Parametric Dimension Reduction by Preserving Local Structure

Chien-Hsun Lai * Ming-Feng Kuo † Yun-Hsuan Lien ‡ Kuan-An Su § Yu-Shuen Wang ¶

National Yang Ming Chiao Tung University, Taiwan

ABSTRACT

We extend a well-known dimension reduction method, t-distributed stochastic neighbor embedding (t-SNE), from non-parametric to parametric by training neural networks. The main advantage of a parametric technique is the generalization of handling new data, which is beneficial for streaming data visualization. While previous parametric methods either require a network pre-training by the restricted Boltzmann machine or intermediate results obtained from the traditional non-parametric t-SNE, we found that recent network training skills can enable a direct optimization for the t-SNE objective function. Accordingly, our method achieves high embedding quality while enjoying generalization. Due to mini-batch network training, our parametric dimension reduction method is highly efficient. For evaluation, we compared our method to several baselines on a variety of datasets. Experiment results demonstrate the feasibility of our method. The source code is available at https://github.com/a07458666/parametric_dr.

Index Terms: Computing methodologies—Dimensionality reduction and manifold learning—; Human-centered computing—Visualization toolkits—

1 INTRODUCTION

Dimension reduction (DR) techniques are widely utilized to facilitate data exploration and visual analysis. The goal is to project high-dimensional data X to low-dimensional embedding Y, while retaining either global or local data structures. Among the techniques, t-distributed stochastic neighbor embedding (t-SNE) [23] is considered a classical method, which attempts to maximize the probability of nearby/distinct points in X to be nearby/distinct in Y. Subsequently, several extensions were presented to improve the performance of t-SNE because it has to update the conditional probability of each data point in the embedding space in each iteration.

T-SNE is a non-parametric DR method. The advantage of a non-parametric method is high flexibility when determining the positions of data points in the low-dimensional space. Without the parametric constraint, it can effectively optimize the objective function and obtain high-quality results [11]. On the other hand, non-parametric methods lack generalization, which cannot apply the transformation obtained from the given dataset to reduce the dimensionality of new data points. To handle new data points, the methods have to merge the old and the new datasets, and then recompute the data positions in the embedding space. Since the goal of DR is retaining only data structures, the results in consecutive runs could differ by a rigid transformation. Users cannot compare results quickly in the runs and must pay more attention when they study online or streaming data by using non-parametric methods.

*e-mail: jxcode.tw@gmail.com
†e-mail: sunnyday90154@gmail.com
‡e-mail: sophia.yh.lien@gmail.com
§e-mail: a07458666@gmail.com
¶e-mail: yushuen@cs.nctu.edu.tw

We train a neural network to reduce data dimensionality by optimizing the t-SNE objective function. The network can be considered a function and is generalized to handle new data points. While previous methods claim that neural networks would get stuck in poor local minimums [37] if they are trained by directly optimizing the t-SNE objective function, we found that recent training skills, particularly the Adam optimizer [16] and Leaky Relu activation function [22, 42], can effectively prevent such a problem. Accordingly, the networks are trained from scratch, and the quality of DR results improves. Furthermore, we adopt the commonly used stochastic gradient descent method to train the network, which considers only a batch of data when updating network parameters in each iteration. The computation complexity of updating conditional probabilities of data points in the embedding space reduces, and the system performance increases. To evaluate the effectiveness of our DR network, we compared the network with several parametric methods in terms of trustworthiness, continuity, and neighborhood hit on a variety of datasets. Experiment results show that our DR network outperformed previous methods.

2 RELATED WORKS

Dimension Reduction. Dimension reduction algorithms project high-dimensional data X to low-dimensional embedding Y, which allows users to visually analyze data structures and identify outliers. Among the methods, principal component analysis (PCA) [34] seeks to maximize the variance of Y in each dimension. Linear discriminant analysis (LDA) [1] extends the PCA by incorporating data labels, which aims to optimally separate data points of different labels. On the other hand, multi-dimensional scaling (MDS) [3, 18, 31], isometric feature mapping (IsoMap) [36], self-organizing map (SOM) [38], locally linear embedding (LLE) [30], maximum variance unfolding (MVU) [40], and Laplacian eigenmaps [2], were presented to preserve the relative distances of data points when projecting X into Y.

Stochastic neighbor embedding (SNE) [13] models the relationship between data points by a conditional probability. By minimizing the KL divergence of data distributions in high- and low- dimensional spaces, it faithfully preserves the local structure of data. Subsequently, t-SNE [23] extends the SNE by symmetrizing the probability distribution and employing a heavy-tailed student t-distribution to compute the joint probability of data points in the embedding space. To preserve other characteristics of high-dimensional data, such as global data distances, Im et al. [15] replaced the KL divergence with f-divergence metrics for different types of structure discovery. LargeVis [35] uses a similar strategy to t-SNE but eliminates the need for normalization in the embedding space. UMAP [24] utilizes the language of algebraic topology to preserve the local distance structure. It achieves comparable quality to t-SNE and LargeVis, while being able to retain the global structure of data and consume lower computation cost.

Parametric Extensions of t-SNE and UMAP. T-SNE has proven to work well on many real-world datasets. One drawback, however, is the lack of an explicit mapping function to handle unseen data. Consequently, several methods were presented to extend t-SNE from non-parametric to parametric at the cost of lower embedding quality, which is a result of the non-flexibility of a parametric form. One way to achieve parametric extension is to pair a part

of high-dimensional data and their embeddings produced by the traditional t-SNE. Then, the methods apply local transformations to reduce the dimensionality of unseen data according to the relative distances to those paired samples [4, 10, 11]. Another way to reduce data dimensionality is training neural networks. Due to the complex parameter interactions, networks often get stuck in bad local minimums if their parameters are updated by backpropagation. To overcome the problem, Parametric t-SNE [37] uses a stack of restricted Boltzmann machines to pre-train a feed-forward network, and then fine-tunes the network using the t-SNE objective function. Subsequently, dt-SEE [25] extends the parametric t-SNE with exemplars in high-dimensional space to avoid pairwise distance calculation. Espadoto et al. [7] applied the traditional t-SNE to project high-dimensional data X to Y. Then, they trained the network on the paired X and Y to achieve generalization. Recently, the authors of [32] extended the non-parametric UMAP by training neural networks. They used negative sampling to prevent the large batch size required by the parametric t-SNE [37] when training. While previous methods apply complex training strategies to prevent networks from getting stuck in bad local minimums, in this study, we point out that recent optimizer and activation functions in deep learning can achieve the goal without additional computation.

3 Parametric Dimension Reduction using t-SNE

3.1 Background

Given a data set $X = \{x_i \in R^d\}$ in high-dimensional space and the embedding set $Y = \{y_i \in R^s\}$ that contains the corresponding points in low-dimensional space. The DR method attempts to find a mapping function that can minimize a predefined loss

$$arg\min_{Y} C(X,Y) \tag{1}$$

to retain data structures. In each iteration, the algorithm calculates the gradient of the loss with respect to the embedding $\frac{\partial C}{\partial y_i}$ and moves each y_i to the desirable position. For simplicity, we use the term *embedding* to denote data points in low-dimensional space in later sections.

T-SNE attempts to maintain points that are nearby/distinct in high-dimensional space to be nearby/distinct in low-dimensional space. To implement this idea, it models the similarity of data points x_i and x_j by the joint probability p_{ij}. The idea can be expressed as:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}, \quad p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k!=i} exp(-||x_i - x_k||^2/2\sigma_i^2)} \quad , \tag{2}$$

where σ_i is the variance determined by the neighbor perplexity. For the corresponding points y_i and y_j in low-dimensional space, t-SNE employs a heavy-tailed student t-distribution to compute a weight w_{ij} between them. It then normalizes the weight to obtain the joint probability as follows:

$$q_{ij} = \frac{w_{ij}}{\sum_k \sum_l w_{kl}}, \quad w_{ij} = \frac{1}{1 + ||y_i - y_j||_2^2} \quad . \tag{3}$$

Let P and Q be the joint distributions of p_{ij} and q_{ij}, respectively, t-SNE minimizes the divergence of the joint probability

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \quad . \tag{4}$$

Then, one can derive the gradient concerning y_i to update point positions in low-dimensional space. Namely,

$$\frac{\partial C}{\partial y_i} = 4 \sum_j \frac{(p_{ij} - q_{ij})(y_i - y_j)}{1 + ||y_i - y_j||_2^2} \quad . \tag{5}$$

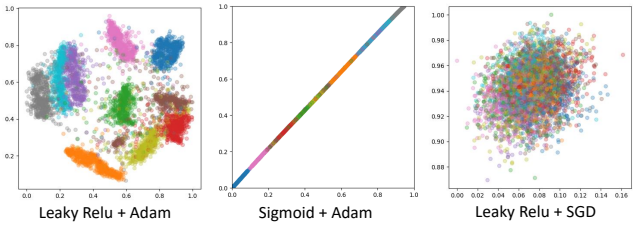

Figure 1: The activation function and optimizer are essential when training neural networks to reduce data's dimensionality. This experiment was conducted on the MNIST dataset.

3.2 Neural Network based t-SNE

We extend the non-parametric t-SNE to a parametric t-SNE by training a deep neural network. The network $f_\theta(x_i)$ maps each input data point x_i to the embedding y_i, where θ indicates the parameters of f. Accordingly, Equation 1 can be written as:

$$arg\min_{\theta} C(X, f_\theta(X)). \tag{6}$$

Because our goal is to compute the function f, the unknowns are the network parameters θ. The gradient then becomes

$$\frac{\partial C(X, f_\theta(X))}{\partial \theta} = \sum_i \frac{\partial C(x_i, f_\theta(x_i))}{\partial f_\theta(x_i)} \frac{\partial f_\theta(x_i)}{\partial \theta}, \tag{7}$$

for updating the network. By substituting $\frac{\partial C}{\partial f_\theta(x_i)}$ in Equation 7 with Equation 5, we obtain

$$\frac{\partial C}{\partial \theta} = 4 \sum_i \sum_j \frac{(p_{ij} - q_{ij})(f_\theta(x_i) - f_\theta(x_j))}{1 + ||f_\theta(x_i)) - f_\theta(x_j)||_2^2} \frac{\partial f_\theta(x_i)}{\partial \theta}. \tag{8}$$

for updating network parameters.

Networks may get stuck in poor local minimums if they are trained by directly optimizing the t-SNE objective function [37]. We accidentally found that recent network training skills, particularly the Adam optimizer [16] and Leaky Relu activation function [22, 42], can effectively prevent such a problem. Figure 1 shows the results achieved using different optimizers and activation functions. Specifically, Relu and leaky Relu can avoid the gradient vanish problem that commonly appears in the Sigmoid function; and the Adam optimizer considers the momentum to prevent high oscillation when partial samples are utilized to estimate the gradients.

Optimizing the network f using Equation 8 demands the computation of each joint probability q_{ij} and $f_\theta(x_i) - f_\theta(x_j)$ whenever the network is updated. The computation complexity is $O(N^2)$, where N is the number of data points in the whole dataset. This is also the reason that the classical t-SNE is slow. Fortunately, the independent and identically distributed assumption enables neural networks to be trained by using the stochastic gradient descent method, where, in each iteration, only a small amount of data are used to update network parameters. Specifically, we randomly sample a batch of data points from the dataset and then use them to compute the joint probability q_{ij}, where i and j are the sample indexes in a batch, respectively. Let n be the batch size, the complexity is then reduced from $O(N^2)$ to $O(n^2)$ in each iteration.

3.3 Training Details

We implemented the presented DR method using PyTorch [28]. Without a loss of generality, the network is composed of fully connected layers. We built the layers with hidden units $1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 2$ to gradually reduce data dimensionality. We used the Xavier initialization [12] to initialize the network parameters,

set the batch size to 256, and set the learning rate to 10^{-3}. The activation function in each layer is Leaky ReLU [22], and the Adam optimizer [16] is used to update network parameters. Moreover, we used the early exaggeration trick the same as the original t-SNE method, in the first 250 iterations for a faster convergence.

4 Evaluation and Experiment Results

4.1 Baselines

We compared our method to several parametric DR methods. The goal is to transform high-dimensional data X to low-dimensional embedding Y. Since the experiment results in the work of [8] reveal that t-SNE and UMAP achieved the highest quality projections, we included their parametric extensions (i.e., *Param_T* [37] and *Param_U* [32]) in the comparison. We used a GitHub implementation of *Param_T*[1] and the official implementation of *Param_U*[2]. In addition, we compared our method with autoencoder (*AE*) and deep learning multidimensional projections (DLMP) because they utilize neural networks to reduce data dimensionality as well. The network architecture of AE's encoder and DLMP was the same to ours, and the AE's decoder was symmetric to the encoder. We also compared the well-known parametric DR method, principal component analysis (PCA)[3], in this study.

4.2 Quality Metrics

DR is an ill-posed problem because information loss is inevitable. Often, several metrics are jointly used to evaluate the quality of DR methods. In this study, we considered trustworthiness M_t, continuity M_c [39], and neighborhood hit M_{NH} [29] because the metrics are well known and interpretable. The metrics evaluate the K-nearest neighbors (KNNs) of each sample in the embedding and the high-dimensional space. Note that the neighborhood hit is applicable only to labeled datasets. It also requires data in the high-dimensional space to be well separable into classes. Following the setting of [8], we set $k = 7$ if the KNNs are needed for evaluation. The values of M_t, M_c, and M_{NH} are all in $[0,1]$. Among the metrics, 1 implies the best quality and 0 indicates the worst. The integrated metrics can be written as

$$\mu = (M_t + M_c + M_{NH})/3. \tag{9}$$

4.3 Datasets

We compared the DR methods on several real-world datasets. They are from different domains and exhibit non-trivial data structures. Specifically, the datasets are: Bank [26], MNIST [19], F-MNIST [41], Cat & Dog [6], Cifar10, Cifar100 [17], IMDB [21], Hatespeech [5], Letter [9], Norb [20], and SVHN [27]. Since the neighborhood hit M_{NH} requires data in the high-dimensional space to be well separated, to achieve a meaningful experiment, we followed the work of [7] and pre-processed the datasets prior to DR comparison. We used the DenseNet201 [14] pre-trained on the ImageNet to extract features from the Cat&Dog, Cifar10, Cifar100, Norb, and SVHN datasets. All of the resulting features were 1920D because of the maxpooling layer at the back of DenseNet201. We also applied the term frequency-inverse document frequency (TF-IDF) [33] to compute the 500D text features from the IMDB and the Hatespeech datasets. Regarding the MNIST and Fashion-MNIST, we simply flatten the images to 784D vectors. Finally, the raw data in the Bank and the Letter datasets were directly used in the comparison.

4.4 Procedures and Experiment results

The evaluation focused on generalization. The projection parameters were optimized on the training set and then evaluated on the testing set. In the experiments, we randomly selected 5000 samples from each dataset for training and another 15000 samples for testing. For the methods based on neural networks, 10% of the training samples were randomly selected for validation. The training process started from the initialization and repeated until the validation loss stopped decreasing (i.e., $< 10^{-5}$) or the epoch number exceeds 200.

Since user-defined parameters could considerably affect projection quality, we did a grid search that could maximize Equation 9 if the method requires a parameter tuning. Specifically, we set the perplexity to $\{50, 150, 300, 500\}$ when the variants of t-SNE were executed. We set the number of neighbors to $\{5, 10, 15\}$ and the minimum distance to $\{0.001, 0.01, 0.1, 0.5\}$ when the variants of UMAP were executed. We also set the iteration number to $\{1000, 3000\}$ when the non-parametric t-SNE was used to generate embeddings required by the DLMP.

Visual Comparison. Figure 2 and Figure **??** in our supplemental material show the 2D embeddings projected by the baselines and our method. We select the embedding of each method on each dataset that has the best quality (Equation 9). In an embedding, each dot is a sample, and each distinct color indicates a class. Ideally, a good embedding should retain the structure of data in the high dimensional space. Therefore, dots in the same color should gather, and dots in different colors should separate. As can be seen, PCA's embeddings were the worst because points with different colors are mixed, and the clusters are indistinguishable. Clusters in AE's embeddings are separable, yet the clusters may not be compact. We suspect the reason is that AE attempts to reconstruct high-dimensional samples. The spread of dots in a cluster allows AE to encode individual and unique properties of each sample. DLMP's embeddings reveal compact clusters in most of the datasets. However, boundaries between clusters are slightly vague, and points around the boundaries are in different colors. Regarding Param_T and Param_U, the clusters are more distinguishable than those achieved by DLMP due to the more compact clusters. Finally, our method generated good 2D embeddings because of noticeable boundaries between clusters.

We do not visually compare the embeddings of Bank, IMDB, Cifar100, and Hatespeech because of two reasons. First, the features of Bank, IMDB, and Hatespeech are unrepresentative. Since samples of different classes mix up in the high dimensional space, visual comparison of embeddings is meaningless. Second, Cifar100 contains 100 classes, and finding as many distinct colors for representing the samples is challenging. The resulting visual clutter would prevent users from examining the embeddings.

Quantitative Evaluation. Table 1 (top) shows the results generated by the baselines and our method, where each value is achieved by the best parameter. To evaluate whether the methods were robust to the given parameters, we also selected a parameter for each method that can work best on all datasets. The results are shown in Table 1 (middle). Due to the page limit, we show only the average of the three quality metrics in Table 1 and refer readers to our supplemental material (Table **??**) for their respective values. Among the methods, PCA was the worst due to its linearity constraint. AE performed reasonably well although its objective was reconstruction. It even outperformed DLMP in most of the datasets. We suspect the reason could be that DLMP learned the projection from results generated by non-parametric DR methods rather than from the projection loss. Besides, the numbers indicated that our method outperformed the baselines in most of the datasets. Finally, we show the results of non-parametric t-SNE for readers to understand the cost of parameterization. We also compared the results achieved using different network architectures in our supplemental material (Table **??**).

Performance. We compared the computation time of our method and the baselines on the selected datasets. As indicated in Table 1 (bottom), PCA was the fastest because of its linear property. Our method was the second-fastest because the network was trained from scratch using the stochastic gradient descent method. We point out that the computation time of DLMP includes the pre-process

[1] https://github.com/luke0201/parametric-tsne-keras

[2] https://github.com/timsainb/ParametricUMAP_paper

[3] https://scikit-learn.org/stable/

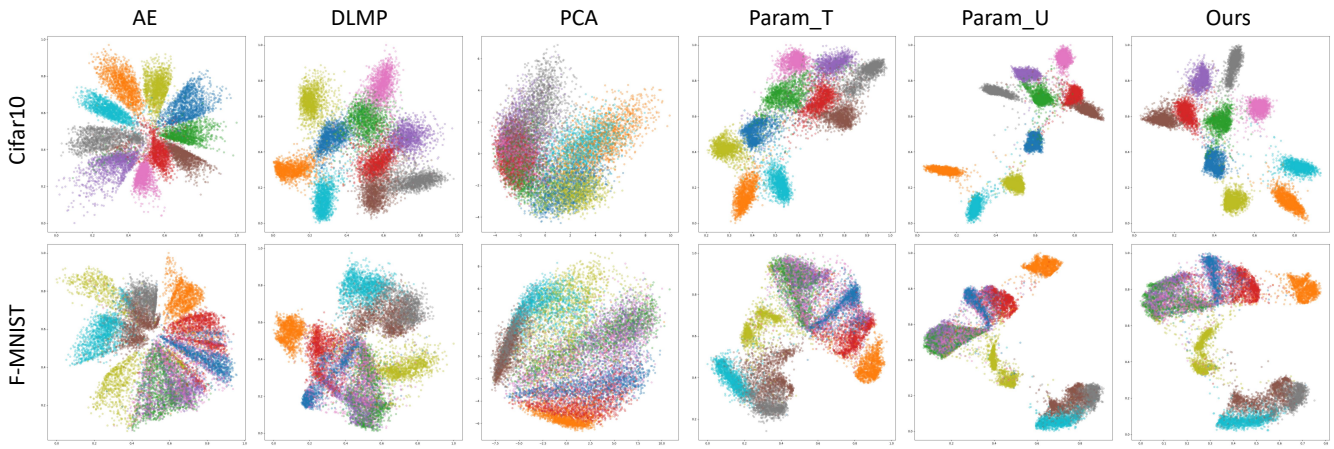

Figure 2: We project high dimensional data to planar space for visual comparison. Each dot is a sample and each distinct color indicates a class.

	Method	Bank	Cat & Dog	Cifar10	Cifar100	F-MNIST	Hatespeech	IMDB	Letter	MNIST	Norb	SVHN	Mean
By dataset	AE	0.928	0.892	0.951	0.696	0.875	**0.862**	0.585	0.799	0.859	0.952	**0.975**	0.852
	DLMP	0.899	0.885	0.916	0.622	0.864	0.822	0.608	0.812	0.820	0.957	0.952	0.832
	PCA	0.898	0.879	0.749	0.574	0.807	0.823	0.539	0.693	0.715	0.877	0.800	0.759
	Param_T	0.929	0.894	0.942	0.681	0.881	**0.862**	**0.662**	0.827	0.883	0.958	0.972	0.863
	Param_U	0.921	0.892	0.959	0.690	0.880	0.858	0.653	0.850	0.905	0.962	0.972	0.867
	Ours	**0.935**	**0.898**	**0.963**	**0.698**	**0.884**	**0.862**	0.639	**0.861**	**0.909**	**0.972**	0.974	**0.872**
	t-SNE	**0.942**	**0.939**	**0.981**	**0.847**	**0.927**	**0.922**	**0.752**	**0.973**	**0.971**	**0.998**	**0.985**	**0.931**
By method	AE	0.928	0.892	0.951	**0.696**	0.875	**0.862**	0.585	0.799	0.859	0.952	**0.975**	0.852
	DLMP	0.892	0.873	0.913	0.609	0.864	0.819	0.606	0.812	0.807	0.951	0.952	0.827
	PCA	0.898	0.879	0.749	0.574	0.807	0.823	0.539	0.693	0.715	0.877	0.800	0.759
	Param_T	0.929	**0.894**	0.939	0.664	0.881	**0.862**	**0.659**	0.827	0.880	0.953	0.970	0.863
	Param_U	0.911	0.884	0.959	0.687	0.879	0.850	0.642	**0.850**	0.902	0.962	0.972	0.863
	Ours	**0.935**	0.893	**0.961**	0.682	**0.882**	**0.862**	0.634	0.822	**0.909**	**0.970**	0.974	**0.866**
	t-SNE	**0.942**	**0.939**	**0.978**	**0.844**	**0.927**	**0.922**	**0.748**	**0.973**	**0.970**	**0.998**	**0.984**	**0.930**
Timing	AE	134.30	479.80	319.19	301.95	255.19	198.25	272.00	171.62	315.25	454.53	229.66	284.70
	DLMP	103.66	188.44	141.08	142.36	127.11	112.30	121.44	97.40	132.24	192.86	106.01	133.17
	PCA	0.05	1.41	0.55	0.33	0.17	0.05	0.12	0.08	0.45	1.53	0.14	0.44
	Param_T	853.79	1119.44	928.89	938.10	1005.43	1032.97	1018.96	1015.88	960.89	1090.83	832.03	981.56
	Param_U	195.46	206.99	139.08	147.78	169.22	93.82	182.58	141.35	118.03	109.02	128.07	148.31
	Ours	16.63	42.05	30.28	27.38	24.46	19.35	30.41	17.83	24.09	47.11	24.59	27.65
	t-SNE	312.59	374.08	260.89	258.69	240.99	372.71	262.24	232.96	266.94	328.34	139.09	277.23

Table 1: (Top and middle) The numbers show the quality (Equation 9) of embeddings generated by the baselines and our method. Each method is given the best parameter that can work well on each dataset (top) and on all datasets (middle), respectively. We highlight the best quality in bold fonts. (Bottom) Timing statistics in seconds. Each number is the average of multiple runs using the grid search parameters (Section 4.4).

of traditional t-SNE since it demands the ground truth embeddings. The training part of DLMP was as fast as our method. Regarding AE, it took longer than our method and DLMP because it had an additional decoder for reconstructing high-dimensional data. Finally, both Param_T and Param_U demands the construction of a graph when training the network. Specifically, for each sample i in the high dimensional space, the two methods find the neighbors of i and apply the attractive forces to pull the samples in positive pairs possibly close in the embedding. They also used the repulsive forces to push samples in negative pairs apart. It deserves noting that Table 1 shows the training time of dimensionality reduction. All the methods can project data to a low dimensional space instantly after the parameters are optimized.

4.5 Limitations

We extend the non-parametric t-SNE to a parametric DR method. Although the experiment results demonstrated the strength of our techniques, the loss function we minimized in this study is identical to non-parametric t-SNE. While the non-parametric methods optimize objectives by directly moving samples in the embedding space, our parametric extension was challenging to outperform the non-parametric techniques in terms of quality because of limited network capacity and the parametric constraint.

5 Conclusions

We extend non-parametric t-SNE to parametric DR methods by training neural networks. Thanks to recent training skills in deep learning, we update network parameters by directly optimizing the t-SNE objective function. In addition, we use a mini-batch of samples rather than the whole dataset when training networks. The extension enables the generalization of t-SNE and reduces its computation cost. Finally, we release the codes for public use since our parametric method is practical for both general and streaming data.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by the Ministry of Science and Technology, Taiwan (110-2221-E-A49 -062 - and 109-2221-E-009 -097 -).

REFERENCES

[1] S. Balakrishnama and A. Ganapathiraju. Linear discriminant analysis-a brief tutorial. In *Institute for Signal and information Processing*, vol. 18, pp. 1–8, 1998.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[3] I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[4] K. Bunte, M. Biehl, and B. Hammer. A general framework for dimensionality-reducing data visualization mapping. *Neural Computation*, 24(3):771–804, 2012.

[5] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of Association for the Advancement of Artificial Intelligence*, vol. 11, 2017.

[6] J. Elson, J. J. Douceur, J. Howell, and J. Saul. Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of ACM Conference on Computer and Communications Security*, 2007.

[7] M. Espadoto, N. S. T. Hirata, and A. C. Telea. Deep learning multidimensional projections. *Information Visualization*, 19(3):247–269, 2020.

[8] M. Espadoto, R. M. Martins, A. Kerren, N. S. Hirata, and A. C. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE transactions on visualization and computer graphics*, 27(3):2153–2173, 2019.

[9] P. W. Frey and D. J. Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

[10] A. Gisbrecht, W. Lueks, B. Mokbel, and B. Hammer. Out-of-sample kernel extensions for nonparametric dimensionality reduction. In *Proceedings of European Symposium on Artificial Neural Networks*, 2012.

[11] A. Gisbrecht, A. Schulz, and B. Hammer. Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, 147:71–82, 2015.

[12] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 249–256, 2010.

[13] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Proceedings of Advances in neural information processing systems*, pp. 857–864, 2003.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[15] D. J. Im, N. Verma, and K. Branson. Stochastic neighbor embedding under f-divergences. *arXiv preprint arXiv:1811.01247*, 2018.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[19] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[20] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of Computer Vision and Pattern Recognition*, vol. 2, pp. II–104, 2004.

[21] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the Association for Computational Linguistics: Human language technologies*, pp. 142–150, 2011.

[22] A. L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[23] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9:2579–2605, 2008.

[24] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction, 2018.

[25] M. R. Min, H. Guo, and D. Song. Exemplar-centered supervised shallow parametric data embedding. In *Proceedings of International Joint Conferences on Artificial Intelligence*, pp. 2479–2485, 2017.

[26] S. Moro, P. Cortez, and P. Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[27] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Proceedings of NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library, 2019.

[29] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, 2008.

[30] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[31] N. Saeed, H. Nam, M. I. U. Haq, and D. M. S. Bhatti. A survey on multidimensional scaling. *ACM Computing Surveys.*, 51(3):47:1–47:25, 2018.

[32] T. Sainburg, L. McInnes, and T. Q. Gentner. Parametric UMAP embeddings for representation and semi-supervised learning. *arXiv preprint arXiv:2009.12981*, 2020.

[33] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[34] L. I. Smith. A tutorial on principal components analysis. Technical report, 2002.

[35] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of World Wide Web conference*, pp. 287–297. ACM, 2016.

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

[37] L. Van Der Maaten. Learning a parametric embedding by preserving local structure. In *Proceedings of Artificial Intelligence and Statistics*, pp. 384–391, 2009.

[38] M. M. Van Hulle. *Self-organizing Maps*, pp. 585–622. 2012.

[39] J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *Proceedings of European Symposium on Artificial Neural Networks*, vol. 6, pp. 557–562. Citeseer, 2006.

[40] K. Q. Weinberger and L. K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *Proceedings of Association for the Advancement of Artificial Intelligence*, vol. 6, pp. 1683–1686, 2006.

[41] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[42] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.