Deep Supervised t-Distributed Embedding

Renqiang Min MINRQ@CS.TORONTO.EDU
Department of Computer Science, University of Toronto

Laurens van der Maaten LVDMAATEN@GMAIL.COM
Department of Computer Science and Engineering, University of California, San Diego
Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology

Zineng Yuan ZINENG.YUAN@UTORONTO.CA
Department of Molecular Genetics, University of Toronto

Anthony Bonner BONNER@CS.TORONTO.EDU
Department of Computer Science, University of Toronto

Zhaolei Zhang ZHAOLEI.ZHANG@UTORONTO.CA
Banting and Best Department of Medical Research, University of Toronto

Abstract

Deep learning has been successfully applied to perform non-linear embedding. In this paper, we present supervised embedding techniques that use a deep network to collapse classes. The network is pre-trained using a stack of RBMs, and finetuned using approaches that try to collapse classes. The finetuning is inspired by ideas from NCA, but it uses a Student t-distribution to model the similarities of data points belonging to the same class in the embedding. We investigate two types of objective functions: deep t-distributed MCML (dt-MCML) and deep t-distributed NCA (dt-NCA). Our experiments on two handwritten digit data sets reveal the strong performance of dt-MCML in supervised parametric data visualization, whereas dt-NCA outperforms alternative techniques when embeddings with more than two or three dimensions are constructed, e.g., to obtain good classification performances. Overall, our results demonstrate the advantage of using a deep architecture and a heavy-tailed t-distribution for measuring pairwise similarities in supervised embedding.

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

1. Introduction

Given the class information of training data points, linear feature transformations and linear dimensionality reductions have been widely applied to perform high-dimensional data embedding either for data visualization or for k-NN classification. Various techniques have been proposed that learn a linear transformation or Mahalanobis metric that tries to decrease the pairwise distances between data points with the same class, while increasing the separation between data points with dissimilar classes (Globerson & Roweis, 2006; Goldberger et al., 2005; Weinberger et al., 2006; Xing et al., 2003). In other words, the techniques aim to collapse classes in a low-dimensional embedding[1].

However, making data points that correspond the same class collapse cannot always be achieved simple linear transformations, especially when the data consists of one or more complex nonlinear (sub)manifolds. Hence, we need to resort to more powerful non-linear transformations. Recent advances in the training of deep networks provide a way to model non-linear transformations of data. Such deep networks are typically pre-trained using, e.g., a stack of Restricted Boltzmann Machines (RBMs; Hinton et al. (2006)) or denoising autoencoders (Larochelle et al., 2007). Subsequently, the pre-trained networks are finetuned as to,

[1]In techniques that learn a Mahalanobis metric, this embedding can be computed by projecting the data onto the eigenvectors of the metric $\mathbf{M}$ (weighted by the square-root of the corresponding eigenvalues).

e.g., construct a generative model of the data (Hinton et al., 2006), learn a classifier (Larochelle et al., 2007), increase the separation between classes (Min et al., 2009; Salakhutdinov & Hinton, 2007; Weston et al., 2008), or preserve pairwise similarities between data points (van der Maaten, 2009).

In this paper, we present two supervised embedding techniques, called deep t-distributed MCML (dt-MCML) and deep t-distributed NCA (dt-NCA), that use a deep feedforward neural network to model the transformation from the high-dimensional to the low-dimensional space. The networks are pre-trained using a stack of RBMs, and finetuned by minimizing objective functions that aim to collapse classes. The objective functions are inspired by the objective functions of Maximally Collapsing Metric Learning (MCML; Globerson & Roweis (2006)) and Neighborhood Components Analysis (NCA; Goldberger et al. (2005)), but they use a Student t-distribution to measure the similarities for pairwise data points in low-dimensional space. The advantage of using a t-distribution to measure these pairwise similarities is four-fold: (1) due to the heavier tail of the distribution, it works better on data sets in which (some of) the class distributions are multimodal, (2) due to the more peaked mode of the distribution, it leads to tighter clusters, i.e., it collapses classes better, (3) the heavier tails lead to more separation between classes, and (4) due to the steeper gradient along the tail of the t-distribution, the optimization using gradient descent is easier.

We performed experiments with dt-MCML and dt-NCA on the USPS and MNIST handwritten digit data sets. The results of the experiments reveal the strong performance of both techniques. In particular, dt-MCML performs very well in learning settings in which the dimensionality of the latent space is very low, e.g., when performing data visualization. In contrast, if the dimensionality of the latent space is higher than two, dt-NCA obtains a superior performance.

The outline of this paper is as follows. In Section 2, we introduce the two new embedding techniques, called dt-MCML and dt-NCA. In Section 3, we present the results of our experiments with the new techniques on two data sets. Section 4 concludes the paper.

2. Deep Supervised Embedding

In this section, we present the two new techniques for supervised parametric embedding. We present dt-MCML in Section 2.1, and dt-NCA in Section 2.2.

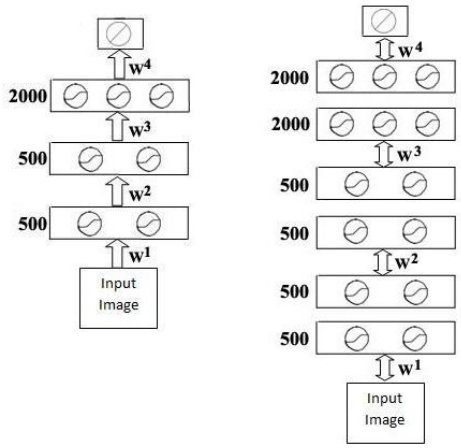

Figure 1. The deep network used in dt-MCML and dt-NCA (left) and an illustration of the pre-training of the deep neural network (right). Adopted from Min et al. (2009).

2.1. Deep t-distributed MCML

Suppose we are given a set of high-dimensional data points and their corresponding labels $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)} : i = 1, \ldots, n\}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^D$, and $y^{(i)} \in \{1, \ldots, c\}$, where c is the total number of classes. MCML learns a Mahalanobis metric $\mathbf{M}$ that aims to simultaneously achieve two main goals: (1) it tries to maximize the sum of the pairwise similarities of data points with the same class, and (2) it tries to minimize the sum of the pairwise similarities of data points with dissimilar classes. In fact, learning a Mahalanobis metric is identical to learning a linear mapping $\mathbf{A}$ of the data: the linear mapping $\mathbf{A}$ is given by the eigenvectors of the Mahalanobis metric $\mathbf{M}$ (weighted by the square-root of their corresponding eigenvalues). MCML can thus be thought of as learning a function $f(\mathbf{x}^{(i)}) = \mathbf{A}\mathbf{x}^{(i)}$ that transforms the high-dimensional data points to a latent space with d dimensions (i.e., $\mathbf{A}$ is a $d \times D$ matrix, where typically, $d < D$).

MCML measures the pairwise similarity of the transformed data point $f(\mathbf{x}^{(i)})$ and the transformed data point $f(\mathbf{x}^{(j)})$ using a stochastic neighborhood criterion. In other words, it centers a Gaussian distribution over $f(\mathbf{x}^{(i)})$, measures the density of $f(\mathbf{x}^{(j)})$ under this Gaussian, and renormalizes:

$$q_{j|i} = \frac{\exp(-d_{ij}^2)}{\sum_{k:k\neq i} \exp(-d_{ik}^2)}, \quad q_{i|i} = 0, \tag{1}$$

where we defined:

$$d_{ij}^2 = ||f(\mathbf{x}^{(i)}) - f(\mathbf{x}^{(j)})||^2.$$

The probabilities $q_{j|i}$ can be viewed as the probability

that the point $f(\mathbf{x}^{(i)})$ picks point $f(\mathbf{x}^{(j)})$ as its nearest neighbor in the latent space. MCML tries to minimize the sum of the Kullback-Leibler divergences between the conditional probabilities $q_{j|i}$ and "ground-truth" probabilities $p_{j|i}$ that are defined based on the class labels of the data. Specifically, MCML defines $p_{j|i} \propto 1$ iff $y^{(i)} = y^{(j)}$, and $p_{j|i} = 0$ iff $y^{(i)} \neq y^{(j)}$. The cost function that MCML minimizes is given by:

$$\ell_{MCML} = \sum_i KL(P_i||Q_i) = \sum_i \sum_{j:j\neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

In deep t-distributed MCML, we follow a similar approach as in MCML, however, we introduce three major changes. First, instead of parametrizing the function f by means of a linear mapping $\mathbf{A}$, we define the function f to be a feedforward neural network mapping that is parametrized by the weights of the network W. Second, instead of measuring the similarities of the transformed data points by means of a Gaussian density, we measure densities under a Student-t distribution. Third, we change the normalization of the pairwise similarities, as this leads to significant simplifications in the gradient. Mathematically, we define:

$$q_{ij} = \frac{(1 + d_{ij}^2/\alpha)^{-\frac{1+\alpha}{2}}}{\sum_{kl:k\neq l}(1 + d_{kl}^2/\alpha)^{-\frac{1+\alpha}{2}}}, \quad q_{ii} = 0, \qquad (2)$$

where α is represents the number of degrees of freedom of the Student-t distribution. The reader should note that the following special cases of the t-distribution: when $\alpha = 1$, the t-distribution is a Cauchy distribution, whereas when $\alpha = \infty$, the t-distribution is a Gaussian distribution. The cost function is now defined to be:

$$\ell_{dt-MCML} = KL(P||Q) = \sum_i \sum_{j:j\neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

where $p_{ij} \propto 1$ iff $y^{(i)} = y^{(j)}$, $p_{ij} = 0$ iff $y^{(i)} \neq y^{(j)}$, and $\sum_{ij} p_{ij} = 1$.

In the definition of q_{ij} above, the pairwise distance d_{ij}^2 is defined similarly as in NCA, except for that the transformation function f is not linear anymore. In particular, the function $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$ is a nonlinear function that is defined by a feed-forward neural network with weights W. The advantage of using a deep network to parametrize the function f instead of a linear mapping is that a deep network is better at learning the complex nonlinear function that is presumably required to collapse classes in the latent space, in particular, when the data consists of very complex nonlinear (sub)manifolds.

The use of a Student-t distribution in the pairwise similarities in Equation 2 has four main advantages over the use of a Gaussian distribution (as in Equation 1). First, the heavy tails of the Student-t distribution make the cost function somewhat more happy when groups of dissimilar points with the same class are modeled far apart in the latent space. This is beneficial in cases in which the distribution of one or more classes is bimodal or multimodal: collapsing data points from the different modes onto a single mode in the latent space is bound to lead to severe overfitting. Second, the peaked mode of the t-distribution (compared to a Gaussian distribution) leads the cost function to favor solutions in which similar points with the same class are modeled closer together, i.e., it leads to tighter clusters in the embedding. Third, the use of t-distribution forces points with different classes to be further apart in the latent space in order for their similarity to become infinitesimal. Fourth, the use of Student-t distributions makes the gradient optimization easier, because the gradient of the tail of a Student t-distribution is much steeper than that of a Gaussian distribution. As a result, the t-distribution provides more "long-range forces", which makes it easier to collapse groups of points with the same class that are separated at some point in the optimization.

The gradient of the cost function of dt-MCML with respect to the location of a latent point $f(\mathbf{x}^{(i)})$ is given by:

$$\frac{\partial \ell_{dt-MCML}}{\partial f(\mathbf{x}^{(i)})} = \sum_{j:j\neq i} \frac{2(\alpha+1)}{\alpha} \left(1 + \frac{d_{ij}^2}{\alpha}\right)^{-\frac{1+\alpha}{2}} (p_{ij} - q_{ij})(f(\mathbf{x}^{(i)}) - f(\mathbf{x}^{(j)})).$$

Using the gradient above, the gradients of the cost function with respect to the weights W of the neural network can be computed using standard backpropagation. Although it is possible to treat the degrees of freedom α as a parameter, we can also try to learn its optimal value using gradient descent. The required gradient is given by:

$$\frac{\partial \ell_{dt-MCML}}{\partial \alpha} = \sum_{ij:i\neq j} (p_{ij} - q_{ij}) \left(\frac{1}{2} \log\left(1 + \frac{d_{ij}^2}{\alpha}\right) - \frac{(1+\alpha)d_{ij}^2}{2\alpha^2(1 + \frac{d_{ij}^2}{\alpha})}\right).$$

We should note that for visualization purposes, a value of $\alpha = 1$ usually performs very well (van der Maaten, 2009), and learning α is superfluous.

2.2. Deep t-distributed NCA

Collapsing classes typically works well for data visualization tasks (Globerson & Roweis, 2006), but col-

lapsing classes is unnecessary to obtain low nearest neighbor errors after high-dimensional data was embedded in a space with a dimensionality that is larger than, say, two. If the dimensionality of the data is reduced to, say, 30 dimensions, the volume difference between the data space and the latent space is exponentially smaller than when the dimensionality of the data is reduced to two dimensions. More importantly, the dimensionality of the latent space can be selected as to match the intrinsic dimensionality of input data. In that case, it is not necessary to completely collapse classes to obtain low nearest neighbor errors, in particular, since collapsing classes may lead to overfitting. Hence, in learning settings in which the dimensionality of the latent space is relatively large, directly minimizing a smooth approximation to the nearest neighbor error, as is done in NCA, may work much better. To this end, we investigate a variant of NCA, called deep t-distributed NCA (dt-NCA), that also parametrizes the mapping from the data to the latent space by means of a deep neural network[2] and that measures similarities in the latent space using a t-distribution.

The key difference between dt-NCA and dt-MCML, analogous to the key difference between NCA and MCML, is in the cost function that is minimized. In particular, dt-NCA tries to minimize the expected nearest neighbor error by minimizing a smooth approximation of the nearest neighbor error:

$$\ell_{dt-NCA} = -\sum_{ij:i\neq j} \delta_{ij} q_{j|i},$$

where δ_{ij} represents an indicator function, i.e., δ_{ij} equals 1 if $y^{(i)} = y^{(j)}$ and 0 otherwise, and where we used an asymmetric definition for the similarities $q_{j|i}$:

$$q_{j|i} = \frac{(1 + d_{ij}^2/\alpha)^{-\frac{1+\alpha}{2}}}{\sum_{k:k\neq i}(1 + d_{ik}^2/\alpha)^{-\frac{1+\alpha}{2}}}, \quad q_{i|i} = 0.$$

The motivation behind this definition of the similarities in the latent space is identical to the motivation for the similarities in dt-MCML: it helps preventing overfitting, it constructs tighter natural clusters of points with the same class, it improves separation between points with different classes, and it makes the gradient optimization easier due to the presence of more longe-range forces.

If we define $u_{ij} = \left(1 + \frac{d_{ij}^2}{\alpha}\right)^{-\frac{1+\alpha}{2}}$ and $\mathbf{v}_{ij} = f(\mathbf{x}^{(i)}) - f(\mathbf{x}^{(j)})$, the gradient of ℓ_{dt-NCA} with respect to the data representation in the latent space $f(\mathbf{x}^{(i)})$ is given by:

$$\frac{\partial \ell_{dt-NCA}}{\partial f(\mathbf{x}^{(i)})} = \frac{1+\alpha}{\alpha}\left[\sum_{j:j\neq i}\delta_{ij}q_{j|i}\left(\sum_{k:k\neq i} q_{k|i}u_{ik}\mathbf{v}_{ik}\right) + \sum_{j:j\neq i} q_{i|j}u_{ji}\mathbf{v}_{ji}\left(\sum_{k:k\neq j}\delta_{jk}q_{k|j}\right) - \sum_{j:j\neq i}\delta_{ij}q_{j|i}u_{ij}\mathbf{v}_{ij} - \sum_{j:j\neq i}\delta_{ji}q_{i|j}u_{ji}\mathbf{v}_{ji}\right].$$

The gradient of ℓ_{dt-NCA} with respect to α can be calculated as follows:

$$\frac{\partial \ell_{dt-NCA}}{\partial \alpha} = \sum_i \sum_{j:i\neq j}\delta_{ij}q_{i|j}\left(\frac{1+\alpha}{2}u_{ij}d_{ij}^2\alpha^{-2} - \frac{1}{2}\log u_{ij}\right) - \sum_i\sum_{j:j\neq i}\delta_{ij}q_{j|i}\left(\sum_{k:k\neq i} q_{k|i}\left(\frac{1+\alpha}{2}u_{ik}d_{ik}^2\alpha^{-2} - \frac{1}{2}\log u_{ik}\right)\right).$$

3. Experiments

We performed experiments with dt-MCML and dt-NCA on the USPS and MNIST handwritten digit data sets in order to evaluate their performance. In order to investigate the effect of using a t-distribution instead of a Gaussian distribution to measure similarities in the latent space, we compare dt-MCML and dt-NCA to their counterparts that use a Gaussian distribution to measure the similarities (but that use the same deep neural network as parametrization of the function f). We refer to these variants as dG-MCML and dG-NCA, respectively. Both the pre-training and the finetuning for dG-MCML and dG-NCA are identical to the pre-training and the finetuning of dt-MCML and dt-NCA, except for that the Student-t distributions in the definition of q_{ij} and $q_{j|i}$ are replaced by Gaussian distributions (with variance $\sigma = \frac{1}{\sqrt{2}}$), respectively. Note that this replacement also leads to different gradients of the respective cost functions (the gradients for dG-NCA are derived by Salakhutdinov & Hinton (2007)).

In our experiments, we used the same network structure that was proposed by Salakhutdinov & Hinton (2007), i.e, we use a $D-500-500-2000-d$ network. Our selection of this architecture facilitates comparisons with methods used in other papers. We pre-trained all neural networks using the pre-training procedure described by Hinton & Salakhutdinov (2006). Following Hinton & Salakhutdinov (2006), we trained

[2]The reader should note another nonlinear variant of NCA was previously investigated by Salakhutdinov & Hinton (2007).

Dimensionality d	**2D**	**30D**
MCML	35.63 ± 0.44	5.53 ± 0.39
dG-MCML	3.37 ± 0.18	1.67 ± 0.21
dt-MCML ($\alpha = d-1$)	$\mathbf{2.46 \pm 0.35}$	1.73 ± 0.47
dt-MCML (learned α)	2.80 ± 0.36	1.61 ± 0.36
dG-NCA	10.22 ± 0.76	1.91 ± 0.22
dt-NCA ($\alpha = d-1$)	5.11 ± 0.28	$\mathbf{1.15 \pm 0.21}$
dt-NCA (learned α)	6.69 ± 0.92	1.17 ± 0.07

Table 1. Mean and standard deviation of test error (in %) on 2-dimensional and 30-dimensional embedding for various techniques on the 6 splits of USPS data set.

the RBMs using contrastive divergence with one Gibbs sweep (CD-1) for 50 iterations using mini-batches of 100 instances, a learning rate of 0.1, and L2 regularization of the weights with a regularization parameter of 2×10^{-4}. We use a momentum of 0.5 in the first 5 iterations, and a momentum of 0.9 in subsequent iterations.

The finetuning is performed by running conjugate gradients (without weight decay) until convergence. In the finetuning phase, we continue the training of dG-MCML, dt-MCML, dG-NCA, and dt-NCA until the value of the cost function measured on the training set does not decrease anymore. We performed classification experiments on the latent representation of the data using a k-nearest neighbor classifier with $k = 5$. In the experiments with dt-MCML and dt-NCA in which we learn α, we train using $\alpha = d - 1$ for one epoch first, and we update W and α simultaneously in the subsequent iterations.

3.1. Experimental Results on USPS Data Set

The USPS data set contains $11,000$ images of handwritten digits. In the data set, each digit is represented by a 16×16 pixel gray-level image, i.e., the dimensionality of the data set is 256. We constructed six random splits of the USPS data set into a training set of $8,000$ images and a test set of $3,000$ images, and we measured generalization performances that were averaged over these six splits.

Table 1 presents the mean classification performances of 5-nearest neighbor classifiers that were trained on embeddings constructed by various techniques, as well as the corresponding standard deviations. The best performance is typeset in boldface. The results show that dt-MCML outperforms all other methods in terms of classification errors measured on a two-dimensional embedding. On a 30-dimensional embedding, the table reveals that dt-NCA (with learned α) outperforms the other techniques. The results also show that dt-MCML always outperforms dG-MCML, and that dt-NCA always outperforms dG-NCA. Both dt-MCML and dt-NCA perform much much better than standard linear MCML. We did not run linear NCA on the USPS data set, but in (Goldberger et al., 2005), the authors report a classification error on a two-dimensional embedding constructed by NCA of approximately 30%, which is much higher than our best classification error of 2.46% on a two-dimensional embedding.

Figure 2 shows embeddings of $3,000$ test data points in the USPS-fixed split that were constructed by, respectively, dG-MCML, dt-MCML, dG-NCA, and dt-NCA. The embedding constructed by linear MCML is included in the supplemental material. The plots in Figure 2 suggest that dt-MCML produced the best embedding. It puts almost all the data points in the same class close to each other, and it creates large separations between class clusters. The embeddings reveal that both dt-NCA and dt-MCML allow large gaps to form between classes, whereas dG-NCA and dG-MCML must allow for more overlaps between classes. Comparing dt-MCML and dt-NCA, we find that dt-NCA allows data points in the same class to be placed at multiple locations, whereas dt-MCML only maintains one large cluster. This is because dt-NCA maximizes the sum of probabilities $q_{j|i}$'s, whereas dt-MCML maximizes the product of the q_{ij}'s.

3.2. Experimental Results on MNIST Data Set

We also performed experiments on the MNIST data set. The MNIST data set contains gray-level images with $28 \times 28 = 784$ pixels. The data set contains $60,000$ training samples and $10,000$ test samples. Because of the large number of training samples in the MNIST data set, we were forced to use batch training using batches of $10,000$ training samples. Because, of the size of the MNIST data set, we could not perform experiments using MCML (the projected gradient procedure using in MCML requires the eigendecomposition of $n \times n$ matrices).

Table 2 presents the classification performance of nearest neighbor classifiers on embedding constructed by the various techniques (computed on the standard MNIST test set). Again, dt-MCML outperforms the other techniques when embedding into two dimensions. In comparison, the test errors of linear NCA (Goldberger et al., 2005), autoencoders (Hinton & Salakhutdinov, 2006), parametric t-SNE with $\alpha = 1$ (van der Maaten, 2009), parametric t-SNE with learned α, and DNet-kNN (Min et al., 2009) on two-dimensional embeddings of the MNIST data set are,

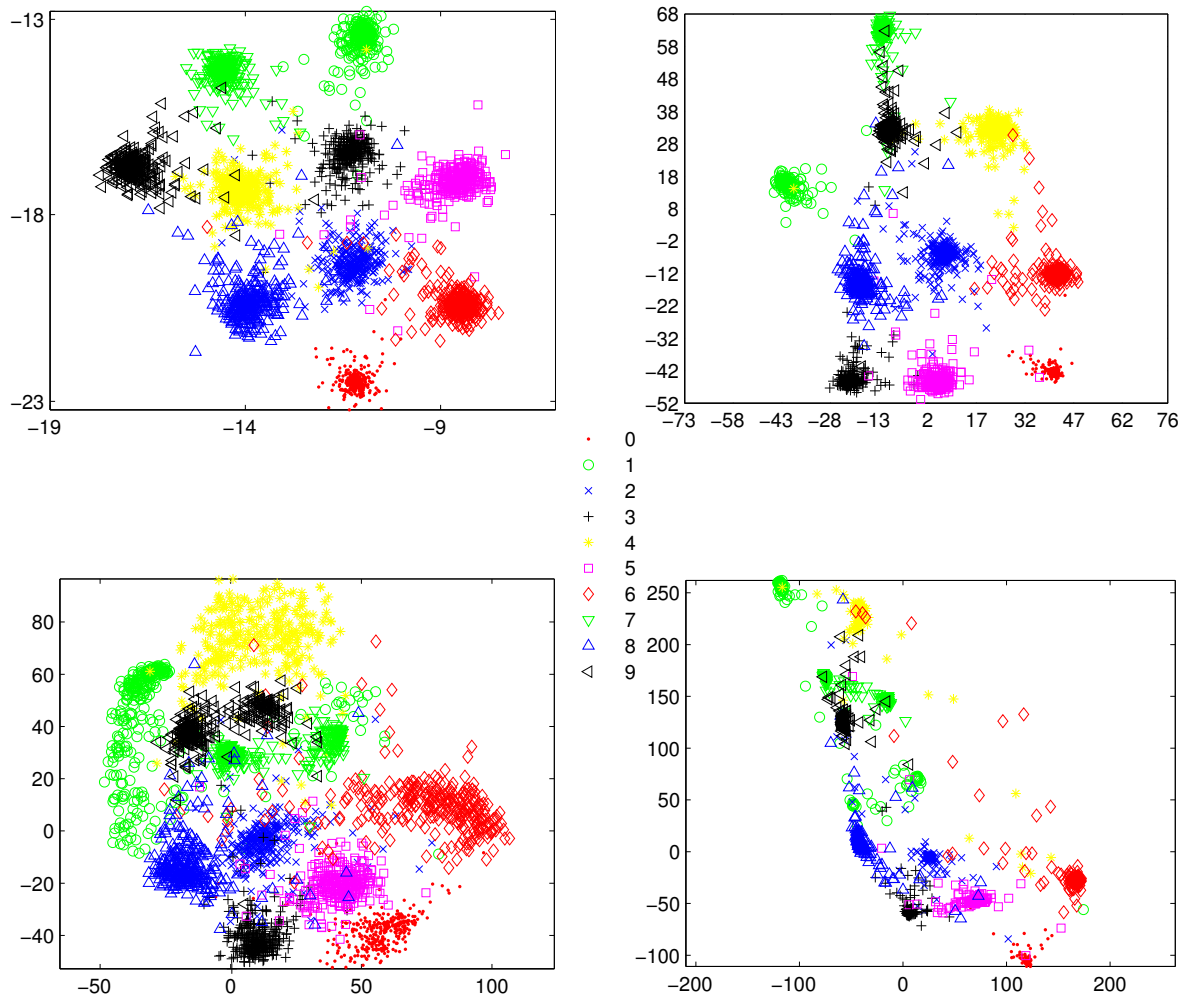

Figure 2. Two-dimensional embeddings of 3,000 USPS test data points constructed by dG-MCML (top left), dt-MCML (top right), dG-NCA (bottom left), and dt-NCA (bottom right).

respectively, 56.84%, 24.7%, 9.9%, 12.68%, and 2.65%. By contrast, the error obtained by dt-MCML is 2.03%.

The results in Table 2 also reveal that, when we are embedding into 30 dimensions, dt-NCA (learned α) outperforms all other techniques. The best performance of dt-NCA of 0.92% is on par with the state-of-the-art results on the MNIST techniques (if adding perturbed digits to the training data is not allowed).

Figure 3 shows embeddings of the 10,000 test data points in the MNIST data set constructed by the various techniques. The embeddings reveal that both dt-NCA and dt-MCML form large separations between classes, whereas dG-NCA and dG-MCML produce overlaps between many of the classes in the data.

4. Concluding Remarks

In this paper, we presented two techniques for supervised parametric dimensionality reduction that used deep networks. The experimental results presented in the previous section reveal that dt-MCML outperforms its linear counterpart MCML and its Gaussian counterpart with a deep architecture (dG-MCML) when it is used to construct two-dimensional embeddings, whereas dt-NCA outperforms its linear counterpart and its Gaussian counterpart (dG-NCA; Salakhutdinov & Hinton (2007)) when it is used to embed data into a space with a dimensionality larger than two. Taken together, our results demonstrate the advantage of using a deep architecture to parametrize the

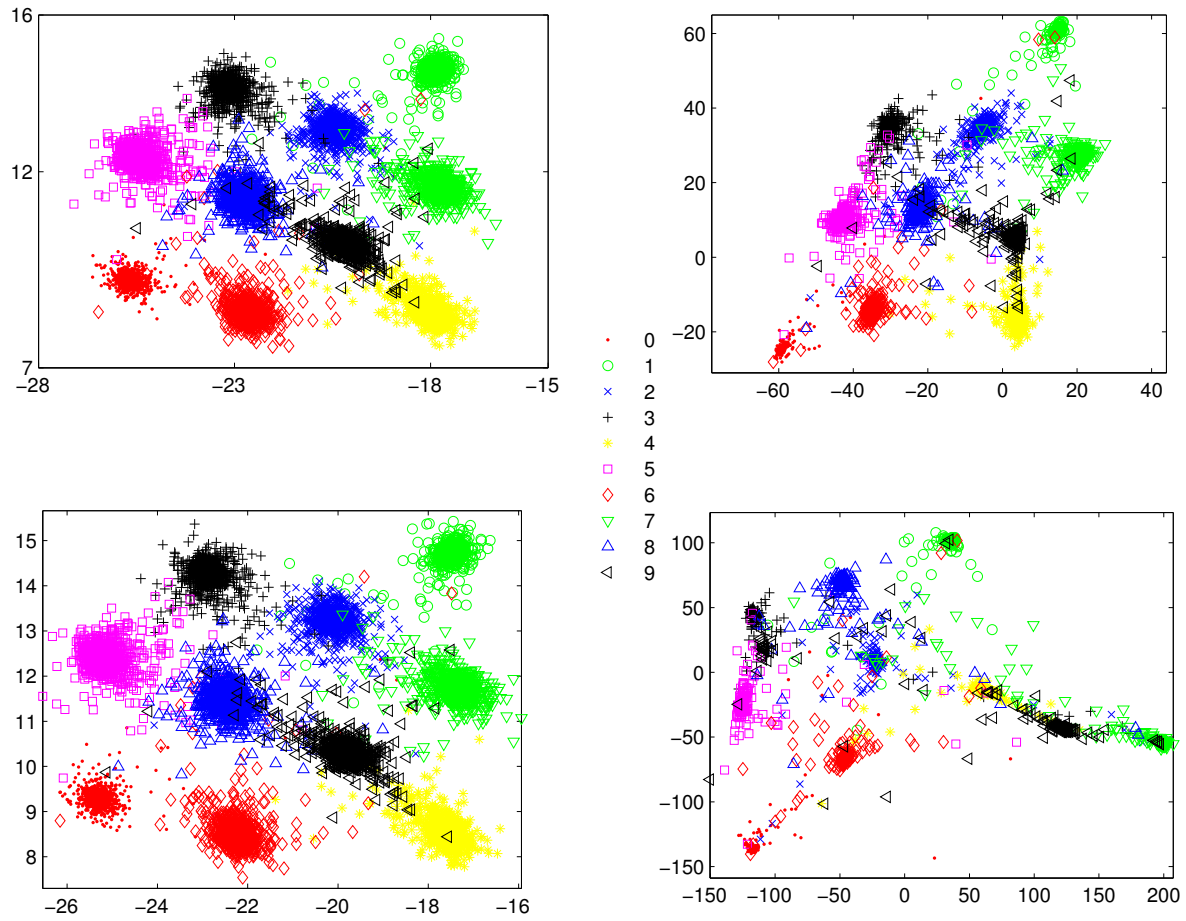

Figure 3. Two-dimensional embeddings of 10,000 MNIST test data points constructed by dG-MCML (top left), dt-MCML (top right), dG-NCA (bottom left), and dt-NCA (bottom right).

Dimensionality d	**2D**	**30D**
dG-MCML	2.13	1.49
dt-MCML ($\alpha = d - 1$)	**2.03**	1.63
dt-MCML (learned α)	2.14	1.49
dG-NCA	7.95	1.11
dt-NCA ($\alpha = d - 1$)	3.48	**0.92**
dt-NCA (learned α)	3.79	0.93

Table 2. Test error (in %) on 2-dimensional and 30-dimensional embedding for various techniques on the MNIST data set.

mapping, as well as the merits of using a heavy-tailed Student t-distribution to measure the pairwise similarities between the low-dimensional representations of the data points.

We did not yet explain in detail why dt-MCML performs better than dt-NCA when constructing two-dimensional embeddings, wheras dt-NCA performs better than dt-MCML when constructing embeddings of a higher dimensionality. This is because dt-MCML tries to collapse all the data points in each class to one point by maximizing the product of the probabilities q_{ij}, whereas dt-NCA tries to bring data points in the same class together by maximizing the sum of the asymmetric probabilities $q_{j|i}$. As a result, when embedding into two dimensions, the objective of dt-NCA can be roughly maximized by setting some of the $q_{i|j}$'s very large and others very small on training data, to compromise for the limited amount of space available in a two-dimensional space. By contrast, such a setting of the q_{ij}'s is not favored by dt-MCML. In fact, the dt-MCML cost function typically does not allow

some of the q_{ij}'s to be set very small, thus prohibiting data points in the same class from spreading out. This property of dt-MCML has advantages and disadvantages. When embedding in a two-dimensional space, collapsing classes is often good because the available space to accommodate multiple clusters corresponding to the same class is very limited. However, in latent spaces with a higher dimensionality, there is enough space to accommodate dissimilar data points, as a result of which the collapsing of classes becomes superfluous and leads to overfitting. In contrast, dt-NCA does not require all the data points in the same class to stay very close to each other in the latent space, as a result of which it performs better when embedding in a, say, 30-dimensional latent space.

The success of dt-MCML is closely related to the recent success of t-SNE (van der Maaten & Hinton, 2008) in unsupervised dimensionality reduction. However, the reader should note that the reason for the success of t-SNE is a different one. In t-SNE, the use of the Student t-distribution helps to avoid the *crowding problem*. The crowding problem occurs when one tries to preserve local similarity structure in a low-dimensional data representation, which forces one to model dissimilar points to far apart. The use of the Student-t distribution resolves this problem, because it allows dissimilar points to modeled to far apart. In dt-MCML, the aim of the use of a Student t-distribution is to force data points with the same class to collapse better, i.e., to form tight clusters, while increasing the gaps between points with dissimilar labels. The latter characteristic of dt-MCML and dt-NCA is shared with t-SNE: they all encourage larger separations to form between different natural clusters.

An additional advantage of dt-MCML over MCML that we did not discuss yet is its computational complexity. Even though dt-MCML is the deep nonlinear extension of MCML, dt-MCML scales very well to massive high-dimensional data sets (although this does require a batch training procedure). By contrast, MCML is slow when it is applied on large data sets (even in batch training procedures) because it uses a projected gradient descent optimizer that performs many eigendecompositions of $n \times n$ matrices.

In future work, we aim to investigate extensions of dt-MCML and dt-NCA that include an additional term penalizing reconstruction error from a deep autoencoder in order to improve the performance of our approach in semi-supervised learning settings. Moreover, we aim to investigate pre-training approaches using denoising autoencoders instead of RBMs.

Acknowledgements

Laurens van der Maaten is supported by the Netherlands Organisation for Scientific Research (NWO; grant no. 680.50.0908), and by EU-FP7 Social Signal Processing. The authors thank Geoffrey Hinton, Amir Globerson, and Xiaojian Shao for helpful discussions.

References

Globerson, A. and Roweis, S. Metric learning by collapsing classes. In *NIPS*, pp. 451–458. 2006.

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. Neighbourhood components analysis. In *NIPS*, pp. 513–520. 2005.

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

Hinton, G., Osindero, S., and Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pp. 473–480. 2007.

Min, R., Stanley, D., Yuan, Z., Bonner, A., and Zhang, Z. A deep non-linear feature mapping for large-margin kNN classification. In *ICDM*, pp. 357–366. 2009.

Salakhutdinov, R. and Hinton, G. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AI-STATS*, pp. 412–419, 2007.

van der Maaten, L. Learning a parametric embedding by preserving local structure. *AI-STATS*, pp. 384–391, 2009.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *JMLR*, 9:2579–2605, November 2008.

Weinberger, K., Blitzer, J., and Saul, L. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, pp. 1473–1480. 2006.

Weston, J., Ratle, F., and Collobert, R. Deep learning via semi-supervised embedding. In *ICML*, pp. 1168–1175, 2008.

Xing, E., Ng, A., Jordan, M., and Russell, S. Distance metric learning with application to clustering with side-information. In *NIPS*, pp. 505–512. 2003.