Laplacian-based Cluster-Contractive t-SNE for High-Dimensional Data Visualization

YAN SUN*, and YI HAN*, The Chinese University of Hong Kong, Shenzhen, China
JICONG FAN, The Chinese University of Hong Kong, Shenzhen and Shenzhen Research Institute of Big Data, China

Dimensionality reduction techniques aim at representing high-dimensional data in low-dimensional spaces to extract hidden and useful information or facilitate visual understanding and interpretation of the data. However, few of them take into consideration the potential cluster information contained implicitly in the high-dimensional data. In this paper, we propose LapSNE, a new graph-layout nonlinear dimensionality reduction method based on t-SNE, one of the best techniques for visualizing high-dimensional data as 2D scatter plots. Specifically, LapSNE leverages the eigenvalue information of the graph Laplacian to shrink the potential clusters in the low-dimensional embedding when learning to preserve the local and global structure from high-dimensional space to low-dimensional space. It is nontrivial to solve the proposed model because the eigenvalues of normalized symmetric Laplacian are functions of the decision variable. We provide a majorization-minimization algorithm with convergence guarantee to solve the optimization problem of LapSNE and show how to calculate the gradient analytically, which may be of broad interest when considering optimization with Laplacian-composited objective. We evaluate our method by a formal comparison with state-of-the-art methods on seven benchmark datasets, both visually and via established quantitative measurements. The results demonstrate the superiority of our method over baselines such as t-SNE and UMAP. We also provide out-of-sample extension, large-scale extension, and mini-batch extension for our LapSNE to facilitate dimensionality reduction in various scenarios.

CCS Concepts: • Computing methodologies → Dimensionality reduction and manifold learning.
Additional Key Words and Phrases: Dimensionality reduction, Data visualization, t-SNE, Graph Laplacian

1 INTRODUCTION

High-dimensional data with cluster information are prevalent in various real-world scenarios and domains [6, 50]. Genomics data in high dimensions, such as gene expression profiles or DNA sequence data, can exhibit cluster structures that correspond to different biological conditions or subtypes of diseases. In medical imaging, the images like CT or ultrasound reflect the health or illness of subjects. Thus identifying the two clusters (health and illness) directly is important for diagnosis and treatment. However, the structures of these high-dimensional data are often complicated, especially when the numbers of samples are large [16, 18]. Therefore, extracting potentially useful and understandable patterns from these intricate datasets becomes necessary and helpful. Among all the

*Both authors contributed equally to this research. Jicong Fan is the corresponding author.

This work was partially supported by the Youth program 62106211 of the National Natural Science Foundation of China, the General Program JCYJ20210324130268042 of Shenzhen Fundamental Research, the research funding T00120210002 of Shenzhen Research Institute of Big Data, and the funding UDF1001770 of The Chinese University of Hong Kong, Shenzhen.

Authors’ addresses: Yan Sun, Yi Han, The Chinese University of Hong Kong, Shenzhen, Longxiang 2001, Shenzhen, China, yansun@link.cuhk.edu.cn; Jicong Fan, The Chinese University of Hong Kong, Shenzhen and Shenzhen Research Institute of Big Data, Longxiang 2001, Shenzhen, China, fanjicong@link.cuhk.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
1556-4681/2023/8-ART $15.00
https://doi.org/10.1145/3612932

ACM Trans. Knowl. Discov. Data.
extant methods, dimensionality reduction (DR) is critically important for understanding the structures of large datasets. DR aims to extract or produce informative low-dimensional features from high-dimensional data. Such features can be easily visualized to identify the hidden patterns of the original data, provided that the reduced dimension is low enough, e.g., 3-D or 2-D. The low-dimensional features provide convenience for downstream tasks such as classification and clustering [16, 50] without the curse of dimensionality and often improve the corresponding performances.

DR has been an active and important research topic for more than fifty years. DR methods can be organized into two categories: linear methods and nonlinear methods. Principal Component Analysis (PCA) [25, 37], Multidimensional Scaling (MDS) [43], and Linear Discriminant Analysis (LDA) [19] are well-known linear DR methods. These methods are simple, effective, and well-understood, though they have difficulty in handling data with nonlinear low-dimensional latent structures. In the past decades, many nonlinear dimensionality reduction (NLDR) methods [17, 27, 42, 46, 49] have been proposed to extract low-dimensional nonlinear features or visualize high-dimensional data in 2D or 3D spaces. Well-known NLDR methods include Self-Organized Map (SOM) [27], Kernel PCA (KPCA) [44], Principal Curves [22], Locally Linear Embedding (LLE)[42], Isomap [46], Laplacian Eigenmap (LE) [2], AutoEncoder [12, 24], t-distributed Stochastic Neighbor Embedding (t-SNE) [49], Adaptive Local Linear Discriminant Analysis (ALLDA) [36], and Uniform Manifold Approximation and Projection (UMAP) [32]. Note that LLE, Isomap, and LE are not robust to noise and outliers and their performance on real data are not satisfactory enough. KPCA, AutoEncoder, and stacked AutoEncoders are actually nonlinear feature extraction methods and are not effective in data visualization. Most of the existing dimensionality reduction algorithms share also the same challenge of preserving the structure in the original space. MDDM [54] implicitly retained cluster information by utilizing the class labels. FEDRA [31] was proposed to ameliorate clustering quality but without exploring visualization tasks.

Among the aforementioned NLDR methods, t-SNE developed by van der Maaten and Hinton [49] is arguably one of the most powerful and state-of-the-art methods in a wide range of applications. t-SNE maps the data points to a two- or three-dimensional space, which exhibits the intrinsic data distribution of the original high-dimensional data. Therefore, the low-dimensional embedding always reveals trends, patterns, and outliers. In many practices of scientific research, t-SNE has become an extraordinary tool for data visualization. It is worth mentioning that there have been a few variants of t-SNE [8, 9, 21, 29, 39, 48, 52, 53]. For instance, Yang et al. [53] generalized t-SNE to accommodate various heavy-tailed embedding similarity functions and presented a fixed-point optimization algorithm that can be applied to all heavy-tailed functions. Van Der Maaten [48] proposed to use tree-based algorithms to accelerate t-SNE.

Data visualization is growing critically important nowadays for understanding structure-complicated high-dimensional datasets, and has been recognized as one of the building blocks of data science [14]. In most data visualization tasks, we are interested in interpreting the visualized embeddings [20], and discovering the potential clusters of the data. However, the outlines of clusters generated by t-SNE are often overlapped and obscure, because t-SNE does not explicitly explore and exploit the potential cluster structure of the data. To tackle this problem, in this paper, we propose a new method, LaptSNE, to construct low-dimensional embedding in a cluster-informative manner. Our contributions are three-fold.

- We present a new NLDR algorithm LaptSNE that explores and exploits the potential clusters of the data and produces cluster-informative low-dimensional visualization.
- The objective function of the proposed method involves the eigenvalues of a graph Laplacian computed from the decision variables, which leads to difficulty in solving the optimization problem. We therefore develop an effective algorithm with convergence guarantee to solve the optimization of LaptSNE.
- We provide out-of-sample extension, large-scale extension, and mini-batch extension for our LaptSNE, which facilitate the implementation of LaptSNE in various scenarios.
Experiments on many benchmark datasets (e.g. COIL20 [34] and MNIST [13]) further verify the effectiveness and superiority of our methods. For instance, compared to the vanilla t-SNE, in our method, the boundaries of clusters in the 2D embedding are clearer and the clusters are more compact. Quantitative evaluations such as the k-NN generalization error and clustering NMI also confirm the superiority of our methods. Before presenting the remaining content, we summarize the key notations of this paper in Table 1.

Table 1. Notations

| Symbol | Definition |
|--------|------------|
| \(N\)  | Number of samples |
| \(A\)  | adjacency/affinity/similarity matrix |
| \(D\)  | degree matrix (a diagonal matrix) of adjacency |
| \(I\)  | identity matrix |
| \(L\)  | graph Laplacian matrix |
| \(\text{Tr}(\cdot)\) | trace of matrix |
| \(\sigma_i(\cdot)\) | \(i\)-th eigenvalue of the matrix |
| \(\|\cdot\|_2\) | \(\ell_2\) norm or Euclidean norm of vector |
| \(\|\cdot\|_0\) | number of nonzero elements of vector |
| \(\|\cdot\|_1\) | \(\ell_1\) norm of vector |
| \(\mathbb{I}(\cdot)\) | indicator function, \(\mathbb{I}(\text{True}) = 1\) and \(\mathbb{I}(\text{False}) = 0\) |
| \(\text{KL}\) | Kullback-Leibler divergence |
| \(\mathcal{P}\) | high dimension space |
| \(D\) | dimension in high dimension space |
| \(\{x_i\}_{i \in [N]} \in \mathbb{R}^D\) | high-dimensional data |
| \(P_X\) | symmetric joint probability of high dimension data |
| \(p_{ij}\) | similarity between \(i\) and \(j\) in high dimension space |
| \(L_X\) | graph Laplacian matrix of \(P_X\) |
| \(\mathcal{Q}\) | low dimension space |
| \(d\) | dimension in low dimension space |
| \(\{y_i\}_{i \in [N]} \in \mathbb{R}^d\) | low-dimensional data |
| \(Q_Y\) | symmetric joint probability of low dimension data |
| \(q_{ij}\) | similarity between \(i\) and \(j\) in low dimension space |
| \(L_Y\) | graph Laplacian matrix of \(Q_Y\) |

2 PRELIMINARY KNOWLEDGE

t-SNE is actually a modification of SNE [23]. They aim to preserve the pair-wise similarities from high-dimension space \(\mathcal{P}\) to low-dimension \(\mathcal{Q}\). Here the pair-wise similarities are quantified by neighborhood probability, i.e., the probability of two data points are neighbors mutually. Specifically, given \(\{x_i\}_{i \in [N]} \in \mathbb{R}^D\) (we define \([N] := \{1, 2, \ldots, N\}\) for convenience), SNE and t-SNE find a low-dimensional embedding \(\{y_i\}_{i \in [N]} \in \mathbb{R}^d\) where \(d \ll D\), such that if \(x_i\) and \(x_j\) are close in the original data space, \(y_i\) and \(y_j\) are also close. t-SNE starts by computing the joint probability distribution over all input data, represented by a symmetric matrix \(P_X\). When \(i = j\), \(p_{ij} = 0\). Otherwise, \(p_{ij} = \frac{p_{iij} + p_{jij}}{2N}\).
where

\[ p_{ji} = \frac{\exp \left( -\frac{\|x_i - x_j\|^2}{2\tau_i^2} \right)}{\sum_{\ell \in \{1, \ldots, N\} \neq i} \exp \left( -\frac{\|x_i - x_\ell\|^2}{2\tau_i^2} \right)} \]  

Here \( \tau_i \) denotes the bandwidth of the Gaussian kernel based on user-specified perplexity \( \text{Perp} \) [49]. Similarly, an affinity matrix \( Q_\ell \) in the low-dimensional space can be computed. Compared to SNE, t-SNE replaces the Gaussian kernel in the low dimension with the T-Student kernel in one degree of freedom (same as Cauchy kernel) [45]. The T-Student kernel can help the learning strongly repel dissimilar data points that are modeled by a small pair-wise distance in the low-dimensional representation, thus alleviating the crowding problem. Specifically, in t-SNE, for \( i \neq j \),

\[ q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{\ell \in \{1, \ldots, N\} \neq \ell} \left(1 + \|y_\ell - y_j\|^2\right)^{-1}}. \]  

t-SNE thus searches for \( \{y_i\}_{i \in \{1, \ldots, N\}} \) to minimize the Kullback-Leibler (KL) divergence between the joint distribution of points in the input data space \( P \) and embedding space \( Q \), i.e.,

\[ (y_1, \ldots, y_N) = \arg \min_{y_1, \ldots, y_N} D_{KL}(P_X, Q_Y) = \arg \min_{y_1, \ldots, y_N} \sum_{i \neq j} q_{ij} \log \frac{p_{ij}}{q_{ij}}. \]  

More details can be found in [49].

3 CLUSTER-CONTRACTIVE T-SNE

3.1 Motivation

Although t-SNE is effective in preserving local and global structures of high-dimensional data [1], it does not explore or utilize the potential cluster structures that are prevalent in real datasets [26, 28]. In fields like proteomics, high-dimensional data requires better visualization to reveal clusters or patterns that indicate similarities or differences in protein profiles, aiding in tasks such as disease subtype identification or biomarker discovery. In social network analysis, it is useful to visualize communities within the network that exhibit distinct characteristics. These scenarios highlight the practical aspects of our motivations.

Given that, the potential cluster structure should be a useful prior and exploited if possible, though it is difficult to know the number of clusters of high-dimensional data in advance. If the high-dimensional data indeed consist of multiple clusters, the clusters should be preserved in the low-dimensional embedding. Thus, besides matching the similarity matrix \( P_X \) and \( Q_Y \), we also want to get a similarity graph (from the low-dimensional embedding) of which the edges between different groups have very low weights and the edges within a group have high weights.

However, the aforementioned objective is intractable because we do not know the clusters (or even the number of clusters) of the high-dimensional data. Note that the affinity matrix \( P_X \) in t-SNE is actually a connected graph owing to the use of the Gaussian kernel. Therefore, solving (3) does not guarantee to preserve clusters in the low-dimension space. The optimal solution of t-SNE may preserve clusters if \( P_X \) is sufficiently sparse and the target dimension \( d \) is sufficiently high. But in practice, it is very difficult to find the optimal solution because of the high nonconvexity of (3), \( P_X \) is not very sparse because we often use a Gaussian kernel with moderate hyperparameters, and \( d \) is often 2 or 3. Even if \( P_X \) is not connected but has multiple connected components, the solution of (3) may not preserve the clusters.

ACM Trans. Knowl. Discov. Data.
To handle the problem, we consider using the graph Laplacian defined as

$$L = I - D^{-1/2}AD^{-1/2}. \quad (4)$$

It is actually a symmetric normalized Laplacian matrix. In (4), $A$ (or $Q_Y$ in this paper) is an adjacency (or similarity) matrix, and $D$ is the degree matrix (a diagonal matrix) of $A$ defined as $D_{ii} = \sum_j a_{ij}$. Laplacian matrix has the property of signifying the number of clusters (Proposition 1 in [10]), i.e., the multiplicity $k$ of the eigenvalue 0 of $L$ is equal to the number of connected components $A_1, \ldots, A_k$ in the graph. When the eigenvalue 0 has a multiplicity of $k$, we can observe $k$ completely disconnected clusters. Therefore, we propose to maximize the number of zero eigenvalues of $L$ such that we may identify more clusters from the data.

### 3.2 Proposed Model

In order to preserve both data structure and cluster information, we may consider the following problem instead of (9)

$$\min_{Y} \text{KL}(P_X, Q_Y)$$

subject to $\sum_{i=1}^{N} \mathbb{I}(\sigma_i(L_Y) = 0) = k,$

where $L_Y$ is the symmetric normalized Laplacian matrix computed from $Q_Y$, $\sigma_i(\cdot)$ denotes the $i$-th eigenvalue of matrix, and $\mathbb{I}(\cdot)$ is an indicator function with $\mathbb{I}(\text{True}) = 1$ and $\mathbb{I}(\text{False}) = 0$. In (5), we hope that there are exactly $k$ clusters in the low-dimensional embedding. But it is difficult to know $k$ in advance. We therefore relax (5) to the following regularized problem

$$\min_{Y} \text{KL}(P_X, Q_Y) + \lambda \|\sigma(L_Y)\|_0, \quad (6)$$

where $\sigma(L_Y) = [\sigma_1(L_Y), \sigma_2(L_Y), \ldots, \sigma_N(L_Y)]^\top$ and $\lambda$ is a tuning parameter. $\| \cdot \|_0$ denotes the number of nonzero elements in a vector and $\|\sigma(L_Y)\|_0$ is actually the rank of $L_Y$. In (6), we want to increase the number of zero eigenvalues as large as possible such that the number of clusters in the data is sufficiently large. Problem (6) is NP-hard due to the presence of the $\ell_0$ norm. It is known that the $\ell_p$ (quasi) norms, $\|x\|_p := (\sum_i |x_i|^p)^{1/p}$ ($0 < p \leq 1$) are popular proxies of the $\ell_0$ norm. Particularly, $\|x\|_1$ is a convex relaxation of $\|x\|_0$. Therefore, we relax (6) to

$$\min_{Y} \text{KL}(P_X, Q_Y) + \lambda \sum_{i=1}^{N} \sigma_i^p(L_Y). \quad (7)$$

Note that $\sum_{i=1}^{N} \sigma_i^p(L_Y) \rightarrow \|\sigma(L_Y)\|_0$ when $p \rightarrow 0$. In this study, we only consider the case $p = 1$ for simplicity. Then we arrive at

$$\min_{Y} \text{KL}(P_X, Q_Y) + \lambda \text{Tr}(V_L^\top L_Y V_L), \quad (8)$$

where $V_L \in \mathbb{R}^{N \times N}$ denotes the eigenvectors of $L_Y$. Since $V_L$ depends on $Y$, we have to rewrite (8) as

$$\min_{Y} \text{KL}(P_X, Q_Y) + \lambda \min_{V_L} \text{Tr}(V_L^\top L_Y V_L). \quad (9)$$

Note that in (9), the matrix multiplications in the trace operator are costly because $V_L$ is a square matrix. On the other hand, in (8), we may just shrink the smallest few eigenvalues because in practice the number of clusters in the high-dimensional data is not large (e.g. 10 or 100) and much less than $n$. In view of the two reasons, we solve the following problem instead of (9)

$$\min_{Y} \text{KL}(P_X, Q_Y) + \lambda \min_{V_k} \text{Tr}(V_k^\top L_Y V_k), \quad (10)$$
where $V_k \in \mathbb{R}^{N \times \hat{k}}$ and $\hat{k}$ is an estimated number of potential clusters in $X$. This is exactly our proposed method LapTSNE.

We can use domain or prior knowledge, if they are available, to determine $\hat{k}$ for LapTSNE. Otherwise, we can use the largest eigengap of the Laplacian matrix of $P_X$ to estimate the number of clusters as below. Since the number of zero eigenvalues of graph Laplacian in the original space implies the potential number of clusters [50], we first investigate the eigenvalue sequence of the Laplacian matrix corresponding to $P_X$ in the original data space. Specifically, we look for the largest, or in other words, the most obvious eigengap in the ascending sequence of the graph Laplacian eigenvalues:

$$\hat{k} = \arg \max_i \sigma_{i+1}(L_X) - \sigma_i(L_X)$$

(11)

where $L_X$ is the symmetric normalized Laplacian matrix computed from $P_X$. $\sigma_i(\cdot)$ denotes the $i$-th eigenvalue of matrix.

It is worth mentioning that there are a few techniques [11, 30, 41] to determine the number of clusters. The goal of these works is clustering, not dimensionality reduction and visualization. In contrast, our goal is dimensionality reduction and visualization. Our LapTSNE is built upon t-SNE, where we obtain a similarity $P_X$ which presents the local connectivity of the data. $P_X$ can be regarded as a graph. Therefore, in terms of $P_X$, the most suitable method to estimate the number of clusters of the data or components of the corresponding graph is utilizing the eigen-gap of the graph Laplacian computed from $P_X$.

### 3.3 Optimization

Note that it is non-trivial to solve (10) due to the presence of $V_k$. We present a majorization-minimization [47] algorithm for (10). Specifically, at iteration $t$, we solve

$$\min_Y KL(P_X, Q_Y) + \lambda \text{Tr}(V_{t-1}^T L_Y V_{t-1}),$$

(12)

where $V_{t-1}$ is the eigenvectors of $L_{Y_{t-1}}$, corresponding to the smallest $\hat{k}$ eigenvalues. There is no need to obtain the exact solution of (12) and hence we propose to just update $Y$ by gradient descent. For convenience, let

$$L_1(Y) = KL(P_X, Q_Y),$$

(13)

$$L_2(Y) = \lambda \min_{V_k^T V_k = I} \text{Tr}(V_k^T L_Y V_k),$$

(14)

$$\hat{L}_2(Y) = \lambda \text{Tr}(V_{t-1}^T L_Y V_{t-1}),$$

(15)

$$L = L_1(Y) + L_2(Y),$$

(16)

$$\hat{L} = L_1(Y) + \hat{L}_2(Y).$$

(17)

Then the one-step gradient descent is given as

$$Y_t \leftarrow Y_{t-1} - \alpha \nabla \hat{L}(Y)$$

(18)

where $\alpha$ is the step size and $\nabla \hat{L}$ denotes the derivative of $L$ with respect to $Y$. The procedures are summarized into Algorithm 1.

**PROPOSITION 1.** Suppose there exists a positive constant $L$ such that $\|\nabla L(Y) - \nabla L(Y')\|_F \leq L\|Y - Y'\|_F$. Let $Y_0, Y_1, \ldots, Y_T$ be the sequence generated by Algorithm 1 with $\alpha \leq \frac{2}{L}$. Then:

(a) $L(Y_t) - L(Y_{t-1}) \leq \frac{2L}{\alpha^2} \|Y_t - Y_{t-1}\|_F \leq 0$;

(b) $L(Y_T) \leq L(Y_0) - \frac{2L}{\alpha^2} \sum_{t=1}^T \|Y_t - Y_{t-1}\|_F$. 

ACM Trans. Knowl. Discov. Data.
Algorithm 1 Optimization for LapSNE

**Input:** dataset $X \in \mathbb{R}^{N \times D}$, target dimension $d$, estimate of cluster numbers $\hat{k}$, tuning parameter $\lambda$, perplexity $\delta_{\text{perp}}$, step size $\alpha$, maximum iteration $T$

1. compute pairwise affinities $p_{ij}$ with perplexity $\delta_{\text{perp}}$ (refer to [49])
2. set $P_X = \{p_{ij}\}_{(i,j) \in [N] \times [N]}$ where $p_{ij} = \frac{e^{-\frac{d(X_i, X_j)}{\delta_{\text{perp}}}}}{\left(\sum_{k \neq i} e^{-\frac{d(X_k, X_j)}{\delta_{\text{perp}}}}\right)}$
3. compute the graph Laplacian $L_X$ of $P_X$
4. Initialization: $Y_0 \leftarrow \{V_1(L_X), \cdots, V_d(L_X)\}$
5. for $t = 1, 2, \ldots, T$ do
6. compute low-dimensional affinities $\{q_{ij}\}_{(i,j) \in [N] \times [N]}$
7. compute gradient of KL divergence $\nabla L_1(Y)$
8. compute gradient of regularization $\nabla \hat{L}_2(Y)$
9. set $Y_t \leftarrow Y_{t-1} - \alpha \left(\nabla L_1(Y) + \nabla \hat{L}_2(Y)\right)$
10. end for

**Output:** cluster-contractive embedding $Y_T \in \mathbb{R}^{N \times d}$

Note that the algorithm can be accelerated by using momentum, i.e.,

$$Y_t \leftarrow Y_{t-1} - \alpha \left(\nabla L_1(Y) + \nabla \hat{L}_2(Y)\right) + \beta(t) \left(Y_{t-1} - Y_{t-2}\right), \tag{19}$$

where $\beta(t)$ is the momentum parameter. Proving the convergence of Algorithm 1 with momentum is out of the scope of this paper and can be future work.

3.4 Gradient of Laplacian-Composited Objective

It is non-trivial to compute the gradient of $\nabla L(Y)$ because $Y$ involves the symmetric normalized Laplacian. Here we elaborate the computation.

First, the gradient of $L_2(Y)$ ($\lambda$ is omitted for convenience) with respect to $Y$ can be expressed as

$$\frac{\partial \text{Tr}(V^TV)}{\partial Y} = \sum_{m,n} \left(\sum_{i,j} (VV^T)_{ij} \frac{\partial q_{ij}}{\partial q_{mn}}\right) \frac{\partial q_{mn}}{\partial Y}. \tag{20}$$

The symmetric normalized Laplacian matrix $L$ is constructed via $Q_Y$ (note that $q_{ii} = 0$, $\forall i$), and the matrix representation of $L$ could be further simplified to the equations in the right:

$$L = \begin{bmatrix}
    1 - \frac{q_{11}}{\left(\sum_b q_{ib} \sum_a q_{a1}\right)^{\frac{1}{2}}} & \cdots & -\frac{q_{1n}}{\left(\sum_b q_{ib} \sum_a q_{a1}\right)^{\frac{1}{2}}} \\
    \cdots & \cdots & \cdots \\
    -\frac{q_{mn}}{\left(\sum_b q_{ib} \sum_a q_{a1}\right)^{\frac{1}{2}}} & \cdots & 1 - \frac{q_{nn}}{\left(\sum_b q_{ib} \sum_a q_{a1}\right)^{\frac{1}{2}}} \\
\end{bmatrix},$$

$$L_{ij} = \begin{cases} 
    1 - \left(\sum_b q_{ib}\right)^{-\frac{1}{2}} q_{ij} \left(\sum_a q_{aj}\right)^{-\frac{1}{2}} & i = j, \\
    -\left(\sum_b q_{ib}\right)^{-\frac{1}{2}} q_{ij} \left(\sum_a q_{aj}\right)^{-\frac{1}{2}} & i \neq j.
\end{cases}$$
These equations will help us discuss the gradient of $L_{ij}$ with respect to $q_{mn}$ in different situations. For each similarity $q_{mn}$, there are actually three gradient components of the graph Laplacian matrix ($L$). It follows that,

$$\sum_{i,j} (VV^T)_{ij} \frac{\partial L_{ij}}{\partial q_{mn}} = \sum_{i \neq m} (VV^T)_{in} \frac{\partial L_{in}}{\partial q_{mn}} + \sum_{j \neq n} (VV^T)_{mj} \frac{\partial L_{mj}}{\partial q_{mn}} + (VV^T)_{mn} \frac{\partial L_{mn}}{\partial q_{mn}}$$

$$= \sum_i \frac{1}{2} (VV^T)_{in} (\sum_b q_{ib})^{-\frac{1}{2}} q_{in} (\sum_a q_{an})^{-\frac{1}{2}} + \sum_j \frac{1}{2} (VV^T)_{mj} (\sum_b q_{mb})^{-\frac{1}{2}} q_{mj} (\sum_a q_{aj})^{-\frac{1}{2}} - \frac{1}{2} (VV^T)_{mn} (\sum_b q_{mb})^{-\frac{1}{2}} (\sum_a q_{an})^{-\frac{1}{2}}.$$

(21)

Next, we introduce three auxiliary matrices $U^0$, $U^1$, and $U^2$. They represent the different components of the formula above. Specifically, let

$$U^0 := \left[ \begin{array}{ccc} \cdots & \cdots & \cdots \\ \cdots & u_n^0 & \cdots \\ \cdots & \cdots & \cdots \end{array} \right]_{N \times N}, \quad U^1 := \left[ \begin{array}{c} \cdots \\ u_m^1 \\ \cdots \end{array} \right]_{N \times N},$$

where

$$u_n^0 = \sum_i \frac{1}{2} (VV^T)_{in} \frac{q_{in}}{(\sum_b q_{ib})^{\frac{1}{2}} (\sum_a q_{an})^{\frac{1}{2}}} 1, \quad n = 1, \ldots, N$$

and

$$u_m^1 = \sum_j \frac{1}{2} (VV^T)_{mj} \frac{q_{mj}}{(\sum_b q_{mb})^{\frac{1}{2}} (\sum_a q_{aj})^{\frac{1}{2}}} 1, \quad m = 1, \ldots, N$$

with all-one vector $1 \in \mathbb{R}^N$. Further we let

$$U^2 := (-\frac{1}{2} (VV^T)_{mn} (\sum_b q_{mb})^{-\frac{1}{2}} (\sum_a q_{an})^{-\frac{1}{2}})_{m,n=1}^N.$$

(23)

Then, we can factorize the chain of gradients into point-wise multiplication. Invoking (22) and (23) into (20), we obtain

$$\frac{\partial \text{Tr}(Y^TLV)}{\partial Y} = \sum_{m,n} \frac{\partial \text{Tr}(Y^TLV)}{\partial q_{mn}} \frac{\partial q_{mn}}{\partial Y}$$

$$= \sum_{m,n} (U^0_{mn} + U^1_{mn} + U^2_{mn}) \odot \frac{\partial q_{mn}}{\partial Y}.$$ 

(24)

Based on different similarity measurement, we discuss $\frac{\partial q_{mn}}{\partial Y}$ accordingly. In this study, we consider the Gaussian kernel and T-Student kernel, though the latter has better performance in our numerical studies. For the Gaussian kernel ($K = (k_{mn})_{m,n=1}^N$) with a constant sigma $\sigma$, the derivative of each row in $Y$ is

$$\frac{\partial k_{mn}}{\partial y_a} = \begin{cases} k_{mn} \cdot \left( -\frac{1}{2\sigma^2} \right) \cdot 2(y_a - y_m) & n = a, \\
\quad k_{mn} \cdot \left( -\frac{1}{2\sigma^2} \right) \cdot 2(y_a - y_n) & m = a. 
\end{cases}$$

(25)

ACM Trans. Knowl. Discov. Data.
Algorithm 2 Large-scale extension of LapSNE

**Input:** dataset $X \in \mathbb{R}^{N \times D}$, target dimension $d$, number of landmark points $S$
1: perform k-means to partition $X$ into $S$ clusters
2: form a dataset $X_S \in \mathbb{R}^{S \times D}$ using the data points in $X$ closest to the $S$ cluster centers
3: perform LapSNE (Algorithm 1) on $X_S$ to get $Y_S \in \mathbb{R}^{S \times d}$
4: train a neural network (with parameters $W$) $f_W : \mathbb{R}^D \rightarrow \mathbb{R}^d$ using $\{X_S, Y_S\}$
5: obtain $Y = f_W(X)$

**Output:** cluster-contractive embedding $Y \in \mathbb{R}^{N \times d}$

Since $q_{mn} = k_{mn}$, substituting (25) into (24), we have
$$
\frac{\partial \text{Tr}(V^T LV)}{\partial Y} = \frac{2}{\sigma^2} (UY - C \odot Y),
$$
where $U := (U^0 + U^1 + U^2) \odot K \in \mathbb{R}^{N \times N}$ and $C := U_{N \times d} \in \mathbb{R}^{N \times d}$.

If we replace Gaussian kernel by T-Student kernel ($T = \{t_{ij}\}_{i,j=1}^N$) and only take the numerator part as the similarity measure, then $q_{mn} = t_{mn} = (1 + \|y_m - y_n\|^2)^{-1}$. The latent matrix $U \in \mathbb{R}^{N \times N}$ takes the form of
$$
U = (U^0 + U^1 + U^2) \odot T \odot T.
$$
Then we arrive at
$$
\frac{\partial \text{Tr}(V^T LV)}{\partial Y} = 2(UY - C \odot Y)
$$
where $C = U_{N \times d} \in \mathbb{R}^{N \times d}$.

It is worth mentioning that the computation of the gradient we have introduced can be adapted to other problems where the eigenvalues of the symmetric normalized Laplacian are functions of decision variables.

### 3.5 Out-of-Sample and Large-Scale Extensions of LapSNE

The quadratic time and space complexities of LapSNE and t-SNE prevent their application to large-scale datasets. In addition, it is difficult to use the learned models of them to reduce the dimension of new data. Here we provide an out-of-sample extension for LapSNE.

Suppose we have already reduced the dimension of $X$ to 2 using our LapSNE and we want to reduce the dimension of some new samples $X_{\text{new}}$ drawn from the same distribution as $X$. There is no need to perform LapSNE on $\{X, X_{\text{new}}\}$ again. We can just learn a nonlinear mapping $f$ from $x$ to $y$ using neural networks on $\{X, Y\}$, where $Y$ denotes the 2-D embedding given by LapSNE. Then we perform $f$ on $X_{\text{new}}$ to get the low-dimensional embedding $Y_{\text{new}}$.

It is worth mentioning that the out-of-sample extension can also be applied to handle large-scale datasets. Specifically, we perform k-means clustering (with a large enough $k$, e.g. 1000) on a large dataset $X$ to get the cluster centers, which are regarded as some landmark data points, forming a small subset of $X$. One may also consider using random sampling (without replacement) to generate the subset. Then we perform LapSNE on the landmark points and use the out-of-sample extension to get the low-dimensional embedding of the large dataset. See Algorithm 2.

### 3.6 Mini-Batch Extension of LapSNE

Besides the out-of-sample extension, mini-batch optimization could also serve as an alternative to reduce the computational complexity on especially large datasets. During the optimization procedure, instead of updating the entire $Y$, we can just update a few rows of $Y$ in each iteration. Specifically, at iteration $t$, let $Y_{Ot} \in \mathbb{R}^{m \times d}$ be a
mini batch of $Y$ indexed by $\Omega_t \subset [N]$ and let $\Omega^*_t$ be the index set of the $\kappa$-nearest neighbors (according to $\mathbf{P}_Y$) of $Y_{\Omega_t}$ in $Y_{[N]/\Omega_t}$. Denote $\Omega_t = \Omega_t \cup \Omega^*_t$. Then we update $Y_{\Omega_t}$ at iteration $t$ such that the local connectivity in the original data space can be preserved:

$$Y_{\Omega_t} \leftarrow Y_{\Omega_t} - \alpha \nabla \hat{L}(Y_{\Omega_t}).$$

(29)

Note that the computation of the gradient $\nabla \hat{L}(Y_{\Omega_t})$ involves evaluating $\mathbf{Q}_Y$, which can be implemented via updating the rows and columns of $\mathbf{Q}_Y$ with indices $\Omega_t$. Since $|\Omega_t| \leq \kappa m \ll N$, the computational complexity is reduced significantly. We call this approach LapSNE-Mini.

3.7 Hyperparameters and Complexity Analysis

As described in Algorithm 1, compared with tSNE, it seems that our LapSNE algorithm has two more hyperparameters $\hat{k}$ (estimated number of potential clusters) and $\lambda$ (the regularization weight serving as contractive strength). Actually, LapSNE at the same time eliminated a few hyperparameters required in t-SNE, such as the exaggerate stage and the corresponding exaggeration rate. Besides, the foregoing hyperparameters are hard to adjust and can lead to non-convergence in t-SNE.

Note that in LapSNE, $\hat{k}$ is not necessarily equal or very close to the true number of clusters $k$. When $\hat{k} < k$, LapSNE only shrinks partial clusters. When $\hat{k} > k$, the overestimated number of classes makes LapSNE try to explore the nuanced subclass information from the original categories. If the subclass structures are not significant, the minimization for the eigenvalues $\sigma_{k+1}, \ldots, \sigma_{\hat{k}}$ will have less impact on the result. With a moderate $\hat{k} \approx k$, one may observe a clearly cluster-contractive structure in the low-dimensional embeddings. For the effect of contractive strength $\lambda$, suppose the data have been reduced to a well-clustered low-dimensional representation, the contractive strength is not necessarily large. Whereas the low-dimensional representations are overlapped, LapSNE requires a larger hyperparameter $\lambda$ to shrink data into different groups.

According to Section 3.3 and Section 3.4, the time complexity (per iteration) and space complexity of our LapSNE are $O(\hat{k}N^2)$ and $O(DN + N^2)$ respectively. Note that we only need to compute the smallest $\hat{k}$ eigenvalues and eigenvectors of $L$, which is efficient. In Table 2, we report the time costs of LapSNE and LapSNE-Mini in comparison to t-SNE on COIL20 and COIL100 datasets. Indeed, LapSNE and LapSNE-Mini are slower than t-SNE but they can provide better dimensionality reduction results, which will be shown in Section 4.

### Table 2. Running time comparison between LapSNE and t-SNE over 100 iterations

| Dataset | COIL20 | COIL100 |
|---------|--------|---------|
| LapSNE  | 60.4s  | 14876.7s|
| LapSNE-Mini | 20.1s  | 887.0s  |
| t-SNE   | 8.4s   | 147.9s  |

4 EXPERIMENTS

4.1 Datasets, Baselines, and Evaluation Metrics

We test our LapSNE with T-Student kernel on seven real-world datasets detailed below.

- **Waveform** [4] dataset contains 5,000 samples generated from 3 classes of waves with 21-dimensional attributes, all of which include noise.
- **PenDigits** [5] is a set of 1,797 grayscale images of digits. The size of each image is 8 × 8 and we directly flatten it into a 64-dimensional vector.
- **COIL 20** [34] is a set of 1,440 grayscale images consisting of 20 objects under 72 different rotations. We flatten each image (128x128) into a 16,384-dimensional vector.
- **COIL 100** [35] is a set of 7,200 color images consisting of 100 objects under 72 different rotations. Each image is a color image of size 128x128. We convert the images to grayscale and resize them to 32x32. Then we flatten each image into a 1,024-dimensional vector.
- **HAR** [40] (Human Activity Recognition) is a dataset of 10,299 instances with 561-dimensional attributes. It was built from the recordings of 30 subjects performing activities of daily living.
- **MNIST** [13] is a dataset of 28x28 pixel grayscale images of handwritten digits (from 0 to 9). There are 70,000 images in total. We treat these images as 784-dimensional vectors.
- **Fashion-MNIST** [51] is a dataset of 28x28 pixel grayscale images of 10 kinds of fashion items, such as clothing and bags. There are 70,000 images. We treat them as 784-dimensional vectors.

We see that Waveform, COIL20, COIL100, and PenDigits are small datasets with no more than 10,000 samples while HAR, MNIST, and Fashion-MNIST are large datasets with more than ten thousand samples. We will show the results on the small datasets and large datasets separately. The proposed LapTSNE and LapTSNE-Mini are compared with the following baselines: *Vanilla t-SNE* [49], *UMAP* [32] and *Laplacian Eigenmaps* [3].

The implementation of our LapTSNE and LapTSNE-Mini are based on the scikit-learn package and gradient descent optimization and details are elaborated below.

- **Vanilla t-SNE** [49]: We implement t-SNE using the TSNE module in scikit-learn package [38]. To ensure a fair comparison, we remove the trick of the exaggeration stage and set desired perplexity and the number of iterations the same as LapTSNE.
- **UMAP** [32]: We perform UMAP visualization using the latest umap-learn package [33]. Specifically, we remain the parameters as default and set the number of iterations the same as LapTSNE.
- **Laplacian Eigenmaps** [3]: We implement it as SpectralEmbedding module in the scikit-learn package [38], where parameters are set as default.

We will present both qualitative results and quantitative results in the comparison studies. We consider the following evaluation metrics.

- **k-nearest neighbor classifier accuracy**: It measures the quantitative performance of the preservation of cluster information in the original space. With the hyper-parameter $k$ varying, we can also consider how structure preservation changes from purely local to more global. When computing the errors, the labels collected beforehand are assumed to contain the inherent cluster information. The metric has been used in many previous works of dimensionality reduction such as [32].

- **Normalized Mutual Information (NMI)** [15]: NMI is a widely-used metric for evaluating the performance of clustering algorithms. It scales between 0 (no mutual information) and 1 (perfect correlation). In this study, we compute NMI based on the ground-truth label and the clustering result of k-means on the low-dimensional embedding $Y$, where $k$ is set as the number of actual classes.

- **Silhouette Coefficient (SC)** [41]: SC ranges from -1 to 1, measuring how distinct or well-separated a cluster is from other clusters. The score of SC is higher when clusters are dense and well separated. Similar to NMI, we compute SC using the result of k-means on $Y$.

- **Davies-Bouldin Index (DBI)** [11]: DBI calculates the ratio of within-cluster and between-cluster distances, and therefore, the lower the score the better separation there is between clusters. Similar to NMI, we compute DBI using the result of k-means on $Y$. ACMSkDD
4.2 Results on Small Datasets

As mentioned in Section 3.2, we first investigate the eigenvalue sequence of the Laplacian matrix corresponding to $P_X$ in the original data space. For Waveform, PenDigits, COIL20, and COIL100 datasets, the Gaussian kernel with perplexity of 25 was used to construct $P_X$. As shown in the first row of Figure 1, for COIL20, the gap between eigenvalues 19 and 20 is more distinguishable than others, although the gaps between eigenvalues 6 and 7 and between eigenvalues 8 and 9 are also large. As analyzed in Section 3.7, we prefer an overestimated cluster number and hence we set $\hat{k} = 19$ for COIL20, even though the true cluster number is 20. Similarly, the estimated number of potential clusters in Waveform, PenDigits, and COIL100 datasets are 19, 11, and 7 respectively. We do not require $\hat{k} = k$ because it is unrealistic in practice and our $\text{LapTSNE}$ is not sensitive to $\hat{k}$. We will illustrate this point by using a wide range of $\hat{k}$ later.
Fig. 2. Row 1: Smallest eigenvalues of graph Laplacian of each dataset (From left to right: MNIST, 14-th gap is large; Fashion-MNIST, 12-th gap is large; HAR, 10-th gap is large). Rows 2-4: Qualitative comparison of LaptSNE, LaptSNE-Mini and t-SNE in visualizing MNIST, Fashion-MNIST and HAR.

As for the results, we first plot the two-dimensional embedding colored with the ground-truth label in Figure 1. We claim that the quality of embedding produced by LaptSNE and LaptSNE-Mini are better than t-SNE for the four small datasets, although the three algorithms are all powerful in preserving the local and global structures from the original space. LaptSNE and LaptSNE-Mini can shrink the clusters into compact clusters in a powerful cluster-contractive manner.

In Table 3, we quantitatively compare LaptSNE, LaptSNE-Mini, t-SNE, UMAP, and Eigenmaps embeddings with respect to the four evaluation metrics. We see that LaptSNE performs remarkably better on PenDigits and COIL20 among all the dimensional reduction methods. It performs at least as well as t-SNE on COIL100. Note that on Waveform, in terms of the SC and DBI metrics, Eigenmaps outperforms other methods, but in terms of NMI, Eigenmaps is outperformed by our LaptSNE. One possible reason is that SC and DBI are not effective...
enough to quantify the with-class and between-class differences of this dataset. In addition, NMI is more reliable than SC and DBI because it utilizes true labels.

Evidently, the embedding quality of LapTSNE is better compared with t-SNE and UMAP at both local and non-local scales in terms of PenDigits and COIL20. As for COIL100, it provides largely comparable performance in embedding at local scales, but performs superior at non-local scales. Besides, as shown in Table 2 and Table 3, the proposed mini-batch based LapTSNE could remarkably save computational time while maintaining good performance. In this way, LapTSNE-Mini could be considered as an efficient alternative when dealing with large datasets.

Table 3. Comparison between LapTSNE, LapTSNE-Mini, t-SNE, UMAP and Eigenmaps on small datasets: 1) k-NN classifier accuracy for different k; 2) clustering performances in terms of NMI, SC, and DBI based on the hyperparameter \( \hat{k} \). Note: ↑ means the higher is better. The best value in each case is marked in bold.

|          | score | LapTSNE | LapTSNE-Mini | t-SNE | UMAP | Eigenmaps |
|----------|-------|---------|--------------|-------|------|-----------|
| PenDigits| 10-nn | 0.990   | 0.989        | 0.977 | 0.988| 0.828     |
|          | 20-nn | 0.987   | 0.989        | 0.973 | 0.983| 0.816     |
|          | 40-nn | 0.975   | 0.966        | 0.956 | 0.972| 0.805     |
|          | 80-nn | 0.970   | 0.952        | 0.948 | 0.956| 0.785     |
|          | NMI   | 0.9084  | 0.8793       | 0.7148| 0.8981| 0.7869    |
|          | SC    | 0.7535  | 0.6742       | 0.4754| 0.6010| 0.6976    |
|          | DBI   | 0.4064  | 0.4531       | 0.7121| 0.5688| 0.4532    |

|          | 10-nn | 0.986   | 0.897        | 0.934 | 0.901| 0.773     |
|          | 20-nn | 0.943   | 0.895        | 0.901 | 0.885| 0.744     |
|          | 40-nn | 0.909   | 0.874        | 0.857 | 0.877| 0.694     |
|          | 80-nn | 0.860   | 0.844        | 0.789 | 0.822| 0.638     |
|          | NMI   | 0.8723  | 0.8723       | 0.8286| 0.8489| 0.5527    |
|          | SC    | 0.7947  | 0.7947       | 0.5105| 0.5689| 0.5760    |
|          | DBI   | 0.3251  | 0.3251       | 0.6866| 0.5997| 0.6394    |

|          | 10-nn | 0.937   | 0.936        | 0.894 | 0.832| 0.633     |
|          | 20-nn | 0.900   | 0.907        | 0.856 | 0.810| 0.582     |
|          | 40-nn | 0.860   | 0.861        | 0.808 | 0.779| 0.524     |
|          | 80-nn | 0.798   | 0.792        | 0.718 | 0.735| 0.468     |
|          | NMI   | 0.8601  | 0.8778       | 0.8694| 0.8785| 0.6727    |
|          | SC    | 0.6026  | 0.6001       | 0.5682| 0.5126| 0.4280    |
|          | DBI   | 0.5497  | 0.5827       | 0.5841| 0.6507| 0.7500    |

|          | 10-nn | 0.859   | 0.859        | 0.849 | 0.850| 0.822     |
|          | 20-nn | 0.850   | 0.853        | 0.842 | 0.849| 0.834     |
|          | 40-nn | 0.845   | 0.844        | 0.836 | 0.843| 0.820     |
|          | 80-nn | 0.838   | 0.841        | 0.838 | 0.841| 0.812     |
|          | NMI   | 0.3863  | 0.3686       | 0.3506| 0.3709| 0.3679    |
|          | SC    | 0.4658  | 0.4811       | 0.4624| 0.5121| 0.6821    |
|          | DBI   | 0.7692  | 0.6980       | 0.7349| 0.6549| 0.4113    |
4.3 Results on Large Datasets
In the same way mentioned above, the largest eigengap of HAR, MNIST, and FMNIST are estimated, which turns out to be 10, 14, and 12 respectively, shown by the first row of Figure 2. Since these are quite large datasets, we use the out-of-sample extension of LapTSNE proposed in Section 3.5. The neural network consists of 4 hidden layers of size $W = \left\{ \left\lfloor \frac{2}{3} \right\rfloor D, \left\lfloor \frac{1}{2} \right\rfloor D, \left\lfloor \frac{1}{3} \right\rfloor D, \left\lfloor \frac{1}{5} \right\rfloor D \right\}$ where $D$ is the dimension of $X$. The activation functions in the hidden layers are ReLU. The number of landmark points is 10000.

As shown in Figure 2, LapTSNE is more effective in capturing the latent cluster information than t-SNE. For instance, in the result of LapTSNE, the global relationships among different clusters of the digits in MNIST are more clearly identified and the clusters themselves are more compact. In Table 4 we present the NMI, DC, DBI, and k-NN scores on MNIST, Fashion-MNIST, and HAR. LapTSNE established comparable performance with t-SNE on Fashion-MNIST at both local and non-local scales, but notably better than UMAP. For MNIST, LapTSNE is slightly better than other algorithms when $k$ varies from 100 to 400, but has significantly higher accuracy for $k$ values of 800.

As evidenced by this comparison, LapTSNE provides largely comparable performance across large datasets both qualitatively and quantitatively. The success of LapTSNE may be owing to the theory proved in [7] that the low-dimensional map in t-SNE algorithm converges cluster-wise towards some limit points in $\mathbb{R}^2$, only depending on the initialization, and each associated with a connected component of the underlying graph. According to our experiments, the Laplacian regularization term helps strengthen this process without undermining the fine properties of t-SNE.

4.4 Sensitivity Analysis
We analyze the sensitivity of LapTSNE to the hyperparameters $\lambda$ and $\hat{k}$ qualitatively and quantitatively. For example, Figure 3 presents the k-NN score ($k=20$) of LapTSNE with different $\hat{k}$ and $\lambda$ on COIL20. We notice that a large scale of $\lambda$ (e.g. 1e-1) would crush the performance but smaller ones from 1e-2 to 1e-6 could yield results even better than that of the vanilla t-SNE ($\lambda = 0$). The hyperparameter $\hat{k}$ may fluctuate the performance in a non-monotonic manner. For the order of $\hat{k}$ where a large eigengap exists, the k-NN scores are the best. This indicates the importance of tuning the estimated number of potential clusters $\hat{k}$ according to the eigenvalues of graph Laplacian in the original space. When tuning $\hat{k} = 19$ and $\lambda = 1e-4$, LapTSNE has the highest k-NN score ($k=20$) as Table 3 shows, and the embedding layout is cluster-contractive, which consists of many circles and lines in Figure 1.

We analyze the sensitivity of LapTSNE to the hyperparameters $\lambda$ and $\hat{k}$ qualitatively and quantitatively as Figure 3. Besides, we can tell from Figure 4 that the clusters shrink with the increase of $\hat{k}$. Accordingly, the experiment further supports the interaction effect between the estimated number of potential clusters $\hat{k}$ and the tuning parameter $\lambda$: relatively larger $\lambda$ and $\hat{k}$ can produce more compact clusters.
Fig. 3. K-NN scores (k=20) on the COIL20 dataset: LapSNE result with different $\lambda$ (ranging from 0 to 1e-1) and estimated number of potential cluster $\hat{k}$ (ranging from 5 to 30).

Fig. 4. Variation of LapSNE embedding with different hyperparameters $\hat{k}$ and $\lambda$ on the PenDigits dataset.

Fig. 5. A comparison of t-SNE and LapSNE in the first 50 iterations on PenDigits. (Left: t-SNE, Right: LapSNE; Row 1 is the initial stage, Row 2 to Row 4 are iteration 10, 30, 50)

Fig. 6. Part of the eigenvalues of T-Student kernel graph in the 2-dimensional embeddings of LapSNE and t-SNE (in different iterations) on PenDigits. The green curve stands for the 11-th eigenvalue.
Table 4. Comparison between LapSNE, LapSNE-Mini, t-SNE, UMAP and Eigenmaps on large datasets: 1) $k$-NN classifier accuracy for different $k$; 2) clustering performances in terms of NMI, SC and DBI based on the hyperparameter $\hat{k}$. Note: ↑ means the higher is better. The best value in each case is marked in bold.

|       | score | LapSNE | LapSNE-Mini | t-SNE | UMAP | Eigenmaps |
|-------|-------|--------|-------------|-------|------|-----------|
| MNIST |       |        |             |       |      |           |
| 100-nn↑ | 0.943 | 0.940 | 0.941 | 0.943 | 0.695 |
| 200-nn↑ | 0.940 | 0.939 | 0.937 | 0.941 | 0.680 |
| 400-nn↑ | 0.937 | 0.934 | 0.933 | 0.935 | 0.659 |
| 800-nn↑ | 0.935 | 0.933 | 0.932 | 0.930 | 0.635 |
| NMI ↑  | 0.7498 | 0.7340 | 0.6221 | 0.6892 | 0.3101 |
| SC ↑   | 0.5886 | 0.5809 | 0.4178 | 0.4949 | 0.3464 |
| DBI ↓  | 0.5592 | 0.6038 | 0.7268 | 0.7476 | 0.8447 |
| F-MNIST|       |        |             |       |      |           |
| 100-nn↑ | 0.787 | 0.783 | 0.782 | 0.754 | 0.695 |
| 200-nn↑ | 0.768 | 0.760 | 0.771 | 0.738 | 0.675 |
| 400-nn↑ | 0.762 | 0.758 | 0.762 | 0.718 | 0.652 |
| 800-nn↑ | 0.753 | 0.753 | 0.754 | 0.697 | 0.636 |
| NMI ↑  | 0.6009 | 0.6004 | 0.5905 | 0.6051 | 0.4595 |
| SC ↑   | 0.6299 | 0.6298 | 0.6113 | 0.5325 | 0.4789 |
| DBI ↓  | 0.4679 | 0.4680 | 0.5035 | 0.6557 | 0.7410 |
| HAR    |       |        |             |       |      |           |
| 100-nn↑ | 0.931 | 0.918 | 0.915 | 0.895 | 0.769 |
| 200-nn↑ | 0.924 | 0.906 | 0.909 | 0.881 | 0.746 |
| 400-nn↑ | 0.910 | 0.893 | 0.897 | 0.860 | 0.739 |
| 800-nn↑ | 0.884 | 0.870 | 0.840 | 0.852 | 0.722 |
| NMI ↑  | 0.734 | 0.6015 | 0.4601 | 0.6878 | 0.6289 |
| SC ↑   | 0.4667 | 0.4059 | 0.3893 | 0.5275 | 0.7657 |
| DBI ↓  | 0.7973 | 0.8046 | 0.8446 | 0.7376 | 0.4756 |

4.5 Trajectory of LapSNE

We discover that LapSNE outperforms t-SNE in the early stage of iterations. As Figure 5 illustrates, the lower-dimensional embeddings of LapSNE start from the same initialization as t-SNE but expand in a cluster-contractive manner, which leads to a well-clustered layout in the end.

Note that, shown in Figure 5, the clusters in the embeddings of LapSNE are more distinguishable than those in the embeddings of t-SNE. This is consistent with the fact that, shown in Figure 6, the gap between the 11-th and 12-th eigenvalues of the lower-dimensional representation’s graph Laplacian is increasingly large during the iteration of LapSNE, compared with vanilla t-SNE.

5 CONCLUSIONS

This work provided a new NLDR method called LapSNE and its several extensions for high-dimensional data visualization. The proposed methods generate cluster-informative low-dimensional embedding and outperform t-SNE and UMAP visually and quantitatively on seven benchmark datasets. It should be pointed out that the proposed method can be adapted to other NLDR methods such as UMAP to boost the visualization performance.

REFERENCES

[1] Sanjeev Arora, Wei Hu, and Pravesh K. Kothari. 2018. An Analysis of the t-SNE Algorithm for Data Visualization. In Proceedings of the 31st Conference On Learning Theory (Proceedings of Machine Learning Research, Vol. 75), Sébastien Bubeck, Vianney Perchet, and Philippe ACM Trans. Knowl. Discov. Data.
[32] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018).

[33] Leland McInnes, John Healy, and James Melville. 2018. UMAP: Uniform Manifold Approximation and Projection. The Journal of Open Source Software 3, 29 (2018), 861.

[34] Sameer A Nene, Shree K Nayar, and Hiroshi Murase. 1996. Columbia university image library (coil-20). Technical Report CUCS-005-96 (1996).

[35] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. 1996. Columbia object image library (coil-100). (1996).

[36] Feiping Nie, Zheng Wang, Rong Wang, Zhen Wang, and Xuelong Li. 2020. Adaptive local linear discriminant analysis. ACM Transactions on Knowledge Discovery from Data (TKDD) 14, 1 (2020), 1–19.

[37] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science 2, 11 (1901), 559–572.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

[39] Nicola Pezzotti, Boudewijn PF Lelieveldt, Laurens Van Der Maaten, Thomas Hollt, Elmar Eisemann, and Anna Vilanova. 2016. Approximated and user steerable tSNE for progressive visual analytics. IEEE transactions on visualization and computer graphics 23, 7 (2016), 1739–1752.

[40] Jorge Luis Reyes-Ortiz, Alessandro Ghio, Xavier Parra, Davide Anguita, Joan Cabestany, and Andrea Catala. 2013. Human Activity and Motion Disorder Recognition: towards smarter Interactive Cognitive Environments,. In ESANN. Citeseer.

[41] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20 (1987), 53–65.

[42] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. science 290, 5500 (2000), 2323–2326.

[43] John W Sammon. 1969. A nonlinear mapping for data structure analysis. IEEE Transactions on computers 100, 5 (1969), 401–409.

[44] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. Neural computation 10, 5 (1998), 1299–1319.

[45] César R Souza. 2010. Kernel functions for machine learning applications. Creative Commons Attribution-Noncommercial-Share Alike 3 (2010), 29.

[46] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. science 290, 5500 (2000), 2319–2323.

[47] Florin Vaida. 2005. Parameter convergence for EM and MM algorithms. Statistica Sinica (2005), 831–840.

[48] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. The Journal of Machine Learning Research 15, 1 (2014), 3221–3245.

[49] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of machine learning research 9, 11 (2008).

[50] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. Statistics and computing 17 (2007), 395–416.

[51] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017).

[52] Bo Xie, Yang Mu, Dacheng Tao, and Kainen Huang. 2011. m-SNE: Multiview stochastic neighbor embedding. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41, 4 (2011), 1088–1096.

[53] Zhirong Yang, Irwin King, Zenglin Xu, and Erkki Oja. 2009. Heavy-tailed symmetric stochastic neighbor embedding. Advances in neural information processing systems 22 (2009), 2169–2177.

[54] Yin Zhang and Zhi-Hua Zhou. 2010. Multilabel dimensionality reduction via dependence maximization. ACM Transactions on Knowledge Discovery from Data (TKDD) 4, 3 (2010), 1–21.

ACM Trans. Knowl. Discov. Data.
A GRADIENT COMPUTATION WITH LOCAL VARIANCE

For Gaussian kernel with \( K_{ij} = \exp \left( -\frac{\|y_i - y_j\|^2}{2\sigma_i^2} \right) \). As the adjacent matrix \( Q \) is symmetric, \( q_{ij} = q_{ji} = \frac{K_{ij} + K_{ji}}{2} \), then the derivative with respect to \( Y \) becomes

\[
\frac{\partial \text{Tr}(V^T L V)}{\partial Y} = \sum_i \sum_j \frac{\partial \text{Tr}(V^T L V)}{\partial K_{ij}} \frac{\partial K_{ij}}{\partial Y} = \sum_i \sum_j \frac{1}{\sigma_i^2} (U_{ij}^0 + U_{ij}^1 + U_{ij}^2) K_{ij} \frac{\partial y_{ij}}{\partial Y} + \frac{1}{\sigma_j^2} (U_{ij}^0 + U_{ij}^1 + U_{ij}^2) K_{ij} \frac{\partial y_{ji}}{\partial Y} 
\]

(30)

In the case of the Gaussian kernel matrix, we can calculate the gradient on element \( y_a \) of \( Y \) as below.

\[
\frac{\partial \text{Tr}(V^T L V)}{\partial y_a} = \sum_i \sum_j \frac{\partial \text{Tr}(V^T L V)}{\partial K_{ij}} \frac{\partial K_{ij}}{\partial y_a} = \left( \frac{1}{\sigma_a^2} \sum_j \frac{\partial \text{Tr}(V^T L V)}{\partial K_{aj}} K_{aj} (y_a - y_j) \right) i = a \\
\left( \frac{1}{\sigma_j^2} \sum_i \frac{\partial \text{Tr}(V^T L V)}{\partial K_{ai}} K_{ai} (y_a - y_i) \right) j = a \\
\left( \text{let } U_{ij} = \frac{U_{ij}^0 + U_{ij}^1 + U_{ij}^2}{\sigma_i^2} \right) 
\]

(31)

\[
\frac{\partial \text{Tr}(V^T L V)}{\partial Y} = \begin{cases} 
U Y - Y \odot C \\
U^T Y - Y \odot C'
\end{cases}
\]

where, \( C = U_1 N_1^T \) \( C' = 1_d 1_N^T U \)

To sum up, the calculation of gradient \( \frac{\partial \text{Tr}(V^T L V)}{\partial Y} \) is

\[
\frac{\partial \text{Tr}(V^T L V)}{\partial Y} = (U + U^T) Y - Y \odot (C + C').
\]

(32)

ACM Trans. Knowl. Discov. Data.
B PROOF FOR PROPOSITION 1

Proof. Denote \( \hat{\mathcal{L}}_2(Y|Y_{t-1}) = \hat{\mathcal{L}}_2(Y) \). Using the definitions of \( \mathcal{L}_2(Y) \) and \( \hat{\mathcal{L}}_2(Y) \), we have

\[
\mathcal{L}_2(Y) \leq \hat{\mathcal{L}}_2(Y|Y_{t-1})
\]

and

\[
\mathcal{L}_1(Y) + \mathcal{L}_2(Y) \leq \mathcal{L}_1(Y) + \hat{\mathcal{L}}_2(Y|Y_{t-1})
\]

It follows that

\[
\mathcal{L}_1(Y_t) + \mathcal{L}_2(Y_t) \leq \mathcal{L}_1(Y_{t-1}) + \mathcal{L}_2(Y_{t-1}) - \Delta_t
\]

Supposing \( \mathcal{L}_1(Y_t) + \hat{\mathcal{L}}_2(Y_t) \leq \mathcal{L}_1(Y_{t-1}) + \hat{\mathcal{L}}_2(Y_{t-1}) \), and combining (35), we arrive at

\[
\mathcal{L}_1(Y_t) + \mathcal{L}_2(Y_t) \leq \mathcal{L}_1(Y_{t-1}) + \mathcal{L}_2(Y_{t-1}) - \Delta_t
\]

Invoking the fact \( \mathcal{L}_2(Y_{t-1}) = \hat{\mathcal{L}}_2(Y_{t-1}|Y_{t-1}) = \hat{\mathcal{L}}_2(Y_{t-1}) \) into (36), we obtain

\[
\mathcal{L}_1(Y_t) + \mathcal{L}_2(Y_t) \leq \mathcal{L}_1(Y_{t-1}) + \mathcal{L}_2(Y_{t-1}) - \Delta_t
\]

Because \( \nabla \hat{\mathcal{L}}(Y) \) is L-Lipschitz continuous, we have

\[
\hat{\mathcal{L}}(Y) \leq \hat{\mathcal{L}}(Y_{t-1}) + \nabla \hat{\mathcal{L}}(Y_{t-1}) \cdot (Y - Y_{t-1}) + \frac{L}{2} \| Y - Y_{t-1} \|_F^2.
\]

Invoking the update of \( Y \), i.e., \( Y_t = Y_{t-1} - \alpha \nabla \hat{\mathcal{L}}(Y_{t-1}) \), into (38), we get

\[
\hat{\mathcal{L}}(Y_t) \leq \hat{\mathcal{L}}(Y_{t-1}) - \left( \frac{1}{\alpha} - \frac{L}{2} \right) \| Y_t - Y_{t-1} \|_F^2.
\]

It means

\[
\Delta_t = \left( \frac{1}{\alpha} - \frac{L}{2} \right) \| Y_t - Y_{t-1} \|_F^2.
\]

Now combining (37) and (40), we arrive at

\[
\mathcal{L}(Y_t) \leq \mathcal{L}(Y_{t-1}) - \left( \frac{1}{\alpha} - \frac{L}{2} \right) \| Y_t - Y_{t-1} \|_F^2.
\]

Sum up (41) from \( t = 1 \) to \( t = T \), we have

\[
\mathcal{L}(Y_T) \leq \mathcal{L}(Y_0) - \left( \frac{1}{\alpha} - \frac{L}{2} \right) \sum_{t=1}^{T} \| Y_t - Y_{t-1} \|_F^2.
\]

This finished the proof. □

C SUPPLEMENTARY RESULTS

Figure 7 shows the qualitative comparison of LapTSNE (in the first row) with other methods. LapTSNE has comparable performance with UMAP on visual results, and it is significantly better than other methods.
Fig. 7. A comparison of LapSNE and other dimensionality reduction methods for all datasets.