-Perpiñán, 2010), and Neighbor Retrieval Visualizer (NeRV; Venna & Kaski, 2007; Venna et al., 2010). We show in experiments that our efficient NLDR methods clearly outperform the original state of the art methods in terms of computation time, and achieve essentially the same quality. We demonstrate that the new methods bring within reach of visualizations very large data sets which could not be processed in a reasonable time by the existing state-of-the-art methods.

2. Neighbor Embedding

We now present the methods we will make scalable. Neighbor Embedding (NE) is a family of methods for finding a configuration of data points in a lower-dimensional "embedding space". Given a set of multivariate data points $\{x_1, x_2, \ldots, x_n\}$, where $x_i \in \mathbb{R}^M$, their neighborhood is encoded in a square nonnegative matrix P, where P_{ij} is proportional to the probability that x_j is a neighbor of x_i. Neighbor Embedding finds a mapping $x_i \mapsto y_i \in \mathbb{R}^m$ for $i = 1, \ldots, n$ such that the neighborhoods are approximately preserved in the mapped space. Usually $m = 2$ or 3, and $m < M$. If the neighborhood in the mapped space is encoded in $Q \in \mathbb{R}^{n \times n}$ such that Q_{ij} is proportional to the probability that y_j is a neighbor of y_i, the NE task is to minimize $\mathcal{D}(P||Q)$ over $Y = [y_1, y_2, \ldots, y_n]^T$ for a certain divergence $\mathcal{D}$.

Different choices of P, Q and $\mathcal{D}$ give different Neighbor Embedding methods. Let $p_{ij} \geq 0$ and $q_{ij} \triangleq q\left(\|y_i - y_j\|^2\right) > 0$. The NE method Stochastic Neighbor Embedding (SNE; Hinton & Roweis, 2002) uses $P_{ij} = \frac{p_{ij}}{\sum_k p_{ik}}$, $Q_{ij} = \frac{q_{ij}}{\sum_k q_{ik}}$ and Kullback-Leibler divergence in the cost function, $D(P||Q) = \sum_i D_{\mathrm{KL}}(P_{i:}||Q_{i:})$. The q_{ij} is typically chosen to be proportional to the Gaussian distribution so that $q_{ij} = \exp\left(-\|y_i - y_j\|^2\right)$, or proportional to the Cauchy distribution so that $q_{ij} = (1 + \|y_i - y_j\|^2)^{-1}$.

Additionally bringing in the dual Kullback-Leibler divergence $D_{\mathrm{KL}}(Q||P)$ weighted by a tradeoff parameter $\lambda \in (0, 1)$, hence minimizing $\sum_i \lambda D_{\mathrm{KL}}(P_{i:}||Q_{i:}) + (1-\lambda) D_{\mathrm{KL}}(Q_{i:}||P_{i:})$ over Y, results in a method called NeRV. NeRV has an information retrieval interpretation of making a tradeoff between precision and recall; SNE is a special case ($\lambda = 1$) maximizing recall, whereas $\lambda = 0$ maximizes precision.

If $P_{ij} = \frac{p_{ij}}{\sum_{kl} p_{kl}}$ and $Q_{ij} = \frac{q_{ij}}{\sum_{kl} q_{kl}}$, the NE method with cost function $\min_Y D_{\mathrm{KL}}(P||Q)$ is called Symmetric SNE (s-SNE). When q_{ij} is proportional to the Cauchy distribution, it is also called t-SNE (van der Maaten & Hinton, 2008; t denotes the Student t-distribution with a single degree of freedom).

If neither neighborhood matrix is normalized and the divergence is the non-normalized Kullback-Leibler divergence $D_I(p||\rho q) = \sum_{ij}\left[p_{ij}\log\frac{p_{ij}}{\rho q_{ij}} - p_{ij} + \rho q_{ij}\right]$, with a constant scalar $\rho > 0$, the NE method is equivalent to the Elastic Embedding (Carreira-Perpiñán, 2010) when q_{ij} is set to be Gaussian. Some additional methods have also been introduced to the NE family, for example, LinLog (Noack, 2007) which uses Itakura-Saito divergence and HSSNE (Yang et al., 2009) which uses other heavy-tailed embedding functions. We skip their description to avoid notational clutter.

We make the common approximation of sparsifying P by zeroing the very small non-local input similarities. This makes sense for two reasons: 1) geodesics of curved manifolds in high-dimensional spaces can only be approximated by Euclidean distances in small neighborhoods; 2) most popular distances computed of weak or noisy indicators are not reliable over long distances. The matrix P is often built upon the k-nearest-neighbor graph, of which there exist various efficient construction methods (see e.g. Yianilos, 1993; Liu et al., 2004; Beygelzimer et al., 2006). Note that in general the output similarity matrix q (or Q) is dense. The costly computation of this part can be avoided by using the technique described in the next section.

3. Scalable Optimization for NE

The neighbor embedding methods in Section 2 have an objective $\mathcal{J}$ and partial derivative $\frac{\partial \mathcal{J}}{\partial y_i}$ of the form

$$\mathcal{J} = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\ , \qquad \frac{\partial \mathcal{J}}{\partial y_i} = \sum_{j=1}^{n} B_{ij}\left(y_i - y_j\right). \qquad (1)$$

where A_{ij} and B_{ij} are pairwise scalar terms computed between data points i and j. To exactly compute $\mathcal{J}$ or its gradient to Y takes $O(n^2)$ time in general, which is infeasible when n is large. To avoid heavy computation in visualizing large datasets, a common simple approach is to only calculate the embedding for a small subset of data samples. However, as shown in Section 4.1, the resulting visualization can be poor even for the subset because information on pairwise interactions outside the subset is not used in optimization. van der Maaten & Hinton (2008) proposed another workaround, where distances within the subset of points were computed taking the rest of the data into account, as multi-path connections between data points in the whole neighbor graph. However, their method cannot visualize the original neighborhood of the whole data, rather it embeds a reduced and blurred neighborhood among the subset. In addition, computing the multi-path connections is expensive.

Fast group-based approximation. We propose an approximation method to speed up the objective and gradient calculation in NE. The computational complexity in NE is essentially due to the coordinates and pairwise distances in the output space, which change at every step of optimization. The idea is that, in each complicated sum computed around a data point i in the output space (each sum over neighbors j of the point), the terms in the sum will be partitioned into several groups G_t^i and each group will be approximated as an interaction with a representative point of the group. Consider the following summation which appears commonly in NE objectives:

$$\sum_j f\left(\|y_i - y_j\|^2\right) = \sum_t \sum_{j \in G_t^i} f\left(\|y_i - y_j\|^2\right) \qquad (2)$$

$$\approx \sum_t |G_t^i| f\left(\|y_i - \hat{y}_t\|^2\right), \qquad (3)$$

where i is the starting data point, j are its neighbors, G_t^i are groups (subsets) of the neighbors j, $|G_t^i|$ is the size of the group, and $\hat{y}_t^i$ is the representative, e.g. mean, of the points in group G_t^i. Similarly, we can approximate the gradient of the above sum. Denote $g_{ij} = f'\left(\|y_i - y_j\|^2\right)$. We have

$$\sum_j g_{ij}\left(y_i - y_j\right) = \sum_t \sum_{j \in G_t^i} g_{ij}\left(y_i - y_j\right)$$

$$\approx \sum_t |G_t^i| f'\left(\|y_i - \hat{y}_t^i\|^2\right)\left(y_i - \hat{y}_t^i\right). \qquad (4)$$

The approximation within each group G_t^i is accurate when all points in the group are far enough from y_i. Otherwise we divide the group into subgroups and recursively apply the approximation principle to each subgroup, until the group contains a single point j where we directly calculate $f\left(\|y_i - y_j\|^2\right)$ or g_{ij}. This grouping hierarchy forms a tree-like structure. In practice, the tree does not need to be constructed separately around each point i; it is enough to use a single global tree. We thus omit the superscript i in what follows. Summations of $O(n)$ cost can be approximately calculated in $O(\log n)$ time if each split in the tree divides the data into roughly equal size parts.

Fast computation of Eqs. 3 and 4 suffices to speed up methods with pairwise separable NE objectives such as Elastic Embedding. For NE methods involving normalized similarities, each normalizing denominator is another sum which can be approximated before computation of the cost function or gradient using the same principle. Some methods like NeRV need inner summations in the objective or gradient, which can be approximated beforehand in the same way.

Tree construction. The above approximation originated in the *n-body problem* in physics, where fast computation of energy (objective) and forces (gradient) is needed to study systems with huge numbers of particles. Various hierarchical structures exist for the speedup, e.g. fast multipole method (Greengard & Rokhlin, 1987) and dual tree (Gray & Moore, 2001; 2003a;b). We choose the one by Barnes & Hut (1986) as it is simple. The cost of building such a tree is negligible compared to computation time of the NE objective or gradient. For notational brevity we only describe Barnes-Hut QuadTree for two dimensional output points; it is straightforward to extend the principle to the three dimensional case (OctTree).

A QuadTree is constructed as follows. First the root node is assigned the smallest bounding box that contains all data points, and a representative which is the mean of all points. If the bounding box contains more than one data point, it is divided into four smaller boxes of equal size, and a child node is constructed at each smaller bounding box if it contains at least one data point. The splitting is done recursively until all leaf nodes contain exactly one data point.

Fig. 1 shows an example QuadTree. Note that the tree respects point density—an area with more data points is split into deeper levels. Each tree node contains a group of data points. Let G_t denote the set of points within the bounding box in a branch (non-leaf) node t. The branch node also has the following data fields: a position $\hat{y}_t$ which is the mean of points in G_t, and a weight $|G_t|$ which equals the size of G_t. An optional coefficient may also be attached to a branch node to facilitate computation (e.g. to precompute means of some functions over the groups, needed for methods like NeRV). The memory cost of a QuadTree is thus linear in the number of data points.

Using the tree in computation. What remains is to define a criterion that a point y_i is far from a tree t. A straightforward idea is that the tree's bounding box is small enough compared to the distance $\|y_i - \hat{y}_t\|$. This can be implemented by

$$\theta \cdot \text{TreeWidth}(t) < \|y_i - \hat{y}_t\|, \qquad (5)$$

where *tree width* is the longer edge length of the tree's root node's bounding box and θ is a positive parameter that controls the tradeoff between computational cost and approximation accuracy. A larger θ gives more accurate approximation but needs more expensive computation. Typically $\theta \in [1.2, 5]$.

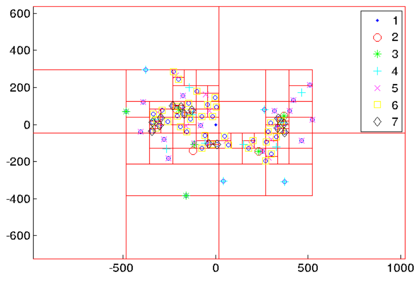

Figure 1. An example QuadTree for 62 data points. The color labeling corresponds to the depth of each data point.

As an example, we show how to apply the approximation to t-SNE. Again denote the objective by $\mathcal{J}$. The t-SNE gradient is $\frac{\partial \mathcal{J}}{\partial y_i} = 4 \sum_j \left(P_{ij} - s^{-1} q_{ij}\right) q_{ij} \left(y_i - y_j\right)$ where $s = \sum_{kl} q_{kl}$. We first approximate s by Eq. 3 and then further approximate the gradient by Eq. 4. This suffices for algorithms that only need gradients. For those which use line search, the objective can also be cheaply obtained: $\mathcal{J} \approx \sum_{ij} P_{ij} \|y_i - y_j\|^2 + \log \hat{s} +$ constant, where $\hat{s}$ is the approximation of s and we compute the first term by exploiting sparsity of P.

4. Experiments

We experimentally verify the performance of the algorithm, and demonstrate that alternative speedups are not comparable. The experiments were run on sixteen publicly available datasets (details in the supplemental document). For NE, each data set was converted to a symmetrized k-Nearest Neighbor matrix (k-NN; $k = 10$ in all reported results) with $p_{ij} = 1$ if j is among k nearest neighbors of i or vice versa, and $p_{ij} = 0$ otherwise. Conclusions are the same for $k = 15$ and $k = 20$, and for Gaussian kernel matrices.

4.1. More data in NE learning helps

We first test how well the straightforward fast alternative of subsampling and interpolation would perform. We compute the embedding for only a random subset of data, and then interpolate the rest in with Locality-constrained Linear Coding (Wang et al., 2010; Carreira-Perpiñán, 2010 presented another method for out-of-sample extension, which is however restricted to the EE method and requires expensive iterative learning). The result of embedding a 700-point subset of the 70,000 digit images from the MNIST data is shown in A1 of Figure 2, and with all points interpolated in C1. When compared to the optimal visualization computed for all the 70,000 points in C3, it is clear that the interpolation misses and mixes up a lot of the structure in the data.

One might argue that the smaller subset is already sufficiently large to give an overview of the whole set, and hence the larger visualization is not needed. In A3 the same small set of 700 points is shown, but on the display optimized for all the 70,000 points. The visualization reveals the classes and cluster structure much clearer than A1 which has been computed only on the small set.

The conclusion is that learning the visualizations on large data is beneficial, irrespective of whether only a subset or all data are to be visualized. The remaining problem is that embeddings with the largest data are expensive to compute: generating subfigure C3 takes about 46 hours. With the new algorithms a visualization of almost the same quality can be achieved in around 1.6 hours (Fig. 4 A).

4.2. Learning speed comparison

We tested the speed-up produced by the new approximation technique on three NE methods: t-SNE, s-SNE with Spectral Direction (SD) optimization (Vladymyrov & Carreira-Perpiñán, 2010), and s-SNE with momentum optimization (van der Maaten & Hinton, 2008). We will refer to the original ones by *exact algorithms* and the new ones by *approximated algorithms*.

First we compared the learning times. Each method was run for each dataset ten times, stopping when $\|Y^{\text{new}} - Y\|_F < 10^{-6} \cdot \|Y\|_F$ or the number of iterations exceeded 1000. To keep these computations manageable, only data sets with less than 20,000 samples were included. A clear speedup was obtained on all the data sets in terms of the mean elapsed times (Table 1).

We then verified that the approximated algorithms produce embeddings of comparable quality to the exact ones. We measured the quality with two quantities: relative difference Δ_o between the objective function values and relative difference Δ_a in the 10-NN classification accuracies ($\mathcal{A}$) computed after the embedding: $\Delta_o = |\mathcal{J}_{\text{exact}} - \mathcal{J}_{\text{approx}}| / |\mathcal{J}_{\text{exact}}|$ and $\Delta_a = |\mathcal{A}_{\text{exact}} - \mathcal{A}_{\text{approx}}| / |\mathcal{A}_{\text{exact}}|$. The measures are generally close to zero (Table 1), except for one dataset on s-SNE (SD). This indicates the approximation accuracy by using the proposed technique is satisfactory.

Finally, we study the computation times for two larger datasets: UCI Shuttle (size 58K) and MNIST handwritten digit images (size 70K). The value of the objective function reduces markedly faster with the approximated algorithm (Fig. 3). Furthermore, the exact and approximated values of the result produced by the approximated algorithm are almost identical, indicating excellent approximation accuracy.

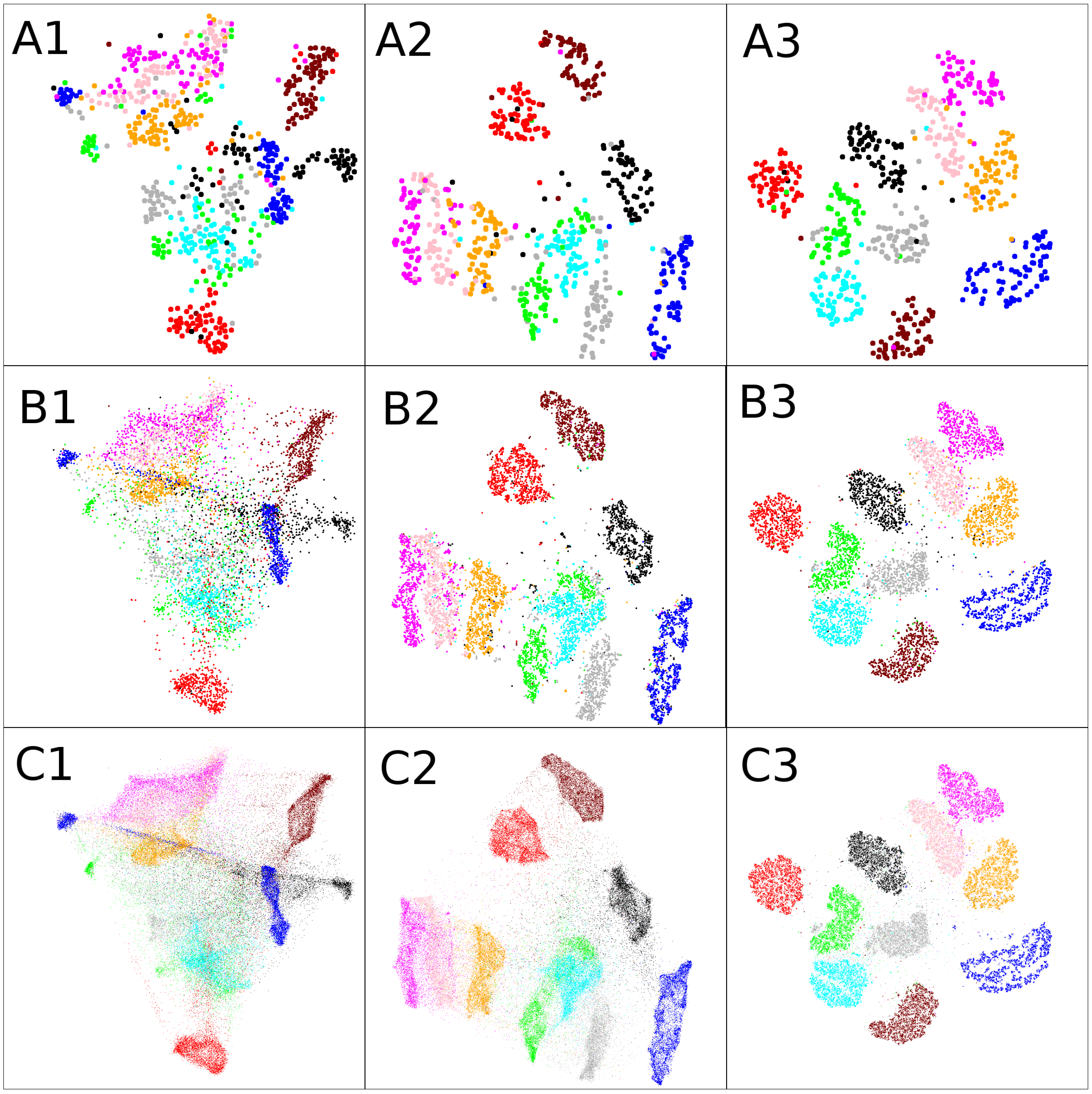

Figure 2. Effects of (random) subsampling on visualization quality. Rows: Amount of data shown on the display; A: 700, B: 7,000, C: 70,000. Columns: Amount of data in *learning* the visualization; 1: 700, 2: 7,000, 3: 70,000. In images below the diagonal, that is B1, C1, and C2, the rest of the samples have been interpolated with Locality-constrained Linear Coding. In images above the diagonal, A2, A3 and B3, the points have been subsampled after the learning. Data: MNIST handwritten digit images, with colors showing the 10 classes. Embedding method: t-SNE with Spectral Direction optimization. In summary, interpolation reduces quality (compare C1 and C3), and subsampling after computing the whole projection improves quality compared to subsampling beforehand (A3 vs A1).

Table 1. Effects of the new approximations on three NE methods, with 12 data sets. In each time column, the first figure is the elapsed time of the original (exact) algorithm and the second the approximated algorithm ("s":second, "m":minute, "h":hour). Δ_o: relative difference in objective function values (± standard deviation); Δ_a: relative difference in k-NN classification accuracy on the display.

		t-SNE (SD)			s-SNE (SD)			t-SNE (momentum)		
dataname	size	time	Δ_a	Δ_o	time	Δ_a	Δ_o	time	Δ_a	Δ_o
Iris	150	19s 2s	0.01 ± 0.00	0.02 ± 0.03	3s 2s	0.00 ± 0.00	0.03 ± 0.06	0.5s 0.5s	0.00 ± 0.00	0.01 ± 0.01
ORL	400	46s 14s	0.02 ± 0.01	0.01 ± 0.01	27s 3s	0.02 ± 0.02	0.04 ± 0.03	2s 4s	0.01 ± 0.01	0.01 ± 0.01
COIL	1K	4m 19s	0.01 ± 0.00	0.07 ± 0.01	7m 11s	0.01 ± 0.01	0.04 ± 0.03	25s 19s	0.00 ± 0.00	0.03 ± 0.01
Seg	2K	6m 1m	0.00 ± 0.00	0.06 ± 0.04	14m 20s	0.02 ± 0.05	0.46 ± 1.27	1m 37s	0.00 ± 0.00	0.03 ± 0.01
WebKB	4K	16m 7m	0.01 ± 0.01	0.01 ± 0.00	34m 2m	0.01 ± 0.02	0.00 ± 0.00	3m 1m	0.00 ± 0.00	0.01 ± 0.00
7Sectors	5K	20m 4m	0.02 ± 0.01	0.09 ± 0.04	52m 2m	0.05 ± 0.04	0.01 ± 0.01	4m 2m	0.00 ± 0.00	0.01 ± 0.00
OptDig	6K	32m 9m	0.00 ± 0.00	0.01 ± 0.00	1.7h 3m	0.01 ± 0.00	0.01 ± 0.01	6m 2m	0.00 ± 0.00	0.02 ± 0.00
Reuters	8K	1.1h 28m	0.01 ± 0.00	0.01 ± 0.01	2.0h 2m	0.07 ± 0.15	0.06 ± 0.15	14m 4m	0.00 ± 0.00	0.01 ± 0.00
RCV1	10K	1.0h 31m	0.00 ± 0.01	0.01 ± 0.00	3.1h 5m	0.02 ± 0.02	0.00 ± 0.00	18m 4m	0.00 ± 0.00	0.02 ± 0.00
Spam	10K	52m 28m	0.00 ± 0.00	0.01 ± 0.01	1.1h 7m	0.00 ± 0.00	0.00 ± 0.00	20m 6m	0.00 ± 0.00	0.01 ± 0.00
PenDig	11K	1.1h 33m	0.00 ± 0.00	0.02 ± 0.02	5.5h 11m	0.01 ± 0.01	0.03 ± 0.02	24m 5m	0.00 ± 0.00	0.05 ± 0.00
Magic	19K	3.4h 1.4h	0.00 ± 0.00	0.02 ± 0.01	7.0h 10m	0.01 ± 0.01	0.03 ± 0.03	1.2h 10m	0.00 ± 0.00	0.06 ± 0.00

4.3. Comparison to workaround in van der Maaten & Hinton (2008)

We compared the efficiency of our approach against another approximation approach (landmark t-SNE) by van der Maaten & Hinton (2008) on the MNIST dataset. The landmark t-SNE also starts from a sparse similarity input, for example k-NN. Differently, they first sample a subset of landmarks from the whole dataset, and then they calculate the similarities among the landmarks by considering multi-path connections, which is implemented with random walk smoothing. Finally a much smaller matrix is fed to t-SNE and only the landmarks will be displayed.

Landmark t-SNE has a number of drawbacks. In application, it cannot visualize the whole dataset. Therefore, it can only discover a macro structure for certain datasets, given that the structure coincides with the blurring distortion by random walk. In terms of computational cost, landmark t-SNT requires solving a very large linear system to find the multipath similarities between the landmarks. In our practice, finding a solution for the MNIST data costs 6-10 hours in a machine with an Intel Core i7 CPU. For even larger datasets, e.g. UCI Covertype and TIMIT in Section 4.4, landmark t-SNE causes out-of-memory failure.

In contrast, the approximated version of t-SNE with SD only takes about 1.6 hours, with small memory cost but an even better visualization (Section 4.4).

4.4. Visualization of large data

We finally demonstrate with the visualizations in Figure 4 that the new fast methods make feasible embedding of very large data sets. Subfigure **A** visualizes 70,000 digit images from MNIST by the approximated t-SNE with SD optimization. The classes are well separated and the plot is almost as good as exact t-SNE in Figure 2 (C3), even though the computation time has been reduced from 46 hours to 1.6 hour. Subfigure **B** shows 58,000 points from the UCI Shuttle data (7 classes of shuttle states) embedded by approximated s-SNE with SD optimization; the classes are well arranged with sharp borders. Subfigure **C** shows our initial results with a fast NeRV by SD optimization on the MNIST data; classes are well arranged although some class overlap is visible. Subfigure **D** shows 581,012 points from the UCI Covertype data (7 classes of forest cover types) embedded by s-SNE with SD; clear structure is visible and classes are locally well arranged although complete class separation is not easy for this challenging data. Subfigure **E** shows 1.3 million points from the DARPA TIMIT acoustic speech database (points are 39ms segments with MFCC features; 49 phoneme classes) embedded by approximated t-SNE with momentum optimization; although full class separation is not easy for the hard TIMIT data, clear structure is again visible and there are several local areas where classes are separated. Especially the visualizations of TIMIT and Covertype would take unfeasibly long with the exact methods.

Overall, the fast methods can embed very large data sets quickly. Even though the methods are unsupervised, the arrangements revealed hidden ground truth classes well. Even more importantly, the methods made visualization possible for huge data sets where the original exact methods would have been infeasible.

4.5. The θ parameter

In the above experiments we fixed $\theta = 2$ in Eq. 5. Here we study the approximation performance using a range of θ values. We used t-SNE and MNIST with $\theta \in [1, 2, \ldots, 20]$. For each θ value, we gener-

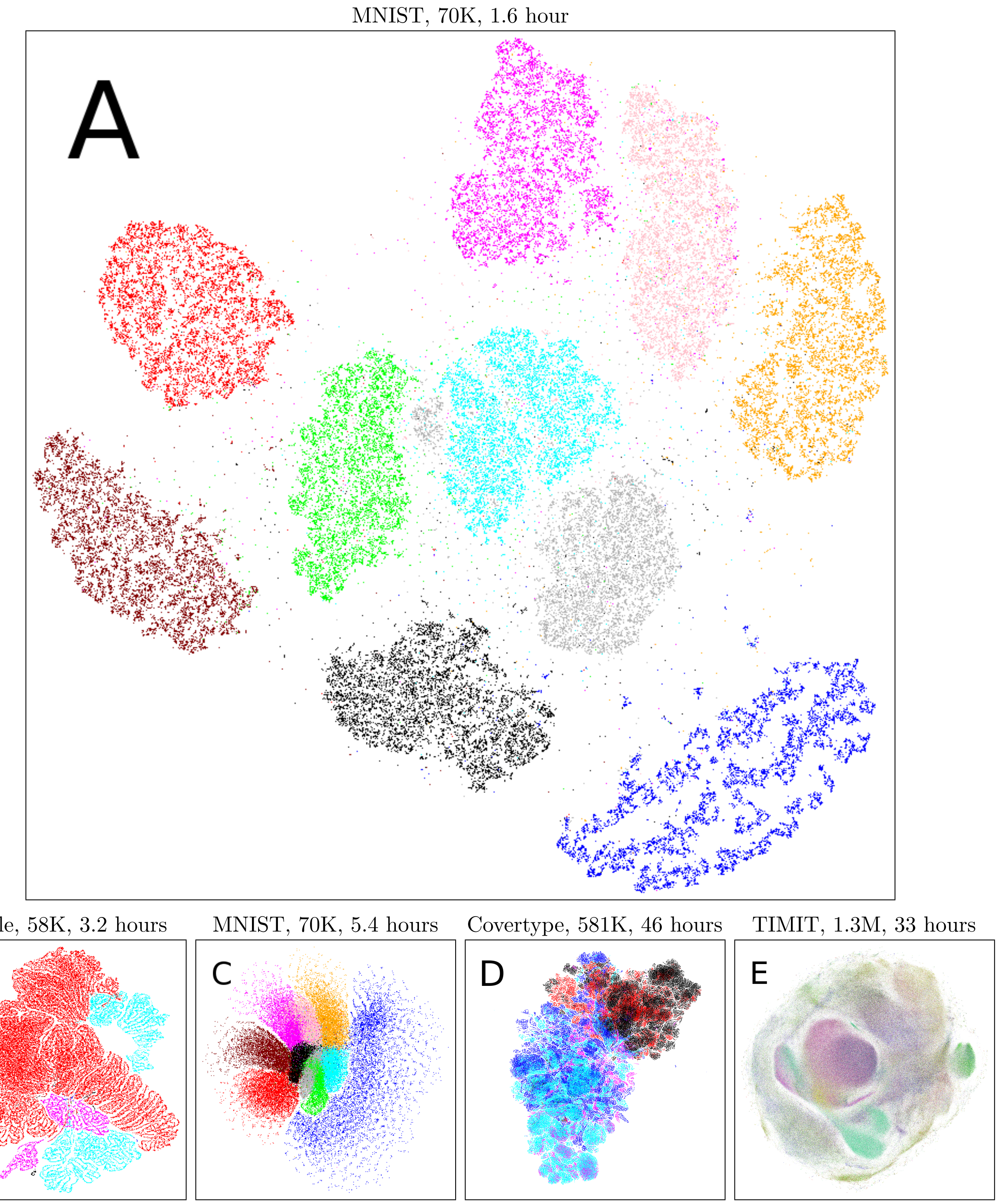

Figure 4. Visualization of large-scale datasets made feasible with the new approximations. A. MNIST using t-SNE and Spectral Direction (SD); B. UCI Shuttle using s-SNE and SD; C. MNIST using NeRV and SD; D. UCI Covertype using s-SNE and SD; and E. TIMIT using t-SNE and momentum. Titles of subfigures show the dataset name, dataset size, and learning time.

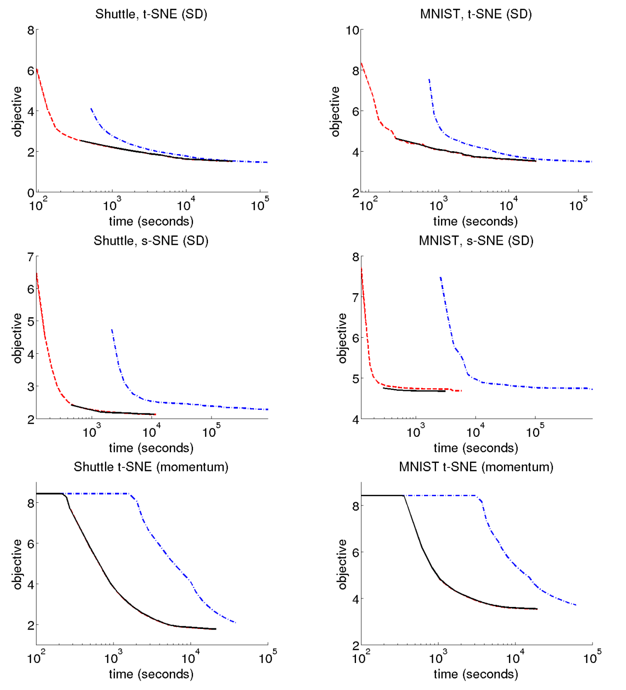

Figure 3. Speedup on three methods (rows) for two large data sets (columns). The curves show the evolution of the objective function value as a function of time for the exact algorithm (dash-dotted line) and approximated algorithm (dashed line). The solid line shows the (off-line computed) value of the exact cost function for the solution produced by the approximated algorithm. For momentum optimization, both dash and solid lines are offline calculated.

ated a 2D mapping by normal distribution with variance uniformly from $[10^{-10}, 10^{10}]$. The resulting approximated objective and gradient is compared with the exact ones. Their relative differences are measured by $\Delta_o = |\mathcal{J}_{\text{exact}} - \mathcal{J}_{\text{approx}}|/|\mathcal{J}_{\text{exact}}|$ and $\Delta_g = \|\nabla_{\text{exact}} - \nabla_{\text{approx}}\|_F / \|\nabla_{\text{exact}}\|_F$, respectively, where ∇ stands for the gradient. We ran the experiment 10 times and recorded the mean and standard deviation.

The results are shown in Figure 5. We can see that relative differences vanish quickly with increasing θ. When $\theta > 5$, more computational cost brings little accuracy improvement. The phenomenon holds for both the objective and gradient. The above experiments indicate that generally a small θ is enough for NE methods such as t-SNE or s-SNE. A relatively larger θ can be used for more complicated methods which require several levels of approximation, for example, NeRV.

5. Conclusions

We have introduced novel fast versions of neighbor embedding methods where interactions to far-away points are approximated at centers-of-mass of a hierarchical quad-tree, yielding fast $O(n \log n)$ complexity for cost function and gradient computation. With this approach we created fast versions of Stochastic Neighbor Embedding (SNE), symmetric SNE, and *t*-distributed SNE; the same approach also yields fast algorithms for Elastic Embedding and the Neighbor Retrieval Visualizer which we did not focus on here.

We demonstrated that embedding large data helps even if only a subset is ultimately plotted, and that the straightforward approach of embedding a subset and bringing in the other data by interpolation does not work well. We showed on several benchmark data sets that our fast methods can embed the full data with minimal difference in quality (measured by cost functions of the original methods and by classification accuracy) and in far less time. We then demonstrated our methods on very large data up to 1.3 million points. Overall, our fast methods make it feasible to visualize "big data" by advanced nonlinear embedding approaches. Our NE software can be found in `http://research.ics.aalto.fi/mi/software/ne/`.

Interestingly, the Barnes-Hut approximation that we used to derive our fast methods has previously been applied in another genre, graph drawing methods, but to our knowledge not for visualizing manifold data. Our results show that state-of-the-art manifold learning methods can greatly benefit from our fast computation approach with very little loss of quality.

The experiments in this work have been performed in a single-core setting. Our method, however, does not prevent distributed implementations using modern parallel computing tools such as GPU or MapReduce. QuadTree as the only centralized component is linear in the number of data points in 2D or 3D. For even larger datasets, one may consider distributed hash tables (e.g. Miller et al., 2010) to de-centralize the tree.

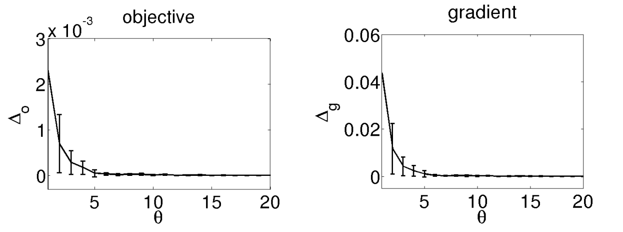

Figure 5. Relative differences between the exact and approximated (left) objectives and (right) gradients across various θ values.

Acknowledgment

Academy of Finland, grants 251170, 140398, and 252845.

References

Barnes, J. and Hut, P. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(4):446–449, 1986.

Beygelzimer, A., Kakade, S., and Langford, J. Cover trees for nearest neighbor. In *International Conference on Machine Learning (ICML)*, pp. 97–104, 2006.

Carreira-Perpiñán, M. The elastic embedding algorithm for dimensionality reduction. In *International Conference on Machine Learning (ICML)*, pp. 167–174, 2010.

Gray, A. and Moore, A. N-body problems in statistical learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 521–527, 2001.

Gray, A. and Moore, A. Rapid evaluation of multiple density models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2003a.

Gray, A. and Moore, A. Nonparametric density estimation: toward computational tractability. In *In SIAM International Conference on Data Mining (ICDM)*, 2003b.

Greengard, L. and Rokhlin, V. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

Hinton, G.E. and Roweis, S.T. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 833–840, 2002.

Liu, T., Moore, A., Gray, A., and Yang, K. An investigation of practical approximate nearest neighbor algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 825–832, 2004.

Miller, F.P., Vandome, A.F., and McBrewster, J. *Distributed Hash Table*. Alphascript Publishing, 2010.

Noack, A. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Venna, J. and Kaski, S. Nonlinear dimensionality reduction as information retrieval. *Journal of Machine Learning Research - Proceedings Track*, 2:572–579, 2007.

Venna, J., Peltonen, J., Nybo, K., Aidos, H., and Kaski, S. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11: 451–490, 2010.

Vladymyrov, M. and Carreira-Perpiñán, M. Partial-hessian strategies for fast learning of nonlinear embeddings. In *International Conference on Machine Learning (ICML)*, pp. 167–174, 2010.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3360–3367, 2010.

Yang, Z., King, I., Xu, Z., and Oja, E. Heavy-tailed symmetric stochastic neighbor embedding. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pp. 2169–2177, 2009.

Yianilos, P. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 311–321, 1993.