COMPUTO

ISSN 2824-7795

Visualizing Data using t-SNE

A practical computo example

Laurens van der Maaten — TiCC, Tilburg University
Geoffrey Hinton — Department of Computer Science, University of Toronto

Abstract

We present a new technique called "t-SNE" that visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map. The technique is a variation of Stochastic Neighbor Embedding hinton:stochastic that is much easier to optimize, and produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map. t-SNE is better than existing techniques at creating a single map that reveals structure at many different scales. This is particularly important for high-dimensional data that lie on several different, but related, low-dimensional manifolds, such as images of objects from multiple classes seen from multiple viewpoints. For visualizing the structure of very large data sets, we show how t-SNE can use random walks on neighborhood graphs to allow the implicit structure of all the data to influence the way in which a subset of the data is displayed. We illustrate the performance of t-SNE on a wide variety of data sets and compare it with many other non-parametric visualization techniques, including Sammon mapping, Isomap, and Locally Linear Embedding. The visualization produced by t-SNE are significantly better than those produced by other techniques on almost all of the data sets.

Keywords: visualization, dimensionality reduction, manifold learning, embedding algorithms, multidimensional scaling

Contents

1 Introduction

Visualization of high-dimensional data is an important problem in many different domains, and deals with data of widely varying dimensionality. Cell nuclei that are relevant to breast cancer, for example, are described by approximately 30 variables (**street:nuclear?**), whereas the pixel intensity vectors used to represent images or the word-count vectors used to represent documents typically have thousands of dimensions. Over the last few decades, a variety of techniques for the visualization of such high-dimensional data have been proposed, many of which are reviewed by (**ferreira:visual?**). Important techniques include iconographic displays such as Chernoff faces (**chernoff:use?**), pixel-based techniques (**keim:designing?**), and techniques that represent the dimensions in the data as vertices in a graph (**battista:algorithms?**). Most of these techniques simply provide tools to display more than two data dimensions, and leave the interpretation of the data to the human observer. This severely limits the applicability of these techniques to real-world data sets that contain thousands of high-dimensional datapoints.

In contrast to the visualization techniques discussed above, dimensionality reduction methods convert the high-dimensional data set $\mathcal{X} = x_1, x_2, \ldots, x_n$ into two or three-dimensional data $\mathcal{Y} = y_1, y_2, \ldots, y_n$ that can be displayed in a scatterplot. In the paper, we refer to the low-dimensional data representation $\mathcal{Y}$ as a map, and to the low-dimensional representations y_i of individual datapoints as map points. The aim of dimensionality reduction is to preserve as much of the significant structure of the high-dimensional data as possible in the low-dimensional map. Various techniques for this problem have been proposed that differ in the type of structure they preserve. Traditional dimensionality reduction techniques such as Principal Components Analysis (**hotelling:analysis?**) and classical multidimensional scaling (**torgerson:multidimensional?**) are linear techniques that focus on keeping the low-dimensional representations of dissimilar datapoints far apart. For high-dimensional data that lies on or near a low-dimensional, non-linear manifold it is usually more important to keep the low-dimensional representations of very similar datapoints close together, which is typically not possible with a linear mapping.

A large number of nonlinear dimensionality reduction techniques that aim to preserve the local structure of data have been proposed, many of which are reviewed by (**lee:nonlinear?**). In particular, we mention the following seven techniques: (1) Sammon mapping (**sammon:nonlinear?**), (2) curvilinear components analysis (**demartines:curvilinear?**), (3) Stochastic Neighbor Embedding (**hinton:stochastic?**); (4) Isomap (**tenenbaum:global?**), (5) Maximum Variance Unfolding (**weinberger:learning?**); (6) Locally Linear Embedding (**roweis:nonlinear?**), and (7) Laplacian Eigenmaps (**belkin:laplacian?**). Despite the strong performance of these techniques on artificial data sets, they are often not very successful at visualizing real, high-dimensional data. In particular, most of the techniques are not capable of retaining both the local and the

global structure of the data in a single map. For instance, a recent study reveals that even a semi-supervised variant of MVU is not capable of separating handwritten digits into their natural clusters (**song:colored?**).

In this paper, we describe a way of converting a high-dimensional data set into a matrix of pairwise similarities and we introduce a new technique, called "t-SNE", for visualizing the resulting similarity data. t-SNE is capable of capturing much of the local structure of the high-dimensional data very well, while also revealing global structure such as the presence of clusters at several scales. We illustrate the performance of t-SNE by comparing it to the seven dimensionality reduction techniques mentioned above on five data sets from a variety of domains. Because of space limitations, most of the $(7 + 1) \times 5 = 40$ maps are presented in the supplemental material, but the maps that we present in the paper are sufficient to demonstrate the superiority of t-SNE.

The outline of the paper is as follows. In Section 2, we outline SNE as presented by (**hinton:stochastic?**), which forms the basis for t-SNE. In Section 3, we present t-SNE, which has two important differences from SNE. In Section Section 4, we describe the experimental setup and the results of our experiments. Subsequently, **?@sec-large-data** shows how t-SNE can be modified to visualize real-world data sets that contain many more than 10,000 datapoints. The results of our experiments are discussed in more detail in **?@sec-discussion**. Our conclusions and suggestions for future work are presented in **?@sec-conclusion**.

2 Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities.[1] The similarity of datapoint x_j to datapoint x_i is the conditional probabilities, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i. For nearby datapoints, $p_{j|i}$ is relatively high, whereas for widely separated datapoints, $p_{j|i}$ will be almost infinitesimal (for reasonable values of the variance of the Gaussian, σ_i). Mathematically, the conditional probability $p_{j|i}$ is given by

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/(2\sigma_i^2))}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)} . \tag{1}$$

where σ_i is the variance of the Gaussian that is centered on datapoint x_i. The method for determining the value of σ_i is presented later in this section. Because we are only interested in modeling pairwise similarities, we set the value of $p_{i|i}$ to zero. For the low-dimensional counterparts y_i and y_j of the high-dimensional datapoints x_i and x_j, it is possible to compute a

[1] SNE can also be applied to data sets that consist of pairwise similarities between objects rather than high-dimensional vector representations of each object, provided these similarities can be interpreted as conditional probabilities. For example, human word associations data consists of the probability of producing each possible word in response to a given word, as a result of which it is already in the form required by SNE.

similar conditional probability, which we denote by $q_{j|i}$. We set [2] the variance of the Gaussian that is employed in the computation of the conditional probabilities $q_{j|i}$ to $\frac{1}{\sqrt{2}}$. Hence, we model the similarity of a map point y_j to map point y_i by

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)} .$$

Again, since we are only interested in modeling pairwise similarities, we set $q_{i|i} = 0$.

If the map points y_i and y_j correctly model the similarity between the high-dimensional datapoints x_i and x_j, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal. Motivated by this observation, SNE aims to find a low-dimensional data representation that minimizes the mismatch between $p_{j|i}$ and $q_{j|i}$. A natural measure of the faithfulness with which $q_{j|i}$ models $p_{j|i}$ is the Kullback-Leibler divergence (which is in the case equal to the cross-entropy up to an additive constant). SNE minimizes the sum of Kullback-Leibler divergences over all datapoints using a gradient descent method. The cost function C is given by

$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} , \tag{2}$$

in which P_i represents the conditional probability distribution over all other datapoints given datapoint x_i, and Q_i represents the conditional probability distribution over all other map points given map point y_i. Because the Kullback-Liebler divergence is not symmetric, different types of error in the pairwise distances in the low-dimensional map are not weighted equally. In particular, there is a large cost for using widely separated map points to represent nearby datapoints (i.e, for using a small $q_{j|i}$ to model a large $p_{j|i}$), but there is only a small cost for using nearby map points to represent widely separated datapoints. This small cost comes from wasting some of the probability mass in the relevant Q distributions. In other words, the SNE cost function focuses on retaining the local structure of the data in the map (for reasonable values of the variance of the Gaussian in the high-dimensional space, σ_i).

The remaining parameter to be selected the variance σ_i of the Gaussian that is centered over each high-dimensional datapoint, x_i. It is not likely that there is a single value of σ_i that is optimal for all datapoints in the data set because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of σ_i induces a probability distribution, P_i, over all of the other datapoints. This distribution has an entropy which increases as σ_i increases. SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by the user[3]. The perplexity is defined as

[2] Setting the variance in the low-dimensional Gaussians to another value only results in a rescaled version of the final map. Note that by using the same variance for every datapoint in the low-dimensional map, we lose the property that the data is a perfect model of itself if we embed it in a space of the same dimensionality, because in the high-dimensional space, we used a different variance σ_i in each Gaussian.

[3] Note that the perplexity increases monotonically with the variance σ_i.

$$Perp(P_i) = 2^{H(P_i)},$$

where $H(P_i)$ is the Shannon entropy of P_i measured in bits

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

The perplexity can be interpreted as a smooth measure of the effective number of neighbors. The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50.

The minimization of the cost function in Equation 2 is performed using a gradient descent method. The gradient has a surprisingly simple form

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

Physically, the gradient may be interpreted as the resultant force created by a set of springs between the map point y_i and all other map points y_j. All springs exert a force along the direction $(y_i - y_j)$. The spring between y_i and y_j repels or attracts the map points depending on whether the distance between the two in the map is too small or too large to represent the similarities between the two high-dimensional datapoints. The force exerted by the spring between y_i and y_j is proportional to its length, and also proportional to its stiffness, which is the mismatch $(p_{j|i} - q_{j|i} + p_{i|j} + q_{i|j})$ between the pairwise similarities of the data points.

The gradient descent is initialized by sampling map points randomly from an isotropic Gaussian with small variance that is centered around the origin. In order to speed up the optimization and to avoid poor local minima, a relatively large momentum term is added to the gradient. In other words, the current gradient is added to an exponentially decaying sum of previous gradients in order to determine the changes in the coordinates of the map points at each iteration of the gradient search. Mathematically, the gradient update with a momentum term is given by

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right),$$

where $\mathcal{Y}^{(t)}$ indicates the solution at iteration t, η indicates the learning rate, and $\alpha(t)$ represents the momentum at iteration t.

In addition, in the early stages of the optimization, Gaussian noise is added to the map points after each iteration. Gradually reducing the variance of this noise performs a type of simulated annealing that helps the optimization to escape from poor local minima in the cost function. If the variance of the noise changes very slowly at the critical point at which the global structure of the map starts to form, SNE tends to find maps with a better global organization. Unfortunately, this requires sensible choices of the initial amount of Gaussian noise and the rate at which it

decays. Moreover, these choices interact with the amount of momentum and the step size that are employed in the gradient descent. It is therefore common to run the optimization several times on a data set to find appropriate values for the parameters.[4] In this respect, SNE is inferior to methods that allow convex optimization and it would be useful to find an optimization method that gives good results without requiring the extra computation time and parameter choices introduced by the simulated annealing.

3 t-Distributed Stochastic Neighbor Embedding

Section 2 discussed SNE as it was presented by (**hinton:stochastic?**). Although SNE constructs reasonably good visualizations, it is hampered by a cost function that is difficult to optimize and by a problem we refer to as the "crowding problem." In this section, we present a new technique called "t-Distributed Stochastic Neighbor Embedding" or "t-SNE" that aims to alleviate these problems. The cost function used by t-SNE differs from the one used by SNE in two ways: (1) it uses a symmetrized version of the SNE cost function with simpler gradients that was briefly introduced by (**cook:visualizing?**) and (2) it uses a Student-t distribution rather than a Gaussian to compute the similarity between two points *in the low dimensional space*. t-SNE employs a heavy-tailed distribution in the low-dimensional space to alleviate both the crowding problem and the optimization problems of SNE.

In this section, we first discuss the symmetric version of SNE (Section 3.1). Subsequently, we discuss the crowding problem (Section 3.2), and the use of heavy-tailed distributions to address this problem (Section 3.3). We conclude the by describing our approach to the optimization of the t-SNE cost function (Section 3.4).

3.1 Symmetric SNE

As an alternative to minimizing the sum of the Kullback-Leibler divergences between the conditional probabilities $p_{j|i}$ and $q_{j|i}$, it is also possible to minimize a single Kullback-Leibler divergence between a joint probability distribution, P, in the high-dimensional space and a joint probability distribution, Q, in the low-dimensional space:

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

where again, we set p_{ij} and q_{ii} to zero. We refer to this type of SNE as symmetric SNE, because it has the property that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ for all i, j. In symmetric SNE, the pairwise similarities in the low-dimensional map q_{ij} are given by

[4] Picking the best map after several runs as visualization of the data is not nearly as problematic as picking the model that does best on a test set during supervised learning. In visualization, the aim is to see the structure in the training data, not to generalize to held out test data.

$$q_{ij} = \frac{\exp(-\|x_i - x_j\|^2)}{\sum_{k \neq l} \exp(-\|x_k - x_l\|^2)} \,. \tag{3}$$

The obvious way to define the pairwise similarities in the high-dimensional space p_{ij} is

$$p_{ij} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma^2)}{\sum_{k \neq l} \exp(-\|x_k - x_l\|^2 / 2\sigma^2)}$$

but this causes problems when a high-dimensional datapoint x_i is an outlier (i.e., all pairwise distances $\|x_i - x_j\|^2$ are large for x_i). For such an outlier, the values of p_{ij} are extremely small for all j, so the location of its low-dimensional map point y_i has very little effect on the cost function. As a result, the position of the map point is not well determined by the positions of the other map points. We circumvent this problem by defining the joint probabilities p_{ij} in the high dimensional space to be symmetrized conditional probabilities, that is, we set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$. This ensures that $\sum_j p_{ij} > \frac{1}{2n}$ for all datapoints x_i, as a result of which each datapoint x_i makes a significant contribution to the cost function. In the low-dimensional space, symmetric SNE simply uses Equation 3. The main advantage of the symmetric version of SNE is the simpler form of its gradient, which is faster to compute. The gradient of symmetric SNE is fairly similar to that of asymmetric SNE, and is given by

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \,.$$

In preliminary experiments, we observed that symmetric SNE seems to produce maps that are just as good as asymmetric SNE, and sometimes even a little better.

3.2 The Crowding Problem

Consider a set of datapoints that lie on a two-dimensional curved manifold which is approximately linear on a small scale, and which is embedded within a higher-dimensional space. It is possible to model the small pairwise distances between datapoints fairly well in a two-dimensional map, which is often illustrated on toy examples such as the "Swiss roll" data set. Now suppose that the manifold has ten intrinsic dimensions[^dataset] and is embedded within a space of much higher dimensionality. There are several reasons why the pairwise distances in a two-dimensional map cannot faithfully model distances between points on the ten-dimensional manifold. For instance, in ten dimensions, it is possible to have 11 datapoints that are mutually equidistant and there is no way to model this faithfully in a two-dimensional map. A related problem is the very different distribution of pairwise distances in the two spaces. The volume of a sphere centered on datapoint i scales as r^m, where r is the radius and m the dimensionality of the sphere. So if the datapoints are approximately uniformly distributed in the region around i on the ten-dimensional manifold, we get the following "crowding problem:" the area of the two-dimensional map that is available to accommodate moderately distant datapoints will not

be nearly large enough compared with the area available to accommodate nearby datapoints. Hence, if we want to model the small distances accurately in the map, most of the points that are at a moderate distance from datapoint i will have to be placed much too far away in the two-dimensional map. In SNE, the spring connecting datapoint i to each of these too-distant map points will thus exert a very small attractive force. Although these attractive forces are very small, the very large number of such forces crushes together the points in the center of the map, which prevents gaps from forming between the natural clusters. Note that the crowding problem is not specific to SNE, but that it occurs in other local techniques for multidimensional scaling such as Sammon mapping.

An attempt to address the crowding problem by adding a slight repulsion to all springs was presented by (**cook:visualizing?**). The slight repulsion is created by introducing a uniform background model with a small mixing proportion, ρ. So however far apart two map points are, q_{ij} can never fall below $\frac{2\rho}{n(n-1)}$ (because the uniform background distribution is over $n(n-1)/2$ pairs). As a result, for datapoints that are far apart in the high-dimensional space, q_{ij} will always be larger than p_{ij}, leading to a slight repulsion. This technique is called UNI-SNE and although it usually outperforms standard SNE, the optimization of the UNI-SNE cost function is tedious. The best optimization method known is to start by setting the background mixing proportion to zero (i.e., by performing standard SNE). Once the SNE cost function has been optimized using simulated annealing, the background mixing proportion can be increased to allow some gaps to form between natural clusters as shown by (**cook:visualizing?**). Optimizing the UNI-SNE cost function directly does not work because two map points that are far apart will get almost all of their q_i from the uniform background. So even if their p_{ij} is large, there will be no attractive force between them, because a small change in their separation will have a vanishingly small *proportional* effect on q_{ij}. This means that if two parts of a cluster get separated early on in the optimization, there is no force to pull them back together.

3.3 Mismatched tails can compensate for mismatched dimensionalities

Since symmetric SNE is actually matching the joint probabilities of pairs of datapoints in the high-dimensional and the low-dimensional spaces rather than their distances, we have a natural way of alleviating the crowing problem that works as follows. In the high-dimensional space, we convert distances into probabilities using a Gaussian distribution. In the low-dimensional map, we can use a probability distribution that has a much heavier tails than a Gaussian to convert distances into probabilities. This allows a moderate distance in the high-dimensional space to be faithfully modeled by a much larger distance in the map and, as a result, it eliminates the unwanted attractive forces between map points that represent moderately dissimilar datapoints.

In t-SNE, we employ a Student t-distribution with a single degree of freedom (which is the same as a Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map. Using this distribution, the joint probabilities q_{ij} are defined as

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{4}$$

We use a Student t-distribution with a single degree of freedom, because it has the particularly nice property that $\left(1 + \|y_i - y_j\|^2\right)^{-1}$ approaches an inverse square law for large pairwise distances $\|y_i - y_j\|$ in the low-dimensional map. This makes the map's representation of joint probabilities (almost) invariant to changes in the scale of the map for map points that are far apart. It also means that large clusters of points that are far apart interact in just the same way as individual points, so the optimization operates in the same way at all but the finest scales. A theoretical justification for our selection of the Student *t*-distribution is that it is closely related to the Gaussian distribution, as the Student *t*-distribution is an infinite mixture of Gaussians. A computationally convenient property is that it is much faster to evaluate the density of a point under a Student *t*-distribution than under a Gaussian because it does not involve an exponential, even though the Student *t*-distribution is equivalent to an infinite mixture of Gaussians with different variances.

The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q (computed using Equation 4) is derived in Appendix A, and is given by

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1} . \tag{5}$$

In **?@fig-gradients**, we show the gradients between the low-dimensional datapoints y_i and y_j as a function of their pairwise Euclidean distances in the high-dimensional and the low-dimensional space (i.e., as a function of $\|x_i - x_j\|$ and $\|y_i - y_j\|$) for the symmetric versions of SNE, UNI-SNE, and t-SNE. In the figures, positive values of the gradient represent an attraction between the low-dimensional datapoints y_i and y_j, whereas negative values represent a repulsion between the two datapoints. From the figures, we observe two main advantages of the t-SNE gradient over the gradients of SNE and UNI-SNE.

First, the t-SNE gradient strongly repels dissimilar datapoints that are modeled by a small pairwise distance in the low-dimensional representation. SNE has such repulsion as well, but its effect is minimal compared to the strong attractions elsewhere in the gradient (the largest attraction in our graphical representation of the gradient is approximately 19, whereas the largest repulsion is approximately 1). In UNI-SNE, the amount of repulsion between dissimilar datapoints is slightly larger, however, this repulsion is only strong when the pairwise distance between the points in the low-dimensional representation is already large (which is often not the case, since the low-dimensional representation is initialized by sampling from a Gaussian with a very small variance that is centered around the origin).

Second, although t-SNE introduces strong repulsions between dissimilar datapoints that are modeled by small pairwise distances, these repulsions do not go to infinity. In this respect, t-SNE differs from UNI-SNE, in which the strength of the repulsion between very dissimilar datapoints is proportional to their pairwise distance in the low-dimensional map, which may cause dissimilar datapoints to move much too far away from each other.

Taken together, t-SNE puts emphasis on (1) modeling dissimilar datapoints by means of large

pairwise distances, and (2) modeling similar datapoints by means of small pairwise distances. Moreover, as a result of these characteristics of the t-SNE cost function (and as a result of the approximate scale invariance of the Student t-distribution), the optimization of the t-SNE cost function is much easier than the optimization of the cost functions of SNE and UNI-SNE. Specifically, t-SNE introduces long-range forces in the low-dimensional map that can pull back together two (clusters of) similar points that get separated early on in the optimization. SNE and UNI-SNE do not have such long-range forces, as a result of which SNE and UNI-SNE need to use simulated annealing to obtain reasonable solutions. Instead, the long-range forces in t-SNE facilitate the identification of good local optima without resorting to simulated annealing

Algorithm 1 Simple version of t-Distributed Stochastic Neighbor Embeding

Data: high-dimensional representation $\mathcal{X} = \{x_1, \dots, x_n\}$
cost function parameters: perplexity $Perp$
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, \dots, y_n\}$
procedure T-SNE($Perp, T, \eta, \alpha(t)$)
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{1}{2n}(p_{j|i} + p_{i|j})$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, \dots, y_n\}$ from $\mathcal{N}(0, 1e^{-4}I)$
    for $t = 0...T$ **do**
        compute low-dimensional affinities q_{ij} (using Equation 4)
        compute gradient $\frac{\partial C}{\partial \mathcal{Y}}$ (using Equation 5)
        et $\mathcal{Y}^t = \mathcal{Y}^{t-1} + \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{t-1} - \mathcal{Y}^{t-2}\right)$

Algorithm 2 Algorithm suggested by Theorem 1.

Data: Sample sizes N and M
Result: Estimation $\hat{E}_N$ of integral E
- Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_M$ on $\mathbb{R}^n$ independently according to g^*
- Estimate $\widehat{\mathbf{m}}^*$ and $\widehat{\Sigma}^*$ defined in Equation 10 and Equation 11 with this sample
- Compute the eigenpairs $(\hat{\lambda}_i^*, \hat{\mathbf{d}}_i^*)$ of $\widehat{\Sigma}^*$ ranked in decreasing ℓ-order
- Compute the matrix $\widehat{\Sigma}_k^* = \sum_{i=1}^k (\hat{\lambda}_i^* - 1)\hat{\mathbf{d}}_i^* (\hat{\mathbf{d}}_i^*)^\top + I_n$ with k obtained by applying Algorithm 2 with input $(\hat{\lambda}_1^*, \dots, \hat{\lambda}_n^*)$
- Generate a new sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ independently from $g' = g_{\widehat{\mathbf{m}}^*, \widehat{\Sigma}_k^*}$
- Return $\hat{E}_N = \frac{1}{N}\sum_{i=1}^N \phi(\mathbf{X}_i)\frac{f(\mathbf{X}_i)}{g'(\mathbf{X}_i)}$

3.4 Optimization methods for t-SNE

We start by presenting a relatively simple, gradient descent procedure for optimizing the t-SNE cost function. This simple procedure uses a momentum term to reduce the number of iterations

required and it works best if the momentum term is small until the map points have become moderately well organized. Pseudocode for this simple algorithm is presented in Algorithm 1 (FIXME: ref not working). The simple algorithm can be sped up using the adaptive learning rate scheme that is described by (**jacobs:rates?**), which gradually increases the learning rate in directions in which the gradient is stable.

Although the simple algorithm produces visualizations that are often much better than those produced by other non-parametric dimensionality reduction techniques, the results can be improved further by using either of two tricks. The first trick, which we call "early compression," is to force the map points to stay close together at the start of the optimization. When the distances between map points are small, it is easy for clusters to move through one another so it is much easier to explore the space of possible global organizations of the data. Early compression is implemented by adding an additional L2-penalty to the cost function that is proportional to the sum of squared distances of the map points from the origin. The magnitude of this penalty term and the iteration at which it is removed are set by hand, but the behavior is fairly robust across variations in these two additional optimization parameters.

A less obvious way to improve the optimization, which we call "early exaggeration," is to multiply all of the p_{ij}'s by, for example, 4, in the initial stages of the optimization. This means that almost all of the q_{ij}'s, which still add up to 1, are much too small to model their corresponding p_{ij}'s. As a result, the optimization is encouraged to focus on modeling the large p_{ij}'s by fairly large qij's. The effect is that the natural clusters in the data tend to form tight widely separated clusters in the map. This creates a lot of relatively empty space in the map, which makes it much easier for the clusters to move around relative to one another in order to find a good global organization.

In all the visualizations presented in this paper and in the supporting material, we used exactly the same optimization procedure. We used the early exaggeration method with an exaggeration of 4 for the first 50 iterations (note that early exaggeration is not included in the pseudocode in Algorithm 1). The number of gradient descent iterations T was set 1000, and the momentum term was set to $\alpha^{(t)} = 0.5$ for $t < 250$ and $\alpha^{(t)} = 0.8$ for $t \geq 250$. The learning rate η is initially set to 100 and it is updated after every iteration by means of the adaptive learning rate scheme described by (**jacobs:rates?**). A Matlab implementation of the resulting algorithm is available at https://lvdmaaten.github.io/tsne/.

4 Experiments

To evaluate t-SNE, we present experiments in which t-SNE is compared to seven other non-parametric techniques for dimensionality reduction. Because of space limitations, in the paper, we only compare t-SNE with: (1) Sammon mapping, (2) Isomap, and (3) LLE. In the supporting material, we also compare t-SNE with: (4) CCA, (5) SNE, (6) MVU, and (7) Laplacian Eigenmaps. We performed experiments on five data sets that represent a variety of application domains. Again, because of space limitations, we restrict ourselves to three data sets in the paper. The results of our experiments on the remaining two data sets are presented in the supplementary material.

In Section 4.1, the data sets that we employed in our experiments are introduced. The setup of the experiments is presented in Section 4.2. In Section 4.3, we present the results of our experiments.

4.1 Data Sets

The five data sets we employed in our experiments are: (1) the MNIST data set, (2) the Olivetti faces data set, (3) the COIL-20 data set, (4) the word-features data set, and (5) the Netflix data set. We only present results on the first three data sets in this section. The results on the remaining two data sets are presented in the supporting material. The first three data sets are introduced below.

The MNIST data set[5] contains 60,000 grayscale images of handwritten digits. For our experiments, we randomly selected 6,000 of the images for computational reasons. The digit images have $28 \times 28 = 784$ pixels (i.e., dimensions). The Olivetti faces data set[6] consists of images of 40 individuals with small variations in viewpoint, large variations in expression, and occasional addition of glasses. The data set consists of 400 images (10 per individual) of size $92 \times 112 = 10,304$ pixels, and is labeled according to identity. The COIL-20 data set (**nene:coil20?**) contains images of 20 different objects viewed from 72 equally spaced orientations, yielding a total of 1,440 images. The images contain $32 \times 32 = 1,024$ pixels.

4.2 Experimental Setup

In all of our experiments, we start by using PCA to reduce the dimensionality of the data to 30. This speeds up the computation of pairwise distances between the datapoints and suppresses some noise without severely distorting the interpoint distances. We then use each of the dimensionality reduction techniques to convert the 30-dimensional representation to a two-dimensional map and we show the resulting map as a scatterplot. For all of the data sets, there is information about the class of each datapoint, but the class information is only used to select a color and/or symbol for the map points. The class information is not used to determine the spatial coordinates of the map points. The coloring thus provides a way of evaluating how well the map preserves the similarities within each class.

The cost function parameter settings we employed in our experiments are listed in Table 1. In the table, *Perp* represents the perplexity of the conditional probability distribution induced by a Gaussian kernel and k represents the number of nearest neighbors employed in a neighborhood graph. In the experiments with Isomap and LLE, we only visualize datapoints that correspond to vertices in the largest connected component of the neighborhood graph. [7] For the Sammon mapping optimization, we performed Newton's method for 500 iterations.

4.3 Results

[5]The MNIST data set is publicly available from http://yann.lecun.com/exdb/mnist/index.html.
[6]The Olivetti data set is publicly available from http://mambo.ucsc.edu/psl/olivetti.html.
[7]Isomap and LLE require data that gives rise to a neighborhood graph that is connected.

Tabl
Cost
func
para
setti
the
men

Technique
t-SNE
Sammon mapping
Isomap
LLE