Learning a Parametric Embedding by Preserving Local Structure

Laurens van der Maaten
TiCC, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands
lvdmaaten@gmail.com

Abstract

The paper presents a new unsupervised dimensionality reduction technique, called parametric t-SNE, that learns a parametric mapping between the high-dimensional data space and the low-dimensional latent space. Parametric t-SNE learns the parametric mapping in such a way that the local structure of the data is preserved as well as possible in the latent space. We evaluate the performance of parametric t-SNE in experiments on three datasets, in which we compare it to the performance of two other unsupervised parametric dimensionality reduction techniques. The results of experiments illustrate the strong performance of parametric t-SNE, in particular, in learning settings in which the dimensionality of the latent space is relatively low.

1 INTRODUCTION

The performance and efficiency of machine learning algorithms is often hampered by the high dimensionality of real-world datasets. Typically, the minimum number of parameters required to account for all properties of the data (i.e., the intrinsic dimensionality) is much smaller than the dimensionality of the data. Dimensionality reduction techniques try to exploit the relatively low intrinsic dimensionality of many real-world datasets. They embed the high-dimensional data in a latent space of lower dimensionality in such a way, that the structure of the data is retained as well as possible. Over the last decade, a large number of new non-parametric dimensionality reduction techniques have been proposed, such as Isomap (Tenenbaum et al., 2000), LLE (Roweis and Saul, 2000), and MVU (Weinberger et al., 2004). The rationale behind these so-called manifold learners is that they attempt to retain the local structure of the data, that is, the small pairwise distances between the datapoints, in the latent space. The main limitation of the non-parametric manifold learners is that they do not provide a parametric mapping between the high-dimensional data space and the low-dimensional latent space, as a result of which the out-of-sample extension for these techniques is non-trivial. For spectral techniques such as the manifold learners listed above, the out-of-sample extension can be realized using the Nyström approximation (Bengio et al., 2004), but this leads to approximation errors and can become computationally expensive. As a result, the lack of a parametric mapping makes non-parametric dimensionality reduction techniques less suitable for use in, e.g., classification or regression tasks.

Despite the recent surge in non-parametric dimensionality reduction techniques, the development of new parametric dimensionality reduction techniques has been limited. Many parametric dimensionality reduction techniques, such as PCA and NCA (Goldberger et al., 2005), are hampered by their linear nature, which makes it difficult to successfully embed highly non-linear real-world data in the latent space. In contrast, autoencoders (Hinton and Salakhutdinov, 2006) can learn the non-linear mappings that are required for such embeddings, but they primarily focus on maximizing the variance of the data in the latent space, as a result of which autoencoders are less successful in retaining the local structure of the data in the latent space than manifold learners.

In this paper, we present a new unsupervised parametric dimensionality reduction technique that attempts to retain the local data structure in the latent space. The new technique, called parametric t-SNE, parametrizes the non-linear mapping between the data space and the latent space by means of a feed-forward neural network. Similar parametrizations have been proposed before, e.g., in NeuroScale (Lowe and Tipping, 1996) and back-constrained GPLVMs (Lawrence and Candela, 2006). The network is trained using a three-stage training procedure that is inspired by the training of autoencoders as described by Hinton and Salakhutdinov

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5. Copyright 2009 by the authors.

(2006). The three-stage training procedure aims to circumvent the problems of backpropagation procedures that are typically used to train neural networks.

The structure of the remainder of this paper is as follows. Section 2 introduces the new unsupervised parametric dimensionality reduction technique, called parametric t-SNE. The result of our experiments with parametric t-SNE on three datasets are presented in Section 3. The results are discussed in more detail in Section 4. Section 5 concludes the paper and presents directions for future research.

2 PARAMETRIC T-SNE

In parametric t-SNE, the parametric mapping $f : X \rightarrow Y$ from the data space X to the low-dimensional latent space Y is parametrized by means of a feed-forward neural network with weights W. We opt for the use of a (deep) neural network, because a neural network with sufficient hidden layers (with non-linear activation functions) is capable of parametrizing arbitrarily complex non-linear functions. The neural network is trained in such a way as to preserve the local structure of the data in the latent space. Herein, the cost function that is minimized in the training of the network is adapted from a recently introduced non-parametric dimensionality reduction technique, called t-SNE, that is good at visualizing the local structure of high-dimensional data (van der Maaten and Hinton, 2008).

The main problem of the training of deep neural networks is that the large number of weights (for typical problems about several millions) in the network cannot be learned successfully using backpropagation, as backpropagation tends to get stuck in poor local minima due to the complex interactions between the layers in the network. In order to circumvent this problem, we use a training procedure that is inspired by the training of autoencoders that is based on Restricted Boltzmann Machines (RBMs). The training procedure consists of three main stages: (1) a stack of RBMs is trained, (2) the stack of RBMs is used to construct a pretrained neural network, and (3) the pretrained network is finetuned using backpropagation as to minimize the cost function that attempts to retain the local structure of the data in the latent space. The training procedure of parametric t-SNE is illustrated in Figure 1. The pretraining (stage 1 and 2) and the finetuning (stage 3) of the parametric t-SNE network are discussed separately in 2.1 and 2.2.

2.1 PRETRAINING

The pretraining of a parametric t-SNE network consists of two stages. First, a stack of RBMs is trained. Second, the stack of RBMs is used to construct a pretrained feed-forward neural network. Below, we first desribe the training of an RBM (in 2.1.1). Subsequently, we turn to the procedure that constructs the pretrained feed-forward neural network (in 2.1.2).

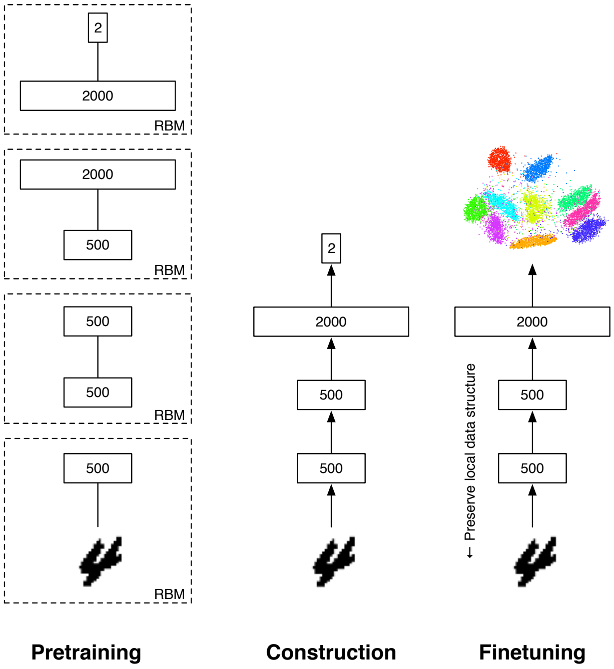


Figure 1: Overview of the three-stage training procedure of a parametric t-SNE network.

2.1.1 Restricted Boltzmann Machine

A Restricted Boltzmann Machine (RBM) is an undirected probabilistic graphical model, i.e., a Markov Random Field. The nodes of an RBM are usually Bernoulli distributed (Hinton, 2002), but if the mean field approximation is employed, the nodes may follow any exponential family distribution (Welling et al., 2004). The structure of an RBM is a fully connected bipartite graph, in which one group of nodes (the visual nodes **v**) models the data, and the other group of nodes (the hidden nodes **h**) models the latent structure of the data.

Since an RBM is a special case of a Markov Random Field, the joint distribution over all nodes is given by a Boltzmann distribution that is specified by the energy function $E(\mathbf{v}, \mathbf{h})$. The most common choice for the energy function is a linear function of the states of the visual and hidden nodes

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j,$$

in which W_{ij} represents the weight of the connection between node v_i and h_j, b_i represents the bias on node v_i, and c_j represents the bias on node h_j. Noting that the states of the visual nodes are conditionally independent given the states of the hidden nodes and vice versa, it can easily be seen[1] that the linear energy function leads to conditional

[1] Note that $p(\mathbf{v}|\mathbf{h}) = \frac{p(\mathbf{v},\mathbf{h})}{\sum_{\mathbf{v}'} p(\mathbf{v}',\mathbf{h})}$, and that if we omit the biases, $p(\mathbf{v}, \mathbf{h}) \propto \exp(\mathbf{h}^T W \mathbf{v})$. Because v_i has a value of either 0 or 1, $p(v_i = 0|\mathbf{h}) = \frac{\exp(0)}{\exp(0)+\exp(\mathbf{h}^T W)} = \frac{1}{1+\exp(\mathbf{h}^T W)}$.

probabilities $P(v_i = 1|\mathbf{h})$ and $P(h_j = 1|\mathbf{v})$ that are given by the sigmoid function of the input into a node

$$P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}, \quad (1)$$

$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - c_j)}. \quad (2)$$

The weights W and the biases $\mathbf{b}$ and $\mathbf{c}$ of an RBM are learned in such a way that the marginal distribution over the visual nodes under the model, $P_{model}(\mathbf{v})$, is close to the observed data distribution $P_{data}(\mathbf{v})$. Specifically, the RBM is trained as to minimize the Kullback-Leibler divergence between the data distribution $P_{data}(\mathbf{v})$ and the model distribution $P_{model}(\mathbf{v})$, which is identical to maximizing the likelihood of the data under the model. The gradient of the Kullback-Leibler divergence with respect to the weights $W_{i,j}$ is given by

$$\frac{\delta KL(P_{data}||P_{model})}{\delta W_{ij}} = \mathbb{E}\left[v_i h_j\right]_{P_{data}} - \mathbb{E}\left[v_i h_j\right]_{P_{model}},$$

where $\mathbb{E}\left[\cdot\right]_{P_{model}}$ represents an expected value under the model distribution, and $\mathbb{E}\left[\cdot\right]_{P_{data}}$ represents an expected value under the data distribution.

Although the form of the gradient is fairly simple, it is impossible to compute the gradient, because the term $\mathbb{E}\left[v_i h_j\right]_{P_{model}}$ cannot be computed analytically. Sampling from the model distribution is also infeasible because this would require the Markov chain to be run infinitely long. In order to alleviate this problem, an alternative gradient has been proposed that minimizes a slightly different objective function that is called the *contrastive divergence* (Hinton, 2002). The constrastive divergence measures the tendency of the model distribution to *walk away* from the data distribution by $KL(P_{data}||P_{model}) - KL(P_1||P_{model})$, where $P_1(\mathbf{v})$ represents the distribution over the visual nodes as the RBM is allowed to run for one iteration (i.e., to perform one Gibbs sweep) when initialized according to the data distribution. The contrastive divergence can be minimized efficiently using standard gradient descent techniques, using an approximate gradient that is given by

$$\mathbb{E}\left[v_i h_j\right]_{P_{data}} - \mathbb{E}\left[v_i h_j\right]_{P_1}.$$

The term $\mathbb{E}\left[v_i h_j\right]_{P_1}$ is now estimated from samples that are obtained using Gibbs sampling (note that the required conditionals are given by Equation 1 and 2). The Markov chain of the sampler may be initialized by clamping a data vector onto the visual nodes, or by using the state of the Markov chain at the previous iteration.

2.1.2 Greedy Layer-Wise Training

The greedy layer-wise training procedure that is used to pretrain the parametric t-SNE network consists of three steps. First, the RBM that corresponds to the first layer is trained on the input data (as described above). Second, the most likely values for the hidden nodes of the RBM are inferred for each datapoint. Third, these values are used as input data to train the RBM that corresponds to the second layer. This process is iterated for all layers in the network. The RBMs that correspond to the bottom layers of the neural network have Bernoulli-distributed hidden units, because this gives rise to a sigmoid activation function in the network. The RBM that corresponds to the top layer of the neural network uses Gaussian distributed hidden units, because this gives rise to a linear activation function in the network. The top layer of a neural network typically has a linear activation function to make the outputs of the network more stable.

The stack of trained RBMs is used to construct a pretrained parametric t-SNE network. Specifically, the undirected weights of the RBMs are untied and the biases on the visible units of the RBMs are dropped. As a result, the stack of RBMs is transformed into a pretrained feed-forward network. The resulting network forms a good initialization for the finetuning stage that aims to preserve the local structure of the data in the latent space (Larochelle et al., 2009). Preliminary experiments revealed that training parametric t-SNE networks without the pretraining stage leads to an inferior performance.

2.2 FINETUNING

In the finetuning stage, the weights of the pretrained neural network are finetuned in such a way that the network retains the local structure of the data in the latent space. This is done by converting the pairwise distances in both the data space and the latent space into probabilities that measure the similarity of two datapoints, and minimizing the Kullback-Leibler divergence between those probabilities (Hinton and Roweis, 2002, Min, 2005, van der Maaten and Hinton, 2008). Specifically, the pairwise distances in the data space are transformed into probabilities by centering an isotropic Gaussian over each datapoint i, computing the density of point j under this Gaussian, and renormalizing, yielding the conditional probabilities $p_{j|i}$

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

The variance of the Gaussian σ_i is set in such a way that the perplexity of each conditional distribution P_i is equal, and $p_{i|i}$ is set to zero. The perplexity is a free parameter that can be thought of as the number of effective neighbors. To form a single joint distribution, the conditional probabilities $p_{j|i}$ are symmetrized[2], i.e., we set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$. The

[2] It is also possible to compute the joint probabilities p_{ij} directly by normalizing over all pairs of datapoints in Equation 2.2, however, such an approach gives inferior results under the presence of outliers.

resulting joint probabilities p_{ij} measure the similarity between datapoints i and j, as a result of which (assuming the variance of the Gaussians is relatively small) they capture the local structure of the data.

To measure the pairwise similarity of datapoints i and j in the latent space, a symmetric distribution is centered over each datapoint i in the latent space as well. Again, the density of all other points j under this distribution is measured, and the result is renormalized to obtain probabilities q_{ij} that represent the local structure of the data in the latent space. The weights of the parametric t-SNE network are now learned in such a way that the Kullback-Leibler divergence between the joint probability distributions P and Q is minimized, i.e., by minimizing

$$C = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{3}$$

The asymmetric nature of the Kullback-Leibler divergence leads the minimization to focus on modeling large p_{ij}'s by large q_{ij}'s. Hence, the objective function focuses on modeling similar datapoints close together in the latent space, as a result of which parametric t-SNE focuses on preserving the local structure of the data.

It may seem logical to use a Gaussian distribution to measure pairwise similarities q_{ij} in the latent space (as is done in, e.g., (Globerson et al., 2007, Hinton, 2002, Min, 2005, Iwata et al., 2007)), but this often leads to inferior results that are due to the *crowding problem* (van der Maaten and Hinton, 2008). The crowding problem is the result of the volume difference between high-dimensional and low-dimensional spaces that we explain in what follows.

Let us suppose that all small pairwise distances are retained perfectly in the low-dimensional latent space. Unless the original data lies in a subspace with an intrinsic dimensionality equal to or smaller than the dimensionality of the latent space, this implies that the larger pairwise distances cannot be modeled well in the latent space. In particular, the large pairwise distances have to be modeled as being larger. As a result, small attractive forces emerge between dissimilar datapoints in the latent space. The large number of such forces cause the crowding problem, as they 'crush' the data representation in the latent space together, which prevents the formation of separations between the natural classes in the data.

The crowding problem can be alleviated by using a heavy-tailed distribution to compute the pairwise similarities q_{ij} in the latent space. The use of a heavy-tailed distribution allows distant points to be modeled as being (too) far apart in the latent space, as a result of which the attractive forces that cause the crowding problem are eliminated. Because of its theoretical relation to the Gaussian distribution, we use a Student-t distribution as the heavy-tailed distribution to measure the pairwise similarities in the latent space. Denoting the mapping from the data space to the latent space that is defined by the feed-forward neural network as $f : X \to Y$, this leads to the following definition of q_{ij}

$$q_{ij} = \frac{\left(1 + \|f(x_i|W) - f(x_j|W)\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{k \neq l} \left(1 + \|f(x_k|W) - f(x_l|W)\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}, \tag{4}$$

where α represents the number of degrees of freedom of the Student-t distribution. We discuss the appropriate setting of α later in this section.

The minimization of the cost function C (that uses the above definition of q_{ij}) can be performed using backpropagation, where the network is initialized using the procedure we described in 2.1. The gradient that is required for the finetuning is given by

$$\frac{\delta C}{\delta W} = \frac{\delta C}{\delta f(x_i|W)} \frac{\delta f(x_i|W)}{\delta W},$$

where $\frac{\delta f(x_i|W)}{\delta W}$ is computed using standard backpropagation, and $\frac{\delta C}{\delta f(x_i|W)}$ is given by

$$\frac{\delta C}{\delta f(x_i|W)} = \frac{2\alpha + 2}{\alpha} \sum_j (p_{ij} - q_{ij}) \left(f(x_i|W) - f(x_j|W)\right) \left(1 + \|f(x_i|W) - f(x_j|W)\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}.$$

Because the number of p_{ij}'s and q_{ij}'s grows quadratically with the number of datapoints in the batch, the minimization of the cost function usually has to be performed using batches of a few thousand points (using larger batches is generally not possible because of memory constraints). The solution is updated after each gradient computation for a batch.

Now that we fully defined parametric t-SNE, we turn to the question of how the number of degrees of freedom α should be set. The Student-t distribution that is used in the latent space may contain a large portion of the probability mass under the distribution, because the volume of the latent space Y grows exponentially with its dimensionality. This leads to problems that may be addressed by setting the degrees of freedom α in such a way as to correct for the exponential growth of the volume of the latent space, because increasing the degrees of freedom α leads to a distribution with lighter tails. In fact, the parameter α determines to what extent the latent space is 'filled up': lower values of α lead to larger separations in the latent space between the natural clusters in the data, because they give rise to stronger repulsive forces between dissimilar datapoints in the latent space. In contrast, higher values of α lead to smaller separations between the natural clusters in the data, as a result of which more space is available in the latent space to appropriately model the local structure of the data. Below, we discuss three approaches to set the degrees of freedom of the Student-t distribution that is used to measure pairwise similarities in the latent space.

1) Fixed value. The first approach is to use a fixed setting of $\alpha = 1$, as is done by van der Maaten and Hinton (2008). This setting is likely to be subject to the problem with the heavy tails discussed above, as a fixed value of α does not correct for the exponential growth of the volume of the latent space (in higher dimensionalities).

2) Linear relation. As the thickness of the tail of a Student-t distribution decreases exponentially with the degrees of freedom α (and the volume of the latent space increases exponentially with the dimensionality of the latent space), it seems likely that the parameter setting for degrees of freedom α should be linearly dependent on the dimensionality d of the latent space. Hence, it seems reasonable to set $\alpha = d - 1$ in order to obtain a single degree of freedom in two-dimensional latent spaces, following van der Maaten and Hinton (2008)).

3) Learning. A possible problem of the second approach is that the appropriate value of α does not only depend on the dimensionality of the latent space. In fact, the most appropriate setting of α depends on the magnitude of the crowding problem, which in turn depends on the ratio between the *intrinsic dimensionality* of the data and the dimensionality of the latent space. For instance, if the intrinsic dimensionality is equal to the dimensionality of the latent space, the crowding problem does not occur at all, and the most appropriate value is thus $\alpha = \infty$ (note that a Student-t distribution with infinite degrees of freedom is equal to a Gaussian distribution). As the intrinsic dimensionality of the data at hand is usually unknown, the third approach treats α as a free parameter that should be optimized with respect to the cost function as well. The required gradient of the cost function C with respect to α is given by

$$\frac{\delta C}{\delta \alpha} = \sum_{i \neq j} \left(\frac{(-\alpha - 1) d_{ij}^2}{2\alpha^2 \left(1 + \frac{d_{ij}^2}{\alpha}\right)} + \frac{1}{2} \log \left(1 + \frac{d_{ij}^2}{\alpha}\right) \right) (p_{ij} - q_{ij}) \,,$$

where d_{ij}^2 represents $\|f(x_i|W) - f(x_j|W)\|^2$. In the following section, we present experiments in which we used three different approaches for setting α.

3 EXPERIMENTS

In order to evaluate the performance of parametric t-SNE and to compare the three different settings for its parameter α, we performed experiments with parametric t-SNE on three datasets. The setup of these experiments is discussed in 3.1. The results of the experiments are presented in 3.2.

3.1 EXPERIMENTAL SETUP

We performed experiments on three datasets: (1) the MNIST dataset, (2) the characters dataset, and (3) the 20 newsgroups dataset. The MNIST dataset contains $70,000$ images of handwritten digits of size 28×28 pixels. The dataset has a fixed division into $60,000$ training images and held out $10,000$ test images. The characters dataset consists of 40,121 grayscale images of handwritten upper-case characters and numerals of size 90×90 pixels, of which we used $35,000$ images as training data and the remainder as test data. The characters dataset comprises 35 classes, viz., 10 numeric classes and 25 alpha classes (the character 'X' is missing in the dataset). The 20 newsgroups dataset contains 100-dimensional binary word-occurence features for $16,242$ documents gathered from 20 different newsgroups. We used $15,000$ documents as training data, and the remaining documents as test data.

In our experiments, we compared parametric t-SNE with two other unsupervised parametric techniques for dimensionality reduction, viz., PCA and multilayer autoencoders (Hinton and Salakhutdinov, 2006). We also compared parametric t-SNE to NCA (Goldberger et al., 2005), which is a supervised linear dimensionality reduction technique. We evaluated the performance of the techniques by means of plotting two-dimensional visualizations, measuring generalization performances of nearest-neighbor classifiers, and evaluating the trustworthiness (Venna and Kaski, 2006) of the low-dimensional embeddings[3]. In order to make the comparison between parametric t-SNE and autoencoders as fair as possible, we used the same layout for both neural networks (where it should be noted that a parametric t-SNE network does not have the decoder part of an autoencoder). Motivated by the experimental setup employed by Salakhutdinov and Hinton (2007), we used $28 \times 28 - 500 - 500 - 2000 - d$ parametric t-SNE networks and autoencoders in our experiments on the MNIST dataset (where d represents the dimensionality of the latent space). In our experiments on the characters dataset, we used $90 \times 90 - 500 - 500 - 2000 - d$ networks. On the 20 newsgroups dataset, we used $100 - 150 - 150 - 500 - d$ networks. The autoencoders were trained using the same three-stage training approach as parametric t-SNE, but the autoencoder is finetuned by performing backpropagation as to minimize the sum of squared errors between the input and the output of the autoencoder (see (Hinton and Salakhutdinov, 2006) for details).

We used exactly the same procedure and parameter settings in the pretraining of the parametric t-SNE networks and the autoencoders. In the training of the RBMs whose hidden units have sigmoid activation functions (the RBMs in the first three layers), the learning rate is set to 0.1 and the

[3] The trustworthiness expresses to what extent the local structure of data is retained in a low-dimensional embedding in a value between 0 and 1. Mathematically, it is defined as $T(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in U_i^{(k)}} (r(i,j) - k)$, where $r(i,j)$ represents the rank of the low-dimensional datapoint j according to the pairwise distances between the low-dimensional datapoints, and $U_i^{(k)}$ represents the set of points that are among the k nearest neighbors in the low-dimensional space but not in the high-dimensional space (Venna and Kaski, 2006).

weight decay is set to 0.0002. The training of the RBMs with a linear activation function in the hidden units (the RBMs in the fourth layer) is performed using a learning rate of 0.01 and a weight decay of 0.0002. In the training of all RBMs, the momentum is set to 0.5 for the first five iterations, and to 0.9 afterwards. The RBMs are all trained using 50 iterations of contrastive divergence with one complete Gibbs sweep per iteration.

Both parametric t-SNE and the autoencoders were finetuned using 30 iterations of backpropagation using conjugate gradients on batches of $5,000$ datapoints. The subdivision of training data into batches was fixed in order to facilitate the precomputation of the P matrices that are required in parametric t-SNE. In the experiments with parametric t-SNE, the variance σ_i of the Gaussian distributions was set such that the perplexity of the conditional distributions P_i was equal to 30. NCA was trained by running conjugate gradients on batches of $5,000$ datapoints for 10 iterations using conjugate gradients.

3.2 RESULTS

In Figure 2, we present the visualizations of the MNIST dataset that were constructed by PCA, an autoencoder, and a parametric t-SNE network (using $\alpha = 1$). The visualizations were constructed by transforming the MNIST test images, that were held out during training, to two dimensions using the trained models. The results reveal the strong performance of parametric t-SNE compared to PCA and autoencoders. In particular, the PCA visualization mixes up most of the natural classes in the data. The autoencoder outperforms PCA, but cannot successfully separate the classes 4, 9, 6, and 8. In contrast, parametric t-SNE clearly separates all classes (although the visualization contains some debris that is mainly due to the presence of distorted digits in the data). In Table 1, we present the generalization errors of 1-nearest neighbor classifiers that were trained on the low-dimensional representations obtained from the three parametric dimensionality reduction techniques (using three different dimensionalities for the latent space). The generalization errors were measured on test data that was held out during the training of both the dimensionality reduction techniques and the classifiers. The corresponding trustworthinesseses $T(12)$ of the embeddings are presented in Table 2. In both tables, the best performance in each experiment is typeset in boldface. From the results presented in Table 1 and 2, we can make the following two observations.

First, we observe that parametric t-SNE performs better or on par with the other techniques in all experiments. In particular, the performance of parametric t-SNE is very strong if the dimensionality of the latent space is not large enough to accomodate for all properties of the data. In this case, the heavy tails of the distribution of parametric t-SNE in the latent space push the natural clusters in the data apart, whereas PCA and autoencoders construct embeddings in which these natural clusters (partially) overlap. The high trustworthinesses of the parametric t-SNE embeddings indicate that parametric t-SNE preserves the local structure of the data in the latent space well. The results reveal that parametric t-SNE also outperforms linear NCA, even though NCA has the advantage of being fully supervised.

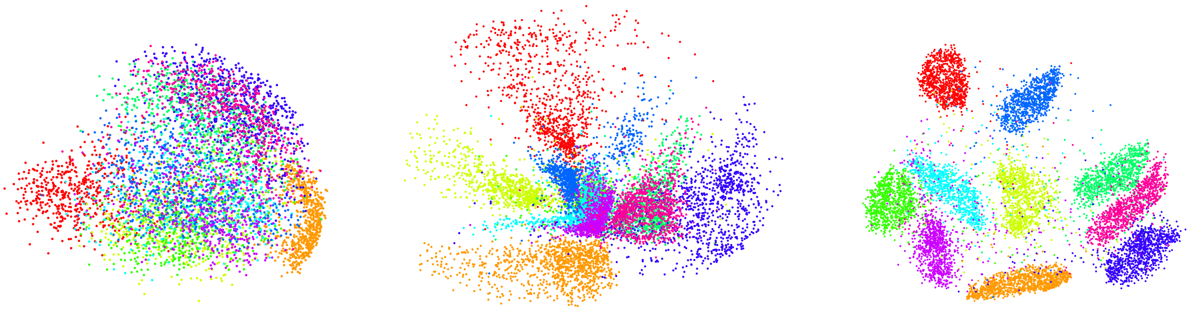
Second, we observe that it is disadvantageous to use a single degree of freedom in the latent space if that latent space has more than, say, two dimensions. Our results reveal that it is better to use make the number of degrees of freedom α linearly dependent on the dimensionality of the latent space d, for reasons we already explained in 2.2. The results also show that learning the appropriate number of degrees of freedom α leads to similar results. The learned value of α was usually slightly smaller than $d-1$ in our experiments.

4 DISCUSSION

From the results of our experiments, we observe that parametric t-SNE often outperforms two other unsupervised parametric dimensionality reduction techniques, in particular, if the dimensionality of the latent space is relatively low. These results are due to the main differences of parametric t-SNE compared to PCA and autoencoders, which we discuss below.

The strong performance of parametric t-SNE compared to PCA can be explained from the two main problems of PCA. First, the linear nature of PCA is too restrictive for the technique to find appropriate embeddings for non-linear real-world data. Second, PCA focuses primarily on retaining large pairwise distances in the latent space (which can be understood from its relation to classical scaling), whereas it is more important to retain the local structure of the data in the latent space.

The strong performance of parametric t-SNE compared to autoencoders, especially if the latent space has a relatively low dimensionality, can be understood from the following difference between parametric t-SNE and autoencoders. Parametric t-SNE aims to model the local structure of the data appropriately in the latent space, and it attempts to create separation between the natural clusters in the data (by means of the heavy-tailed distribution in the latent space). In contrast, autoencoders mainly aim to maximize the variance of the data in the latent space, in order to achieve low reconstruction errors. As a result of the maximization of the variance, autoencoders generally do not construct low-dimensional data representations in which the natural classes in the data are widely separated (as this would decrease the variance of the low-dimensional data representation, and increase the reconstruction error). The relatively poor separation between natural classes in low-dimensional data representations constructed by autoencoders leads to inferior generalization performance of nearest neighbors classifiers compared to parametric t-SNE, in



(a) Visualization by PCA.

(b) Visualization by an autoencoder.

(c) Visualization by parametric t-SNE.

Figure 2: Visualizations of $10,000$ digits from the MNIST dataset by parametric dimensionality reduction techniques.

	MNIST			Characters			20 Newsgroups		
	2D	10D	30D	2D	10D	30D	2D	10D	30D
PCA	78.16%	43.03%	10.78%	86.72%	60.73%	20.50%	35.99%	27.05%	28.82%
NCA	56.84%	8.84%	7.32%	72.90%	24.68%	17.95%	**30.76%**	26.65%	26.09%
Autoencoder	66.84%	6.33%	**2.70%**	82.93%	17.91%	**11.11%**	37.60%	29.15%	27.62%
Par. t-SNE, $\alpha = 1$	**9.90%**	5.38%	5.41%	**43.90%**	26.01%	23.98%	34.30%	**24.40%**	24.88%
Par. t-SNE, $\alpha = d - 1$	**9.90%**	**4.58%**	2.76%	**43.90%**	**17.13%**	13.55%	35.10%	25.28%	**23.75%**
Par. t-SNE, learned α	12.68%	4.85%	**2.70%**	44.78%	17.30%	14.31%	33.82%	27.21%	24.72%

Table 1: Generalization errors of 1-nearest neighbor classifiers on low-dimensional representations of the MNIST dataset, the characters dataset, and the 20 newsgroups dataset.

particular, if the dimensionality of the latent space is relatively low. Moreover, parametric t-SNE provides computational advantages over autoencoders. An autoencoder consists of an encoder part and a decoder part, whereas parametric t-SNE only employs an encoder network. As a result, errors have to be backpropagated through half the number of layers in parametric t-SNE (compared to autoencoders), which gives it an computational advantage over autoencoders (even though the computation of the errors is somewhat more expensive in parametric t-SNE).

A notable advantage of autoencoders is that they provide the capability to reconstruct the original data from its low-dimensional representation in the latent space. In other words, autoencoders do not only provide a parametric mapping from the data space to the latent space, but also the other way around. A possible approach to address this shortcoming is to use the decoder part of an autoencoder as a regularizer on the parametric t-SNE network, i.e., to minimize a weighted sum of Equation 3 and the reconstruction error (as is done for non-linear NCA by Salakhutdinov and Hinton (2007)).

As the number of parameters in parametric t-SNE and autoencoders is larger than in PCA, these techniques are likely to be more susceptible to overfitting. However, we did not observe overfitting effects in our experiments, probably because of the relatively large number of instances in our training data. If parametric t-SNE or autonencoders are trained on smaller datasets, it may be necessary to use early stopping (Caruana et al., 2001).

The results of our experiments not only reveal the strong performance of parametric t-SNE compared to PCA and autoencoders, but also provide insight into the nature of the crowding problem. In particular, the results reveal that the severity of the crowding problem depends on the ratio between the intrinsic dimensionality of the data and the dimensionality of the latent space. The number of degrees of freedom α should thus be set accordingly. We suggested to treat α as a parameter that has to be learned as well, and although competitive, learning α does not always outperform a setting in which α depends linearly on the dimensionality of the latent space. Presumably, this observation is due to the following. When α is learned, it is set in such a way as to 'fill up' the latent space. This decreases the Kullback-Leibler divergence that parametric t-SNE minimizes, because it provides more space to model the local structure of the data appropriately (recall that the cost function focuses on retaining local structure). Although the 'filling up' of the space is advantageous for modeling the local structure of the data (as is illustrated by the high trustworthinesses when α is learned), it has a negative influence on the generalization performance of nearest neighbor classifiers on the low-dimensional data representation, as it decreases the separation between the natural clusters in the data.

	MNIST			Characters			20 Newsgroups		
	2D	10D	30D	2D	10D	30D	2D	10D	30D
PCA	0.744	0.991	0.998	0.735	0.971	0.994	0.634	0.847	0.953
NCA	0.721	0.968	0.971	0.721	0.935	0.957	0.633	0.705	0.728
Autoencoder	0.729	0.996	**0.999**	0.721	0.976	0.992	0.612	0.856	**0.961**
Par. t-SNE, $\alpha = 1$	0.926	0.983	0.983	**0.866**	0.957	0.959	0.720	0.854	0.866
Par. t-SNE, $\alpha = d - 1$	**0.927**	**0.997**	**0.999**	**0.866**	**0.988**	**0.995**	0.714	**0.864**	0.942
Par. t-SNE, learned α	0.921	0.996	**0.999**	0.861	**0.988**	**0.995**	**0.722**	0.857	0.941

Table 2: Trustworthiness $T(12)$ of low-dimensional representations of the MNIST dataset, the characters dataset, and the 20 newsgroups dataset.

5 CONCLUSIONS

We have shown how a deep feed-forward neural network can be trained that reduces the dimensionality of data, while preserving its local structure. The results of our experiments with parametric t-SNE on three datasets showed thatit outperforms other unsupervised parametric dimensionality reduction techniques such as autoencoders. A Matlab implementation of parametric t-SNE is available from `http://ticc.uvt.nl/~lvdrmaaten/tsne`. In future work, we aim to investigate parametric t-SNE networks that use the decoder part of an autoencoder as a regularizer. Also, we aim to investigate how parametric t-SNE can be combined with supervised dimensionality reduction techniques to obtain better generalization performances in semi-supervised learning settings.

Acknowledgements

The author thanks Geoffrey Hinton for many helpful discussions, and Eric Postma for his comments. Laurens van der Maaten is supported by the Netherlands Organization for Scientific Research, project RICH (grant 640.002.401).

References

Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems*, volume 16, pages 177–184, 2004.

R. Caruana, S. Lawrence, and L. Giles. Overtting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances of Neural Information Processing Systems*, volume 13, pages 402–408, 2001.

A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 8:2265–2295, 2007.

J. Goldberger, S. Roweis, G.E. Hinton, and R.R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, volume 17, pages 513–520, 2005.

G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

G.E. Hinton and S.T. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840, 2002.

G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

T. Iwata, K. Saito, N. Ueda, S. Stromsten, T.L. Griffiths, and J.B. Tenenbaum. Parametric embedding for class visualization. *Neural Computation*, 19(9):2536–2556, 2007.

H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10(Jan):1–40, 2009.

N.D. Lawrence and J. Quiñonero Candela. Local distance preservation in the GP-LVM through back constraints. In *Proceedings of the International Conference on Machine Learning*, pages 513–520, 2006.

D. Lowe and M. Tipping. Feed-forward neural networks and topographic mappings for exploratory data analysis. *Neural Computing and Applications*, 4(2):83–95, 1996.

R. Min. A non-linear dimensionality reduction method for improving nearest neighbour classification. Master's thesis, University of Toronto, Canada, 2005.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

R.R. Salakhutdinov and G.E. Hinton. Learning a non-linear embedding by preserving class neighbourhood structure. In *Proceedings of the 11^{th} International Conference on Artificial Intelligence and Statistics*, pages 412–419, 2007.

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

L.J.P. van der Maaten and G.E. Hinton. Visualizating data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *Proceedings of the 14^{th} European Symposium on Artificial Neural Networks*, pages 557–562, 2006.

K.Q. Weinberger, F. Sha, and L.K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the 21^{st} International Conference on Machine Learning*, pages 839–846, 2004.

M. Welling, M. Rosen-Zvi, and G. Hinton. Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems*, volume 17, pages 1481–1488, 2004.