Feature Learning for Nonlinear Dimensionality Reduction toward Maximal Extraction of Hidden Patterns

Takanori Fujiwara* Linköping University
Yun-Hsin Kuo† University of California, Davis
Anders Ynnerman* Linköping University
Kwan-Liu Ma† University of California, Davis

2023 IEEE 16th Pacific Visualization Symposium (PacificVis) | 979-8-3503-2124-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/PacificVis56936.2023.00021

ABSTRACT

Dimensionality reduction (DR) plays a vital role in the visual analysis of high-dimensional data. One main aim of DR is to reveal hidden patterns that lie on intrinsic low-dimensional manifolds. However, DR often overlooks important patterns when the manifolds are distorted or masked by certain influential data attributes. This paper presents a feature learning framework, *FEALM*, designed to generate a set of optimized data projections for nonlinear DR in order to capture important patterns in the hidden manifolds. These projections produce maximally different nearest-neighbor graphs so that resultant DR outcomes are significantly different. To achieve such a capability, we design an optimization algorithm as well as introduce a new graph dissimilarity measure, named *neighbor-shape dissimilarity*. Additionally, we develop interactive visualizations to assist comparison of obtained DR results and interpretation of each DR result. We demonstrate FEALM's effectiveness through experiments and case studies using synthetic and real-world datasets.

Keywords: Dimensionality reduction, feature learning, network comparison, Nelder-Mead optimization, UMAP, visual analytics.

1 INTRODUCTION

High-dimensional data can contain a rich set of observations measured from phenomena. Dimensionality reduction (DR) constitutes a tool for the understanding of the phenomena by visually revealing patterns in the data and facilitating human interpretation of the patterns [4, 10, 34], leading to important and fundamental insights. Among others, nonlinear DR, such as t-SNE [40] and UMAP [28], is especially helpful when the patterns are hidden in nonlinear structures (or manifolds) and infeasible to be found from conventional depictions of data (e.g., with scatterplot matrices, heatmaps, and parallel coordinates [26]).

However, the nonlinear DR process is sensitive to an attribute's influence on manifolds. While nonlinear DR is commonly applied to all available attributes, it may fail to capture patterns underlying manifolds that are apparent only in a particular subset of attributes [23]. A similar problem also happens when manifolds are entangled by the relationships among attributes. Although researchers have investigated the effect of attribute selection on linear DR results [36], there is a lack of studies dealing with nonlinear DR as well as the case where manifolds are entangled.

In this work, we complement nonlinear DR methods to enable them to extract various important patterns existing in the manifolds embedded in the attributes or combinations of attributes. We first demonstrate that nonlinear DR suffers from the aforementioned problems even when a trivial change in data, such as the inclusion of one additional attribute, is made. We then present a feature learning framework, *FEALM*, designed to discover latent features of data, with which nonlinear DR produces significantly different results from the one using all the available attributes as they are. These latent features can be constructed with a linear projection that is equivalent to a combination of data scaling and transformation, which are commonly used for data preprocessing [14]. FEALM's feature learning is performed through the maximization of the differences between data representations (e.g., nearest neighbor graphs) highly related to nonlinear DR results. Within this framework, we design an exemplifying method for UMAP. We develop an algorithm utilizing the Nelder-Mead optimization method (NMM) [13] to find latent features to produce maximally different nearest-neighbor graphs, which are intermediate products of UMAP, and consequently generate diverse UMAP results. To detect the difference of the graphs, we introduce a new graph dissimilarity measure, called *neighbor-shape dissimilarity* (or *NSD*). Using this method, analysts can find multiple relevant UMAP results that are difficult to find through manual preprocessing of data.

We further develop an interactive visual interface to allow analysts to flexibly seek more patterns and gain insights from them. The interface depicts the similarities of DR results generated during the optimization process to notify unexplored embeddings. Also, through brushing and linking, analysts can conveniently compare multiple DR results. To help review each DR result, our interface integrates an existing contrastive-learning-based interpretation method [10], which highlights characteristics of a group of instances through comparison with others.

We demonstrate the effectiveness of FEALM, the exemplifying method, and the visual interface through experiments using synthetic datasets and multiple case studies on real-world datasets. We also conduct a performance evaluation to assess the efficiency of NSD and the optimization algorithm. We provide a demonstration video of the interface, detailed evaluation results, and the source code of FEALM in the supplementary material [1].

In summary, we consider our primary contributions to be:

- a feature learning framework, FEALM, designed to extract a set of latent features for nonlinear DR, each of which produces a significantly different DR result;
- an exemplifying method for UMAP, where we introduce an NMM-based algorithm as well as a graph dissimilarity measure, NSD;
- a visual interface that assists exploration of DR results and interpretation of each DR result; and
- designed examples that illustrate nonlinear DR's sensitiveness to trivial disturbance to intrinsic manifolds.

2 RELATED WORK

Our work supplements existing DR methods by learning appropriate features from high-dimensional data. Our feature learning explores various linear subspaces of the original data to generate significantly different DR results. We provide the background and relevant works in DR methods and subspace exploration.

2.1 Dimensionality Reduction Methods

DR is widely used for visual exploration of high-dimensional data [30, 34]. Many visualization-purpose DR methods aim to reveal overall data distributions (e.g., variances with principal component analysis, or PCA) or patterns (e.g., clusters in t-SNE results [10, 40]) in a low-dimensional space. When only performing a linear projection [6], a DR method is categorized as linear DR. More precisely, it produces an embedding (or representation), $\mathbf{Y}$, from input data, $\mathbf{X}$, with $\mathbf{Y} = \mathbf{XP}$, where $\mathbf{Y} \in \mathbb{R}^{n \times m'}$, $\mathbf{X} \in \mathbb{R}^{n \times m}$, $\mathbf{P} \in \mathbb{R}^{m \times m'}$, and n, m, m' are the numbers of instances, attributes, and latent features, respectively. For example, PCA is a well-known linear DR method. While linear DR only captures the linear structure of $\mathbf{X}$, nonlinear DR can uncover the nonlinear structure [42]. For example, t-SNE [40] and UMAP [28] aim to preserve local neighborhoods of each instance, which is often difficult when relying only on a linear projection.

*e-mail: {takanori.fujiwara, anders.ynnerman}@liu.se.
†e-mail: {yskuo, klma}@ucdavis.edu.

2165-8773/23/$31.00 ©2023 IEEE
DOI 10.1109/PacificVis56936.2023.00021

Despite the frequent use of the aforementioned DR methods for visual analytics [34], they may fail to show important patterns when data contains noises or influential attributes to the overall data distribution. Several DR methods have been developed to address this limitation. For example, discriminant analysis [6, 17], such as linear discriminant analysis (LDA), utilizes class information to reduce noises that are irrelevant to class separation. Contrastive learning [2], such as contrastive PCA, compares two datasets to reveal patterns that are more salient in one dataset when compared to another. Unified linear comparative analysis [12] flexibly incorporates the strengths of both discriminant analysis and contrastive learning. However, all these methods require additional information (e.g., class labels) and focus only on revealing patterns related to a well-defined analysis interest (e.g., classification). Thus, these methods would not be suitable when performing an early-stage exploration without complete knowledge of data and/or expected findings, which is an important task that visual analytics should support.

Other than visualization, some of DR methods can be utilized for feature selection and feature learning [49]. For example, PCA and LDA are frequently used in data preprocessing for subsequent machine learning (ML) methods, such as deep neural networks or even other DR methods (e.g., t-SNE), as reducing dimensions is helpful to avoid high computational costs and the curse of dimensionality. As shown in a comprehensive survey by Zebari et al. [49], a large portion of this type of DR targets classification tasks.

FEALM can be seen as an unsupervised linear DR method for feature learning. We design FEALM to preprocess data to produce significantly different nonlinear DR results from one obtained using original data as is. FEALM does not require additional information, such as class labels; thus, it supports an early-stage exploration.

2.2 Exploration of Axis-Parallel Subspaces

There are two major types of subspaces: axis-parallel subspaces and linear subspaces. Axis-parallel subspaces are composed of a subset of original data attributes. Thus, there are 2^m subspaces we can explore. On the other hand, linear subspaces consist of axes obtained through linear projections of original data. Although linear subspaces embrace axis-parallel subspaces, here we only describe studies on axis-parallel subspaces and the rest in Sec. 2.3.

Scatterplot matrices and parallel coordinates are classic visualizations to explore axis-parallel subspaces [26]. A set of methods have been developed to improve these visualizations' scalability and usability [45,47]. In response to the increase in the available number of attributes, more efforts have been devoted to comparing a large set of subspaces. A common approach is finding meaningful subspaces with subspace selection, visualizing each subspace's dissimilarity, and informing patterns seen in each subspace with DR [19, 37, 43]. While this approach uses DR to understand subspaces, Sun et al. [36] investigated subspace selection's influences on PCA and multidimensional scaling (MDS) results. More comprehensive descriptions of relevant works can be found in a survey by Liu et al. [26].

Subspace clustering [23, 31] in the ML field shares a closely related concept with our work. Subspace clustering aims to find clusters within axis-parallel subspaces. By limiting the use of data to a subset of attributes, subspace clustering can uncover clusters that are masked by irrelevant attributes. Note that, in the visualization field, "subspace clustering" is often confusingly used to represent clustering *of* subspaces; however, in standard ML terminology, subspace clustering performs clustering *within* axis-parallel subspaces.

Our work seeks subspaces that produce significantly different nonlinear DR results to reveal hidden patterns. The work by Sun et al. [36] and subspace clustering methods [23, 31] are closely related to our work in terms of analyzing the subspace change's influence on data patterns. However, our work is for the use with nonlinear DR methods and provides optimization and visualization methods to find appropriate linear subspaces, which are not limited to axis-parallel subspaces.

2.3 Exploration of Linear Subspaces

To show high-dimensional data distributions in a selected 2D linear subspace, scatterplots and star coordinates are often used with interactive enhancements [26, 44]. We can see linear DR as a method that selects a view suitable to see some characteristics of data (e.g., variance with PCA) [15]. Various interactive adjustment methods for linear DR have been introduced, as summarized in surveys [30, 34].

When dealing with many attributes, manually finding informative subspaces becomes almost infeasible. Thus, researchers have designed (semi)automatic and visual recommendations. For example, Wang et al. [44] utilized LDA to suggest star coordinates that show clear cluster separations. Zhou et al. [50] visualized the similarities of original attributes as well as 1D subspaces to help analysts construct interesting subspaces. Gleicher et al. [15] utilized support-vector machines to suggest simple linear subspaces that satisfy specifications provided by analysts. Grassmannian Atlas [25] provides an overview of projection qualities (e.g., skewness) of all 2D subspaces sampled from the Grassmannian manifold. Lehmann and Theisel [24] developed an optimization method to provide a set of 2D subspaces that are significantly different from each other.

Lehmann and Theisel's work [24] is the most related work as they also aimed to mine various patterns in different subspaces. While theirs only finds and visualizes 2D subspaces(i.e., equivalent to linear DR onto 2D planes), ours searches multidimensional subspaces and uses them as nonlinear DR inputs to uncover patterns hidden in complex data.

3 Motivating Examples

Using UMAP [28] as a representative nonlinear DR method, we provide concrete cases where nonlinear DR misses important patterns. The datasets that we created, source code for the data generation, and comprehensive experiment results are made available in the supplementary materials [1]. As shown in Fig. 1-a, we first generated a dataset with three attributes, with which instances are placed around two different spherical surfaces. This dataset has approximately 200 blue (`Sphere 1`) and 100 orange (`Sphere 2`) instances. The inner sphere corresponding to `Sphere 2` has a radius with 40% length of the outer sphere's radius. Also, small noises following a normal distribution are added for the placement of each instance.

When visualizing this dataset in a 2D space, linear DR can only produce a plane cut of the 3D spheres; consequently, we cannot see a clear distinction between them (see Fig. 1-b). Any linear DR causes a similar issue when data has curved shapes, such as those in the Swiss-Rolls dataset [32]. On the other hand, nonlinear DR methods that aim to preserve each instance's local neighbors such as t-SNE and UMAP can separate `Spheres 1` and `2`, as shown in Fig. 1-c.

However, even nonlinear DR easily fails to find important patterns when some attributes affect manifolds that contain the corresponding patterns. This situation can happen even with subtle changes in a dataset. To illustrate this, as shown in Fig. 1-d, we generated a new dataset by adding one attribute that has three classes (`Classes A`–`C`) having clearly different values. We shuffled the order of the 3-class attribute's instances (i.e., there are no correspondence among `Spheres 1`–`2` and `Classes A`–`C`). Also, we applied the Z-score normalization for each attribute to follow the standard preprocessing for DR [14] and to avoid creating strong influences from the 3-class attribute. UMAP results for this dataset are presented in Fig. 1-e, f. We can see that UMAP does not show the separation of `Spheres 1` and `2` anymore. Moreover, UMAP does not clearly distinguish `Classes A`–`C` either. Similar issues happen even when using other nonlinear DR methods, such as t-SNE. Also, hyperparameter adjustments of UMAP, such as k used for the k-nearest neighbor (k-NN) graph construction, cannot solve this issue as the manifold itself is distorted (for details, refer to [1]).

The issue seen in this dataset (Fig. 1-d) can be solved by assigning larger weights to the 2-sphere attributes (or excluding the 3-class attribute), which leads to a clear separation of `Spheres 1`–`2`. Simi-

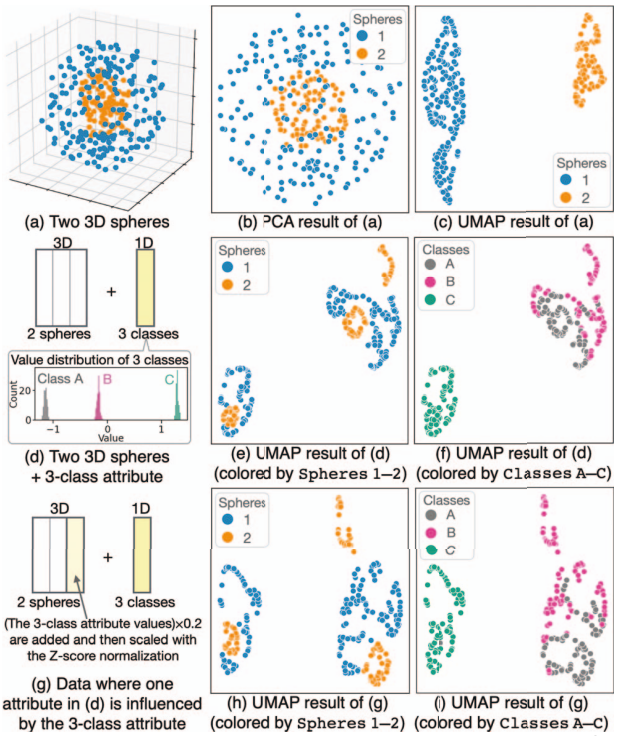

Figure 1: Three datasets (a,d,g) and corresponding DR results: PCA (b) and UMAP (c,e,f,h,i).

larly, by assigning a large weight only to the 3-class attribute, DR can find Classes A–C. In fact, our method for UMAP, which we introduce in Sec. 5, can reveal these two patterns by automatically adjusting the weights. Fig. 2 shows three examples suggested by our method. The results are produced with the same UMAP's hyperparameters as those used in Fig. 1-e. In Fig. 2-c1, by using a relatively small weight for the 3-class attribute (i.e., 0.2), UMAP shows clear clusters of Spheres 1–2. Similarly, Fig. 2-b2 shows three clusters of Classes A–C by assigning the 3-class attribute a large weight.

The above issue can be more complicated when patterns are entangled in multiple attributes' relationships. We create a dataset exhibiting such a case by adding 20% portions of the 3-class attribute values into one of the 2-sphere attributes and then applying the Z-score normalization again, as described in Fig. 1-g. This type of entanglements can be found in, for example, income statistics partially influenced by age and a political opinion influenced by a voter's general ideology. The attribute weighting or selection cannot resolve the issue in this dataset. Fig. 3-a1, a2 show UMAP results on this dataset after removing the 3-class attribute, which still do not show the separation of Spheres 1–2. This can be solved by learning latent features by our method. The right two columns of Fig. 3 show a subset of the generated UMAP results. We can see b1 and c2 clearly separate Spheres 1–2 and Classes A–C, respectively.

The above examples demonstrate that nonlinear DR can easily overlook important, obvious data patterns when certain attributes influence intrinsic manifolds—even a single attribute can cause this situation. While the examples are the case for finding distinct data groups, similar issues can happen even when finding, for example, continuous value changes on manifolds as in the Swiss-Rolls dataset.

4 FEALM Framework

We introduce a feature learning framework, *FEALM*, to address the stated issues in nonlinear DR. We name the framework FEALM because it performs *FEA*ture *L*earning to capture or *film* patterns underlying hidden *M*anifolds. As the problem can be overly complicated based on the combinations of DR methods, their hyperparameters, and hidden manifolds we should consider, we first specify our scope. We then describe the architecture of FEALM, where optimized projection matrices are generated with the following steps: (1) constructing a (graph) representation of data, (2) performing optimization to find a projection matrix with which a (graph) representation corresponding to projected data is maximally different from the one constructed in the first step, and (3) repeating the optimized projection matrix generation to produce a maximally different (graph) representation from those obtained so far.

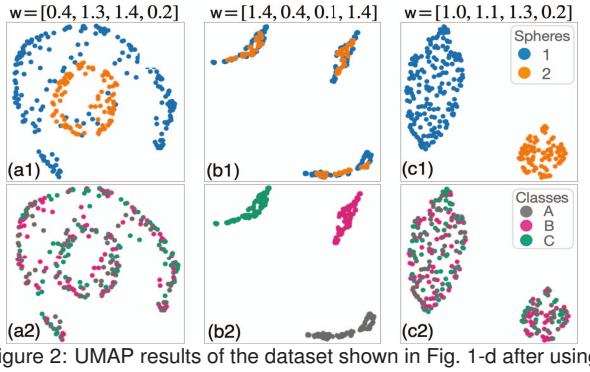

Figure 2: UMAP results of the dataset shown in Fig. 1-d after using our feature learning method. **w** shows the attribute weights.

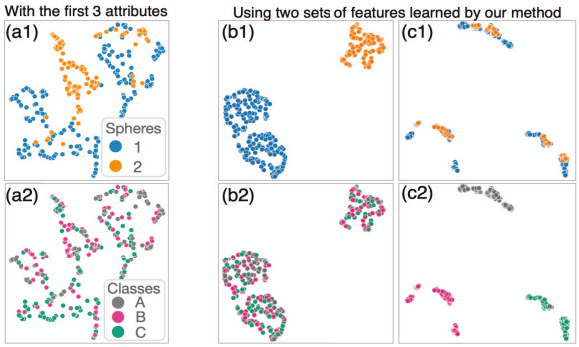

Figure 3: UMAP results of the dataset shown in Fig. 1-g after selecting only the first three attributes (a) and using our feature learning (b,c).

4.1 Problem Scope

FEALM aims to supplement ***nonlinear DR*** methods, even more specifically, for those construct a ***graph-based*** data representation as their intermediate product or those generate DR results highly related to a graph-based data representation. This scope is reasonable and still provides enough flexibility in FEALM because many DR methods can be considered graph-based [28, 46]. Such methods include MDS, the Barnes-Hut t-SNE (common t-SNE implementation) [40], and UMAP [28]. For example, MDS constructs a dissimilarity matrix of instances, which can be converted to a kernel/similarity matrix—corresponding to a weighted graph where nodes and edges represent instances and their similarities, respectively. The Barnes-Hut t-SNE and UMAP perform DR based on a similarity/weighted graph derived from the k-NN graph of instances.

FEALM searches significantly different DR results for a ***given DR method*** with ***given hyperparameters***. FEALM is not designed to select a DR method nor hyperparameters to reveal hidden patterns.

FEALM can only find patterns underlying (nonlinear) manifolds that exist in a ***linear subspace*** of the original data. We set this scope because of two reasons. First, we do not want to allow unintuitive or excessive data manipulation not only to provide more interpretable data preprocessing but also to avoid leading to false patterns as much as possible. A linear projection only allows a set of linear transformations that can be converted into a single matrix multiplication (i.e., $\mathbf{Y} = \mathbf{XP}$, as described in Sec. 2.1). Based on analysts' demands,

FEALM's linear projection can be limited to data scaling (or attribute weighting), orthogonal transformation, and a combination of both, all of which are commonly used for data preprocessing. Second, it is computationally challenging to handle manifolds that cannot be uncovered even by applying linear projections to the data. Such manifolds might be able to be found, for example, by utilizing neural networks; however, it requires expensive parameter tuning.

4.2 Optimization Architecture

We describe the architecture of FEALM designed with considerations of flexibility and computational efficiency.

Forms of the problem. For input data, $\mathbf{X} \in \mathbb{R}^{n \times m}$ (n, m: the numbers of instances and attributes), latent features can be computed with $\mathbf{XP}_i$, where $\mathbf{P}_i \in \mathbb{R}^{m \times m'}$ (m' is the number of latent features and $m' \leq m$) is a projection matrix. With f_{DR}, a function that performs DR with a given method and hyperparameters, we can obtain a DR result or representation, $\mathbf{Y}_i$, i.e., $\mathbf{Y}_i = f_{\mathrm{DR}}(\mathbf{XP}_i)$. We can measure a dissimilarity of two DR results, $\mathbf{Y}_i$ and $\mathbf{Y}_j$, with a certain function, $d_{\mathrm{DR}}(\mathbf{Y}_i, \mathbf{Y}_j)$ (e.g., the Frobenius norm). This expression shows that the "difference" in DR results can be varied by d_{DR}, and it should be selected based on general analytical interests. Let $\mathbf{Y}_0$ denote a DR result using $\mathbf{X}$ as is (i.e., $\mathbf{Y}_0 = f_{\mathrm{DR}}(\mathbf{X})$). Then, we can write an optimization problem to find $\mathbf{P}_1$ that generates $\mathbf{Y}_1$ maximally different from $\mathbf{Y}_0$ as: $\mathrm{argmax}_{\mathbf{P}_1} d_{\mathrm{DR}}(\mathbf{Y}_1, \mathbf{Y}_0)$. When iteratively finding a new projection matrix, $\mathbf{P}_{i+1}$, we want to find one that generates $\mathbf{Y}_{i+1}$ maximally different from a set of DR results already produced, $\mathcal{Y}_i = \{\mathbf{Y}_0, \cdots, \mathbf{Y}_i\}$. With some reduce function, Φ (e.g., mean), the problem of finding $\mathbf{P}_{i+1}$ can be written as:

$$\underset{\mathbf{P}_{i+1}}{\mathrm{argmax}}\, \Phi\left(d_{\mathrm{DR}}\left(\mathbf{Y}_{i+1}, \mathbf{Y}_0\right), \cdots, d_{\mathrm{DR}}\left(\mathbf{Y}_{i+1}, \mathbf{Y}_i\right)\right). \tag{1}$$

Note that Eq. 1 shares some similarities with the one presented by Lehmann and Theisel [24]. However, their optimization is only to produce 2D linear DR results, and their case limits to $\mathbf{Y}_i = \mathbf{XP}_i$ ($\mathbf{P}_i \in \mathbb{R}^{m \times 2}$). Also, they only use the extended version of the Procrustes distance [16] and Frobenius norm as a combination of d_{DR} and Φ (for more details, refer to [24]). FEALM can be used for nonlinear DR and provides flexibility for each function choice. For Φ, rather than computing a norm or maximum, we recommend taking a minimum of dissimilarities (i.e., Eq. 1 maximizes the minimum of dissimilarities). With this, we can find a DR result that is different from all the existing results and avoid a case where the optimization keeps producing the same or similar results (e.g., a case where $\mathbf{Y}_0$ has an extremely larger dissimilarity with $\mathbf{Y}_1$ than with other potential DR results). We provide a graphical explanation of such a case in the supplementary materials [1].

While Eq. 1 is a straightforward description of our goal, directly performing this optimization for nonlinear DR methods is often difficult due to two-folded reasons. First, f_{DR} is often computationally expensive. For example, UMAP took 5 seconds to produce Fig. 1-h from the data containing only 300 instances and 4 attributes. If the optimization requires many evaluations/trials, completion time can easily surpass several hours (e.g., about 1.5 hours for 1000 evaluations). Second, popularly used nonlinear DR methods such as t-SNE and UMAP contain randomness in f_{DR}. For example, random initialization of a representation (as in t-SNE) or random sampling during the optimization (as in UMAP) highly influences the final result [22]. Consequently, it becomes difficult to adjust $\mathbf{P}_i$ during the optimization—when the objective value of Eq. 1 becomes better, we do not know whether it is the improvement from changes in $\mathbf{P}_i$ or caused by the randomness in f_{DR}.

Thus, FEALM also introduces an optimization problem that maximizes the differences among graph representations of data. We can use graph representations that are the same as or similar to intermediate products of given DR. This optimization is based on a general observation: if such graph representations have maximal differences, derived DR results are also significantly different. Let f_{Gr} denote a function that generates some graph (e.g., k-NN graph, similarity matrix) from an input matrix. Then, a graph, G_i, corresponding to $\mathbf{P}_i$ can be obtained with $G_i = f_{\mathrm{Gr}}(\mathbf{XP}_i)$. Also, we denote a function that measures a dissimilarity of two graphs, G_i and G_j, by $d_{\mathrm{Gr}}(G_i, G_j)$. With a set of already produced graphs, $\mathcal{G}_i = \{G_0, \cdots, G_i\}$, we can write a relaxed version of the optimization problem:

$$\underset{\mathbf{P}_{i+1}}{\mathrm{argmax}}\, \Phi(d_{\mathrm{Gr}}(G_{i+1}, G_0), \cdots, d_{\mathrm{Gr}}(G_{i+1}, G_i)). \tag{2}$$

When developing a method within FEALM, we can choose Eq. 1 or 2 based on the characteristics of a DR method, such as the computational efficiency and stability. We can also use Eq. 1 and 2 in a hybrid manner. For example, we can generate a large number of projection matrices with Eq. 2 and then filter them with Eq. 1 to obtain refined results.

Constraints on a linear projection. Another important consideration of the optimization is the constraints on $\mathbf{P}_i$. We should decide the constraints based on data manipulation allowed for an analysis goal and the optimization difficulty for a given dataset (Sec. 6 provides the detailed discussions). Here we list representative options: (1) no constraint; (2) allowing only data scaling; (3) allowing data scaling and orthogonal transformation. With any option, we can interpret how data is transformed by reviewing values in $\mathbf{P}_i$.

When there is (1) no constraint in a projection matrix, $\mathbf{P}$, FEALM most flexibly learns features. However, as orthogonality between each learned feature is not guaranteed, distance-related functions (e.g., k-NN graph construction using the Euclidean distance) might be heavily influenced by the distortion. Also, the optimization needs to search the best values for $m \times m'$ parameters in $\mathbf{P}$.

When (2) allowing only data scaling, $\mathbf{P} = \mathrm{diag}(\mathbf{w})$ (i.e., with $\mathbf{XP}$, each column of $\mathbf{X}$ is multiplied by the corresponding weight in $\mathbf{w}$) where $\mathbf{w}$ is an m-dimensional vector. Practically, we can restrict $\mathbf{w} = \sqrt{m}\mathbf{u}$ where $\mathbf{u}$ is a unit vector. When $\mathbf{u}$ consists of uniform values, $\mathbf{w} = (1 \cdots 1)^\top$ (i.e., no scaling). Then, $\mathbf{w}$ can be identified by searching a unit vector. This constraint is used when generating the results in Fig. 2. As this search is only on m parameters, finding the best $\mathbf{P}$ is much easier than the case with no constraint.

The last constraint (3) can be written as $\mathbf{P} = \mathrm{diag}(\mathbf{w})\mathbf{M}\,\mathrm{diag}(\mathbf{v})$, where $\mathbf{M} \in \mathbb{R}^{m \times m'}$ is an orthogonal matrix (i.e., $\mathbf{M}^\top \mathbf{M} = I_{m'}$; $I_{m'}$ is an $m' \times m'$ identity matrix) and $\mathbf{v}$ is an m'-dimensional vector. Here $\mathbf{M}$ ensures that $\mathbf{M}\,\mathrm{diag}(\mathbf{v})$ generates orthogonal features of $\mathbf{X}\,\mathrm{diag}(\mathbf{w})$. And, $\mathbf{v}$ weights the features to control their influence on a projection. Similar to $\mathbf{w}$, we can decompose $\mathbf{v}$ with $\mathbf{v} = \sqrt{m'}\mathbf{u}'$, where $\mathbf{u}'$ is a unit vector. A projection under this constraint resembles a combination of standard preprocessing steps (i.e., data scaling and orthogonal data transformation). This constraint still needs to find the best values for $m \times m'$ parameters.

Regularization. To control how strongly $\mathbf{P}$ can be of uniform or nonuniform values, we can *optionally* apply regularization by adding a penalty term into Eq. 1 or Eq. 2. When applying data scaling (i.e., $\mathbf{P} = \mathrm{diag}(\mathbf{w})$), we can add an L1-norm-based penalty: $-\lambda_1 \|\mathbf{w}\|_1$ ($\lambda_1 \in \mathbb{R}$). As λ_1 becomes a larger *positive* value, the optimization tends to produce $\mathbf{w}$ with *nonuniform* values. On the other hand, by using large *negative* λ_1, $\mathbf{w}$ can consist of more *uniform* values (e.g., when $\lambda_1 = -\infty$, $\mathbf{w}$ becomes $(1 \cdots 1)^\top$). Using negative λ_1 is especially effective when we want to avoid generating dissimilar graphs, $\mathcal{G}_i$ (or dissimilar DR results, $\mathcal{Y}_i$), that can be derived from a selection of few attributes (e.g., when analyzing binary or ordinal data). Note that the L2 norm of $\mathbf{w}$ is always constant (i.e., $\|\mathbf{w}\|_2 = \sqrt{m}$) and is not suitable for this regularization.

For the other cases (e.g., $\mathbf{P} = \mathrm{diag}(\mathbf{w})\mathbf{M}\,\mathrm{diag}(\mathbf{v})$), similar to the above, we can control the sparsity of $\mathbf{P}$ with the L1-norm-based penalty: $-\lambda_1 \mathbf{1}_{\mathrm{m}}^\top |\mathbf{P}| \mathbf{1}_{\mathrm{m}'}$ where $\mathbf{1}_m = (1 \cdots 1)^\top \in \mathbb{R}^m$, $\mathbf{1}_{m'} = (1 \cdots 1)^\top \in \mathbb{R}^{m'}$, and $\mathbf{1}_{\mathrm{m}}^\top |\mathbf{P}| \mathbf{1}_{\mathrm{m}'}$ is the sum of all elements of $|\mathbf{P}|$. To further regulate the difference of each column in $\mathbf{P}$, we can add a penalty based on the sum of each $\mathbf{P}$ row's L2 norm: $-\lambda_2 \mathbf{1}_m^\top ((\mathbf{P} \circ \mathbf{P}) \mathbf{1}_{m'})^{1/2}$ where $\lambda_2 \in \mathbb{R}$ and $\circ$ is the Hadamard product. With large negative λ_2, FEALM generates $\mathbf{P}$ in which each attribute has diverse weights across columns, and vice versa. Note

that the sum of each $\mathbf{P}$ row's L1 norm is equal to the sum of each $\mathbf{P}$ column's L1 norm; thus, the L2 norm should be used to control the differences in the columns.

General optimization strategies. Eq. 1 and 2 can be considered as the optimization over manifolds (or often called manifold optimization) [6, 38]. For example, for the no-constraint option (1), $\mathbf{P}$ can be found from the Euclidean manifold. A solution under the constraint (2) is derived by finding $\mathbf{u}$ from a unit sphere manifold. Lastly, for the constraint (3), we can find $\mathbf{u}$, $\mathbf{u}'$ from unit sphere manifolds and $\mathbf{M}$ from the Grassmann manifold [38] that is a manifold of m'-dimensional subspaces of m-dimensional space. To perform manifold optimization, we can utilize existing libraries, such as Pymanopt [38]. These libraries can help us, for example, generate parameters on a specified manifold.

To solve the optimization, when all functions involved in Eq. 1 (or Eq. 2) are differentiable, we can utilize automatic differentiation together with a solver for differentiable functions (e.g., gradient descent) through existing libraries [38]. When some functions are not differentiable (e.g., k-NN graph construction), we can use a derivative-free solver, such as the NMM.

In addition to the problem, constraints, and solver, we need to select or design f_{Gr}, d_{Gr}, and/or d_{DR} based on a DR method.

5 Exemplifying Method

We design an exemplifying method for UMAP, using FEALM. In the rest of the paper, we denote this method *FEALM-UMAP*. We chose UMAP because it is computationally rather efficient (e.g., when compared with t-SNE) [28] and frequently used for visualization in various applications [7, 11, 22]. The specific designs for UMAP can also be generalized and easily adapted to other DR methods. For example, we expect that a method for t-SNE can be developed based on FEALM-UMAP with minor adjustments in f_{Gr}.

5.1 Graph Generation Function

UMAP processes data in two steps: graph construction and graph layout. Through iterative optimization, the graph layout process performs the placement of instances (often in 2D) based on a constructed graph. This iterative optimization involves random sampling and expensive computations. Thus, we design a method using Eq. 2, which requires f_{Gr} and d_{Gr}. During the graph construction process, UMAP computes the instance dissimilarities (by default, using the Euclidean distance) and then produces a k-NN graph based on the dissimilarities. Afterward, UMAP constructs a fuzzy graph, which is a weighted graph where the dissimilarities are converted to the fuzzy topological representation (refer to [28] for details). From this fact, f_{Gr} can be the generation of a k-NN or fuzzy graph. While a fuzzy graph contains richer information of instances' relationships, many of the state-of-the-art graph dissimilarity measures, including those we utilize to design our measure, are only available for unweighted graphs [27]. Therefore, we use f_{Gr} that generates a ***k-NN graph***; however, we can replace this with a fuzzy graph once d_{Gr} suitable for weighted graphs is developed.

5.2 Graph Dissimilarity Measure

We can select a dissimilarity measure for unweighted graphs based on analysis interest. When using nonlinear DR for visualization, however, we usually want to reveal patterns that are visually apparent and related to the instances' neighborhood relationships (i.e., *shape* and *neighbors*), such as clusters and outliers [30]. Another critical consideration is computational efficiency as graph comparison itself is often expensive. But, it is difficult to judge only based on theoretical time complexity because of their detailed implementation differences (e.g., requiring only fast matrix operations or slow iterative loops). Based on our experiments (see Sec. C.2 in our supplementary materials [1]), we identify that NetLSD [39] can better capture differences of graph shapes with a greatly shorter runtime than many other measures available in a library of graph dissimilarities [27]. With the eigenvalue-based approximation [39], NetLSD's time complexity is $\mathcal{O}(qkn + q^2 n)$, where q is the total number of the top and bottom eigenvalues to take for the approximation and k is the number of neighbors set to generate an k-NN graph. NetLSD does not consider the neighbor dissimilarity (i.e., the difference of neighborhood relationships); thus, for d_{Gr}, we introduce a new measure, ***NSD***, to capture both neighbor and shape dissimilarities.

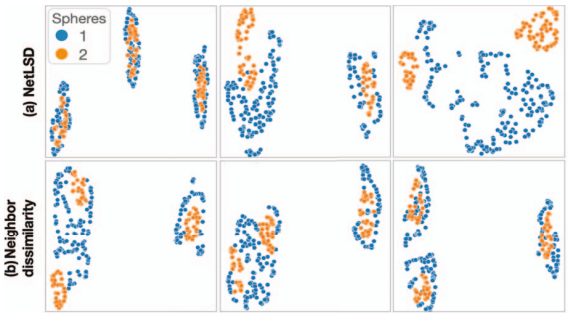

Figure 4: UMAP results of the dataset shown in Fig. 1-d, using the same settings with Fig. 2 except for d_{Gr}. While Fig. 2 is generated with NSD, here we use NetLSD (a) and the neighbor dissimilarity (b).

5.2.1 Neighbor-Shape Dissimilarity (NSD)

We design a neighbor dissimilarity measure, *ND*, and combine it with NetLSD to introduce *NSD*. Fig. 2 and Fig. 4 demonstrate how these measures affect DR results when applying FEALM-UMAP to the same dataset. For example, as seen in Fig. 4-b, ND only considers the changes of k-neighbors around each instance; as a result, they tend to form similar shapes, where the orders of adjacent nodes are likely different. On the other hand, the results with NetLSD (Fig. 4-a) show different shapes; however, they might not involve many neighborhood changes. Although we need more investigations to precisely conclude these tendencies, the algorithms used for NetLSD and ND only consider the shape and neighbor differences, respectively. When using NSD (Fig. 2), we can find patterns related to both types of changes.

For the neighbor dissimilarity, one option is to utilize the steadiness and cohesiveness (SnC) [20], which are developed to assess the DR quality by measuring the changes of the neighborhood relationships in the original data and a DR result. While SnC is adaptable to graph comparison, it involves random-walk-based sampling and expensive clustering steps. Thus, similar to the reasons why using Eq. 2 instead of Eq. 1, SnC is not suitable for use in the optimization. Inspired by SnC, we design ND based on shared-nearest neighbor (SNN) similarity [9].

SNN similarity measures how much of neighbors are shared in each pair of instances in a graph. Let $\mathbf{A}$ denote a directed adjacency matrix containing the information of each instance's k-NNs. Then, all instance pairs' SNN similarities, $\mathbf{S}$, can be computed with $\mathbf{S} = \mathbf{A}\mathbf{A}^\top / k$. Given two graphs, G_i and G_j, we can obtain the difference of each instance's SNN similarity with $\mathbf{D}_{i,j} = \mathbf{S}_i - \mathbf{S}_j$. Let $\mathbf{D}^+_{i,j}$ and $\mathbf{D}^-_{i,j}$ denote matrices only taking positive and negative values of $\mathbf{D}_{i,j}$, respectively. Then, $\mathbf{D}^+_{i,j}$ and $\mathbf{D}^-_{i,j}$ capture the increase and decrease of SNNs for each instance in G_i when compared to G_j. We can compute the total increase and decrease with the Frobenius norm, i.e., $\|\mathbf{D}^+_{i,j}\|_F$ and $\|\mathbf{D}^-_{i,j}\|_F$. Lastly, we reduce them to one value by taking the maximum. That is, ND of G_i and G_j is defined as:

$$d_{\mathrm{ND}}(G_i, G_j) = \max(\|\mathbf{D}^+_{i,j}\|_F, \|\mathbf{D}^-_{i,j}\|_F). \tag{3}$$

Unlike SnC, ND involves only simple matrix computations while keeping a similar strength to SnC to capture the neighbor dissimilarity. We compare SnC and ND in Sec. 6.

Let d_{SD} ($d_{\mathrm{SD}} \geq 0$) denote the shape dissimilarity measure using NetLSD. Since NetLSD is only for undirected graphs, we use undirected k-NN graphs as NetLSD's inputs (i.e., $\mathbf{A} + \mathbf{A}^\top - \mathbf{A} \circ \mathbf{A}^\top$ instead of $\mathbf{A}$). Also, by default, we set $q = 50$ for the approximation.

Then, we define the dissimilarity measured by NSD as:

$$d_{\mathrm{NSD}}(G_i, G_j) = d_{\mathrm{ND}}(G_i, G_j)^{\beta} \cdot \log\left(1 + d_{\mathrm{SD}}(G_i, G_j)\right) \quad (4)$$

where $d_{\mathrm{NSD}}(G_i, G_j) \geq 0$ and β ($0 \leq \beta \leq \infty$) is a hyperparameter that controls how strongly NSD focuses on the neighbor dissimilarity vs. the shape dissimilarity. When $\beta = 0$, NSD is equivalent to using NetLSD. As β increases, ND becomes more influential on NSD. Based on our experiment, we set $\beta = 1$ by default. β can be adjusted based on the patterns we look for. Since NetLSD involves an exponential function when computing the dissimilarity (refer to [39]), we take a logarithm of $1 + d_{\mathrm{SD}}$ (1 is added to avoid taking a logarithm of 0) to avoid excessive influence from the shape difference.

5.3 Optimization

FEALM-UMAP optimizes Eq. 2 while using the k-NN graph construction as f_{Gr} and d_{NSD} as d_{Gr}. As recommended, we use Φ to take a minimum of the dissimilarities. For the constraints of a linear projection, FEALM-UMAP supports all the three representative options described in Sec. 4.2. Since the k-NN graph construction is a non-differentiable function, we develop an ***NMM-based derivative-free solver***. We provide pseudocode in the supplementary material [1].

The optimization by the ordinary NMM begins with initial $(p+1)$ solutions in a p-dimensional space, where p is the number of parameters (i.e., when only allowing data scaling, $p = m$; for the other cases, $p = m \times m'$). The initial solutions are typically generated at random. Then, based on the evaluation of each solution's objective value, the NMM iteratively moves each solution toward a direction along which a better solution can be likely found while gradually shrinking a searching space. Then, after the user-indicated number of evaluations or the convergence, the NMM returns the best solution so far as a final result. When compared with other derivative-free solvers such as the particle-swarm optimization [21], the NMM does not involve many evaluations of the objective function, and efficiently finds a reasonable solution [48]. By employing the NMM, we can avoid an excessive number of graph dissimilarity calculations that are part of the objective function.

However, based on the initial solutions, the ordinary NMM easily falls into the local minimum. To mitigate this issue, similar to other hybrid approaches of global and local optimization solvers [48], we incorporate random search optimization into the NMM. Specifically, instead of $(p+1)$ initial solutions, our solver generates a large number of random solutions (by default, $(10p+1)$ solutions). Then, the solver selects the $(p+1)$ best solutions and applies the NMM to them to find the refined solution. For this refining step, to achieve faster convergence than the ordinary NMM for a case with large p (e.g., $p > 5$), we employ the adaptive NMM introduced by Gao and Han [13]. Even with this adaptive method, the NMM is usually suitable for the optimization involving a considerably small number of parameters (e.g., $p < 30$). Thus, when $\mathbf{X}$ has extremely large m (e.g., $m = 100$), we recommend preprocessing $\mathbf{X}$ with, for example, PCA or clustering to generate compressed attributes. This type of approaches is often recommended for a complex optimization (e.g., t-SNE [41] often employs PCA when $m > 30$).

Random initialization of solutions and restriction of their movement on a specified manifold (e.g., the Grassmann manifold) can be easily achieved by utilizing the manifold optimization libraries [38]. By repeating the above optimization, for example, till the evaluation result of Eq. 2 converges (refer to [1]), we can obtain a set of projection matrices, $\mathcal{P} = \{\mathbf{P}_0, \cdots, \mathbf{P}_r\}$ where r is the number repeats. When r is large (e.g., $r = 100$), we can perform spectral clustering [29] on $\mathcal{P}$ to recommend a small number of projections (e.g., 10 projections) that produce significantly different DR results.

5.4 Implementation Details and Complexity Analysis

Implementation. FEALM and FEALM-UMAP are implemented with Python and libraries for matrix computations and optimizations: NumPy/SciPy, Scikit-learn [32], and Pymanopt [38]. While many graph dissimilarities, including NetLSD, are available in netrd [27], we use our implementation, which fully utilizes matrix computations to achieve faster calculation (e.g., our implementation of NetLSD is approximately 20 times faster [1]). Moreover, Pathos is utilized to use multiprocessing for the NMM-based solver.

Time complexity analysis. The k-NN graph construction used for f_{Gr} has $\mathcal{O}(n\log(n)m)$ when using a ball-tree method [32]. NSD is composed of NetLSD ($\mathcal{O}(qkn + q^2n)$) and ND ($\mathcal{O}(kn^2)$), where q is the number of eigenvalues (see Sec. 5.2); because $q \leq n$, NSD has $\mathcal{O}(q^2n + kn^2)$. Thus, when computing the best solution with the NMM, the cost calculation for each solution takes $\mathcal{O}(rn(q^2 + kn))$, where r is the number of produced graphs so far.

6 Computational Evaluations

We evaluate the performance of computations related to FEALM-UMAP as well as the design of ND by comparing it with SnC [20]. As an experimental platform, we used the MacBook Pro (16-inch, 2019) with 2.3 GHz 8-Core Intel Core i9 and 64 GB 2,667 MHz DDR4. We prepared datasets with the data generation code provided in [12]. From the 20 Newsgroups dataset [8], their code can generate data with various numbers of instances (documents) and attributes (topics) by utilizing the latent Dirichlet allocation. All source code used for the evaluations is available online [1].

Comparison of ND and SnC. We have introduced ND as a faster, more stable alternative to SnC. Here we validate that ND and SnC similarly capture the neighbor changes. Analogous to ND's $\|\mathbf{D}^{+}_{i,j}\|_F$ and $\|\mathbf{D}^{-}_{i,j}\|_F$, SnC produces two distinct values, the steadiness and cohesiveness. We define the SnC-based dissimilarity measure as $d_{\mathrm{SnC}} = 1 - \min(\mathit{steadiness}, \mathit{cohesiveness})$. Note that steadiness and cohesiveness take a range of 0–1; the larger, the fewer changes. For this experiment, we set n=200, m=10, and k=15 and randomly generated 500 different projection matrices with the size of 10×5 and graphs corresponding to the projection matrices (with $f_{\mathrm{Gr}}(\mathbf{XP})$).

Fig. 5-a shows d_{ND} and d_{SnC} of a graph corresponding to the original data and each of the 500 generated graphs. As SnC contains the randomness and d_{SnC} can be inconsistent, we took the mean of 50 executions in Fig. 5-a. The mean d_{SnC} of 50×500 results was 0.20 and the mean of 500 standard deviations was 0.02 (i.e., 10% of the mean). Fig. 5-a presents strong correlations between d_{ND} and d_{SnC} with Pearson's and Spearman's correlation coefficients of 0.73 and 0.72, respectively. Thus, similar to SnC, ND captures the neighbor changes, while ND has no randomness and a significantly smaller computational cost, as discussed in the performance evaluation below. ND's strengths enable us to provide computationally efficient, stable NSD.

Performance of f_{DR}, f_{Gr}, d_{Gr}, and the optimization. We evaluate the efficiency of functions related to Eq. 1 and 2, specifically, UMAP (f_{DR}), k-NN graph construction (f_{Gr}), ND (d_{ND}), NetLSD (d_{SD}), NSD (d_{NSD}), and SnC. The number of instances, n, dominates these functions' complexities. Thus, we ran the functions with different n (n=50, $\cdots$, 3200) but fixed k, m, and q: k=15 (UMAP's default), m=10, and q=50 (NSD's default). As we expect many executions for the NMM, we measured the completion time of 1000 executions.

As expected, UMAP and SnC spent much longer completion time than others: e.g., 1616 and 527 seconds, respectively, for 1000 executions when n=50. Therefore, these functions are not suitable to be used in optimizations that require many evaluations or deal with a larger n—this is the reason why we have designed Eq. 2 and ND. Other functions' completion times are shown in Fig. 5-b. We observe that, as n increases, d_{ND} requires more computations, and dominates the completion time of d_{NSD}. However, 1000 executions of d_{NSD} still can be completed within 520 seconds when n=3200.

We next evaluate the performance of FEALM-UMAP as a whole. In addition to the same settings above, we set m'=2, no constraint on a projection matrix, and 1000 as the number of objective function evaluations. We then generated a single UMAP result through the

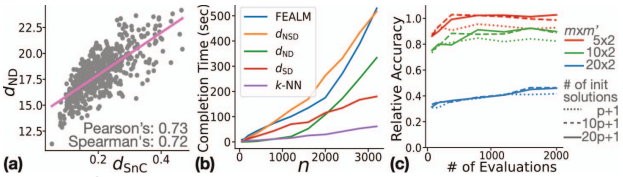

Figure 5: Computational evaluation results: (a) relationships between SnC-based dissimilarity and ND; (b) completion time for 1000 executions; (c) relative accuracy of solutions.

optimization (i.e., r=1). From the result shown in Fig. 5-b, we can see the completion time of FEALM-UMAP generally follows d_{NSD}. Even though FEALM-UMAP involves more computations (e.g., k-NN graph construction and the solution update for the NMM), FEALM-UMAP tends to show faster completion than d_{NSD}. This is probably because our implementation of the NMM partially employs parallel computations, as described in Sec. 5.4.

Quality of the optimization. Similar to the above experiment, we generated a single UMAP result with no constraint. For this experiment, we used the same number of instances (n=800) but a different number of attributes (m=5, 10, 20). We set m'=2. Also, to see the effect of the NMM's settings on the optimization quality, we tested different numbers of evaluations (from 100 to 2000) and initial random solutions (p+1, 10p+1, 20p+1). As we do not know the truly best solution, we analyzed relative accuracy to the optimized solution with large numbers of initial solutions and evaluations, specifically, (50p+1) solutions and 5000 evaluations. We also set a randomly selected solution as the baseline solution. Let v, v_{best}, and v_{base} denote objective values of Eq. 2 for the comparing, best, and baseline solutions, respectively; then, the relative accuracy is $(v - v_{\mathrm{base}})/(v_{\mathrm{best}} - v_{\mathrm{base}})$. As the NMM's solutions can be varied based on the initialization, we computed each of v, v_{best}, and v_{base} by taking a mean of 10 trials.

Fig. 5-c shows the relative accuracy. Generally, the increase in the number of evaluations improves the accuracies. Also, our hybrid approach using many initial random solutions improves the accuracies (e.g., (10p+1) reaches better accuracies than (p+1)). Also, we observe that the relative accuracy tends to be higher for the smaller searching space (e.g., the red lines have higher accuracies). For the 20×2 searching space, the accuracy relative to v_{best} is still low even with 2000 executions and (20p+1) initial solutions. Thus, we should use even larger numbers of evaluations and initial solutions to obtain better results. However, as discussed, the NMM is more suitable for a small search space; thus, PCA or clustering of attributes can be applied for data preprocessing when the original search space is too large. An analysis example applying PCA can be found in the supplementary materials [1]. For a small search space, our default parameter, (10p+1) initial solutions, with 1000 evaluations would provide reasonable results.

7 Visual Interface

To efficiently investigate FEALM-UMAP's results, as shown in Fig. 6, we develop a visual user interface (UI), which is also applicable to other methods developed within FEALM. The UI is developed as a web application with Python, JavaScript, and D3. We provide a supplementary demonstration video of the UI [1].

Exploration of DR results. The views in Fig. 6-a and b are designed for exploration and comparison of the DR results. Fig. 6-a visualizes the information obtained through the optimization described in Sec. 5.3, including the set of UMAP results (i.e., $\mathcal{Y}$), dissimilarities of each result (i.e., $d_{\mathrm{DR}}(\mathbf{Y}_i, \mathbf{Y}_j)$), and clustering-based recommendations. To visually convey the dissimilarities of UMAP results, we generate a 2D plot by applying UMAP based on $d_{\mathrm{DR}}(\mathbf{Y}_i, \mathbf{Y}_j)$ (i.e., UMAP on the UMAP results). In this plot, each *square* point corresponds to a single UMAP result and their spatial proximities represent the similarities of the UMAP results. We indicate each point's belonging cluster by coloring each point and the isocontour generated by Bubble Sets [5]. We use black color to distinguish the selected points, which initially correspond to the recommended UMAP results. Also, a point of the original DR (i.e., $\mathbf{Y}_0$) is annotated with the cross mark and text, "OG". We also support fundamental interactions, such as zooming and tooltiping for previewing UMAP results (e.g., one in Fig. 6-a). A scrollable view in Fig. 6-b shows the UMAP results corresponding to $\mathbf{Y}_0$ and the selected black points. Their belonging clusters are indicated with texts and colored boxes (e.g., 7 with the green box). A *circle* point in each UMAP result represents a data instance. Also, we color these points based on their group labels with a different color scheme from Fig. 6-a. For the comparison, each instance's color is consistent across all the UMAP results. Analysts can select one UMAP result from this view for more detailed-level investigations, as explained below.

Interpretation of a DR result. To help interpret the selected UMAP result, the view in Fig. 6-d shows the information on (1) the projection matrix used to generate the UMAP result and (2) each attribute's contribution to the characteristics of groups in the UMAP result.

The projection matrix, $\mathbf{P}$, contains the information of more (dis)regarded attributes for the generation of the result. This information is useful to understand the cause of the selected UMAP result's difference from the others. As described in Sec. 4.2, $\mathbf{P}$ can be either diagonal (i.e., $\mathbf{P} = \mathrm{diag}(\mathbf{w})$) or more dense (i.e., when using no constraint or $\mathbf{P} = \mathrm{diag}(\mathbf{w})\mathbf{M}\mathrm{diag}(\mathbf{v})$). When $\mathbf{P} = \mathrm{diag}(\mathbf{w})$, we visualize values of $\mathbf{w}$ as a bar chart, as shown on the left side of Fig. 6-d. For the other case, we visualize values in $\mathbf{P}$ as a heatmap using a diverging colormap (e.g., Fig. 8-b (left)).

Reviewing patterns shown in the DR result is essential to uncover analytical insights as well as to avoid deriving insights from false patterns due to excessive data transformation. The patterns are often examined through the comparison of data groups in the DR result [10, 30]. To assist group comparison, as shown on the right side of Fig. 6-d, the UI integrates an existing contrastive-learning-based interpretation method, called ccPCA, and the heatmap-based visualization [10]. ccPCA contrasts a target group with a background group to reveal highly-contributed attributes to the characteristics of the target. The attributes' contributions are obtained as a weight vector, where the larger magnitude, the stronger contribution to the target group's characteristics. In addition, the sign of the weight vector can represent the direction of the contribution when using the sign adjustment method [11]. For example, while `alcohol` in Fig. 6-d contributes to the characteristics of both `Cultivars 1` and `2`, according to their sign, they likely have higher and lower alcohol percentages than others, respectively. Also, to enable the comparison of attribute values of instances, we update the size of each point in Fig. 6-c when a certain attribute name is hovered in Fig. 6-d.

Although the UI uses predefined labels by default (e.g., the cultivar classes in Fig. 6), interactive refinement of groups can be performed with the lasso-selection available in Fig. 6-c and controls shown in Fig. 6-e. The changes in groups automatically update the attributes' contributions in Fig. 6-d. By default, when computing the contributions with ccPCA, each group is selected as a target group and the other groups are set as one background group. The UI also allows explicit selection of a background group. This is useful when the comparison of two specific groups is more desired.

Through a collective use of the above functionalities and visualizations, we can assess the DR result and patterns. When the observed groups do not result from false patterns, the projection matrix values, attributes' contributions, and distribution of attribute values should show some consistency. This is because the separation visible in the DR result should be highly related to the projection matrix, the differences should be captured in the attributes' contributions, and the attributes' contributions should reflect the attribute value distribution. We provide a concrete example in our case studies.

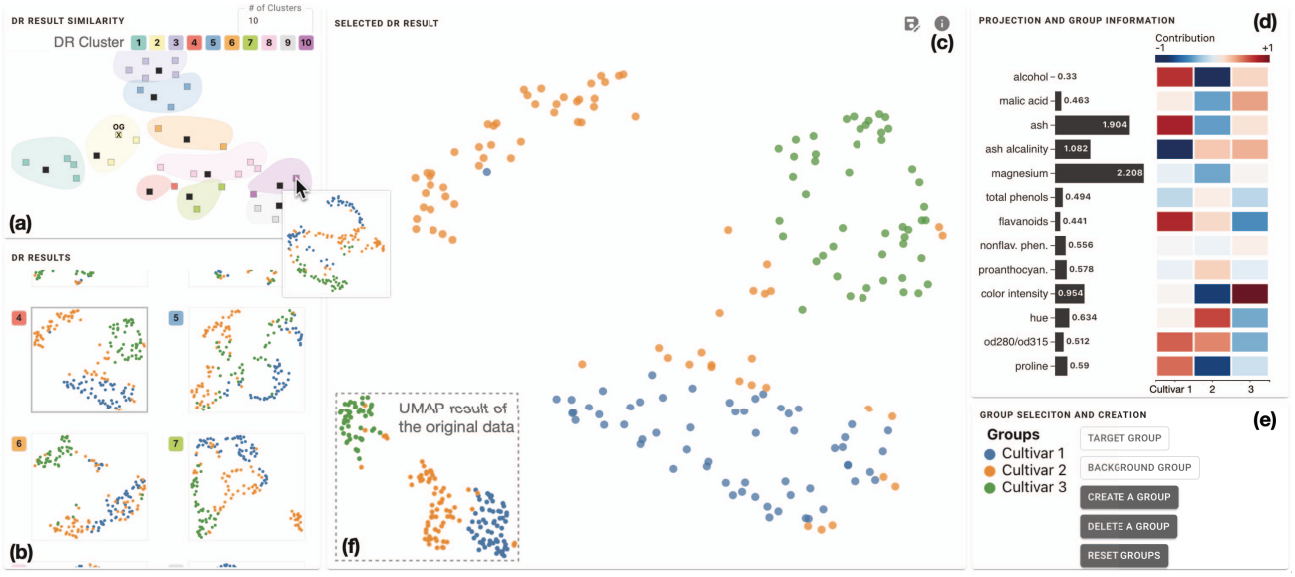

Figure 6: Analyzing the Wine dataset [8] with FEALM-UMAP and the UI. The UI shows the similarities of the generated DR results (a), a representative subset of the DR results (b), a single selected DR result (c), the information for interpreting the DR result (d), and the group information (e). The DR result of the original data is also shown as a reference (f).

8 Case Studies

We demonstrate the effectiveness of our approach through case studies on real-world datasets. Throughout all case studies, we generate UMAP results using two different constraints: $\mathbf{P} = \mathrm{diag}(\mathbf{w})$ and $\mathbf{P} = \mathrm{diag}(\mathbf{w})\mathbf{M}\mathrm{diag}(\mathbf{v})$. Here we only describe the essential information to present the analysis results. In the supplementary materials [1], we provide all the other details and one additional case study as an analysis example dealing with a large number of attributes (over 700). Note that the patterns uncovered in the case studies are difficult to identify with the aforesaid optimization method by Lehmann and Theisel [24] or attribute selection (refer to [1]).

8.1 Study 1: Diverse Categorization of Wines

We analyze the Wine dataset [8], which consists of 178 instances, 13 attributes, and cultivar labels. As shown in Fig. 6-f, DR on this dataset usually reveals three clusters highly related to the cultivars. With FEALM-UMAP, we seek patterns different from those clusters.

As shown in Fig. 6-b, FEALM-UMAP produces the results with greatly different patterns. We select a UMAP result that contains three clusters (see Fig. 6-c). These clusters (especially, the cluster mainly consisting of `Cultivar 1`) seem to contain different wines from the three clusters in the original UMAP result shown in Fig. 6-f. Also, the proximity of each cluster is clearly different from the original UMAP result (e.g., the clusters mainly consisting of `Cultivars 1` and `3` are placed close with each other in Fig. 6-c). As shown in Fig. 7-a1, we interactively define the clusters seen in Fig. 6-c as `Groups A`–`C`. From the auxiliary information displayed in Fig. 7-a2, the UMAP result is generated with the constraint of $\mathbf{P} = \mathrm{diag}(\mathbf{w})$, where `ash`, `ash alcalinity`, and `magnesium` have larger weights than others. Also, these attributes show strong contributions to each clusters' characteristics, especially for `Groups A` and `C` (see Fig. 7-a2(right)). We further verify the strong associations between the clusters and each of the three attributes. For example, as shown in Fig. 7-a1, `Group C` has small `magnesium`. According to existing research on this data [3], only these three attributes represent the mineral content of wines. Thus, FEALM-UMAP seems to find a new wine categorization that highly corresponds to the mineral content.

From Fig. 6-b, we select a representative UMAP result of DR Cluster 6 (orange) as another categorization example. This result contains multiple clusters, each of which is composed of multiple

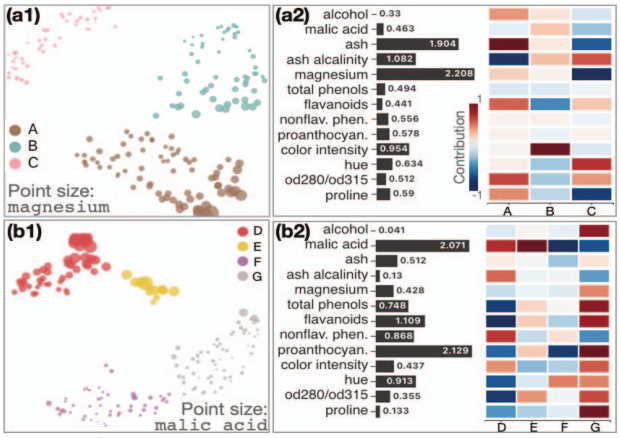

Figure 7: Study 1: Diverse categorizations of wines. DR results (a1, b1) selected from the Fig. 6-b and their auxiliary information (a2, b2).

cultivars. As shown in Fig. 7-b1 and b2, we create `Groups D`–`G` and visualize the related auxiliary information. From the weights and contributions in Fig. 7-b2, we see that multiple attributes, such as `malic acid`, `proanthocyanins`, `flavanoids`, strongly influence the forming of `Groups D`–`G`. These attributes are related to wine taste (e.g., `proanthocyanins` contributes to the dryness) [3]. As low weights are assigned for the attributes related to fermentation (i.e., `alcohol` and `proline`), mineral content (e.g., `ash`), and appearance (e.g., `color intensity`), we can say that FEALM-UMAP identifies `Groups D`–`G` that have the difference more in the taste.

8.2 Study 2: Investigation of Political Opinion Patterns

As a case with a larger number of instances, we analyze a survey dataset from the 2020 Cooperative Election Study [35], which consists of US residents' responses on various political opinions. From this dataset, we select 12 ordinal attributes/questions that do not have a high correlation with each other (specifically, less than 0.7 Pearson's correlation coefficient). We focus on instances/respondents who support either the Democratic (`Dem`) or Republican (`Rep`) party and discard instances that have missing values for the 12 attributes. After the above process, 4462 instances are included in `Dem` in con-

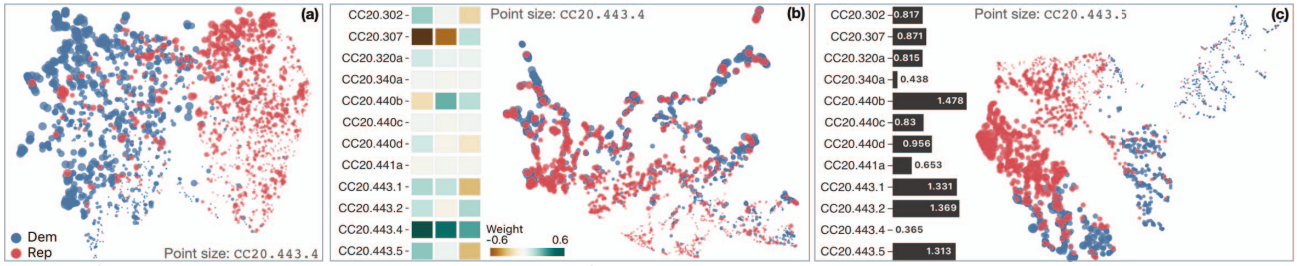

Figure 8: Study 2: Investigation of political opinion patterns in US residents. The original UMAP result (a) and representative FEALM-UMAP's results produced with different constraints—(b): $\mathbf{P} = \mathrm{diag}(\mathbf{w})\mathbf{M}\,\mathrm{diag}(\mathbf{v})$ and (c): $\mathbf{P} = \mathrm{diag}(\mathbf{w})$.

trast to 1154 instances in Rep. We randomly sampled 1154 instances from Dem to avoid producing patterns related to the sample size difference. The resulting dataset contains 2308 instances with 12 attributes. For detailed information of each attribute, refer to [35].

As in [18], DR on political survey data of US residents often only reveals two clusters related to the left-right ideology. In fact, as shown in Fig. 8-a, when applying UMAP to the dataset, we can only see such clusters corresponding to Dem and Rep. We utilize FEALM-UMAP to reveal political opinion patterns hidden in the data. Here we describe a few representative findings from our analysis.

As shown in Fig. 8-b, we find one UMAP result where Dem and Rep are heavily overlapped with each other. Because the information of the projection matrix, $\mathbf{P}$, is shown as a heatmap (see Fig. 8-b (left)), the result is generated with the constraint of $\mathbf{P} = \mathrm{diag}(\mathbf{w})\mathbf{M}\,\mathrm{diag}(\mathbf{v})$. Based on each attribute weight, we see that CC20.307 and CC20.443.4 have dominant weights in two of three columns of $\mathbf{P}$. While the question of CC20.307 is if the US police make the respondent feel safe or unsafe (1: mostly safe–4: mostly unsafe), CC20.443.4 is how the respondent would like to their legislature to spend money on law enforcement (1: greatly increase–4: greatly decrease). As these two attributes' weights have different signs (CC20.307: negative; CC20.443.4: positive), $\mathbf{P}$ seems to derive new features while debiasing the opinion of future money usage on law enforcement from the current opinion of the police. We can say that, within the resultant features by $\mathbf{P}$, there are no clear opinion differences between Dem and Rep unlike the result in Fig. 8-a.

Fig. 8-c shows another UMAP result, where we can still see the separation between Dem and Rep (Dem's instances tend to be around the bottom-right side) as well as new clusters that cannot be seen in Fig. 8-a. As shown in Fig. 8-c (left), large weights are assigned to CC20.440b (racial problems), CC20.443.1, CC20.443.2, and CC20.443.5 (legislature's money use on welfare, healthcare, and transportation/infrastructure, respectively). By interactively changing the point size based on each of these attributes, we observe that CC20.440b (racial problems) and CC20.443.2 (healthcare) closely associate with the separation between Dem and Rep. On the other hand, as seen in the sizes of the points in Fig. 8-c (right), CC20.443.5 is highly related to the clusters aligned along the diagonal direction (e.g., small CC20.443.5 can be seen around the top right). Thus, we can say that the opinions on the money use on transportation/infrastructure are diverse even within each party's supporters. By assigning large weights to the above attributes while minimizing the influences from certain attributes (e.g., CC20.340a, political ideology), FEALM-UMAP seems to find political subgroups that are difficult to find with the conventional use of DR.

9 Discussion

FEALM and FEALM-UMAP have provided a primary step in addressing the stated problem of hidden manifolds. We discuss our approach's limitations as observed through the theoretical and experimental analyses, and discuss further potential enhancements.

Scalability. As discussed in Sec. 5.4 and Sec. 6, FEALM-UMAP has limited scalability for the numbers of instances (n) and attributes (m). Because of NSD's time complexity, FEALM-UMAP is computationally expensive when n is large. Based on the results in Sec. 6, FEALM-UMAP is practical up until data with a few thousand instances when using a midrange computer. When m is large, on the other hand, the search space becomes very large; consequently, the NMM requires large numbers of initial solutions and evaluations to find solutions of adequate quality. Thus, we recommend preprocessing with PCA or attribute clustering for the case with large m. One potential approach for efficient optimization is to make all functions differentiable and use derivative-based solvers. We expect this can be achieved by utilizing differentiable variants of k-neighbor selections [33] as well as developing NSD for weighted graphs. While the equations used in ND and NetLSD [39] can be naturally extended weighted graphs, we need further investigations to understand their characteristics in the context of weighted graphs.

Reliability. Similar to other ML methods, FEALM-UMAP could suffer from overfitting when n is relatively small compared to m [17]. Therefore, FEALM-UMAP is suitable for data where n is considerably larger than m (e.g., n=1000, m=20). When n is relatively small, we can apply PCA to reduce m or define stronger constraints on the projection matrix (e.g., only data scaling). In addition to the projection constraint, FEALM-UMAP has several hyperparameters, m', λ_1, λ_2, q, β, r, which can also influence the DR results. We have suggested default values for q and β (i.e., q=50 and β=1 as explained in Sec. 5.2.1), while r (the number of different DR results) can be increased till the convergence of the optimization or set as large as possible based on the available computational power. The remaining hyperparameters that need to be adjusted are m', λ_1, and λ_2 (the number of latent features in a projection matrix and the weights for L1 and L2 norm-based regularizations). As in other ML methods, currently, we recommend manually searching appropriate values based on the observed results (e.g., the quality of optimization in Sec. 6). In future work, we would like to investigate automatic hyperparameter selection. To reduce the risk of false findings, our visual interface assists the analyst to inspect the obtained DR results with expert knowledge of their data.

Generalizability. FEALM is designed as a general framework for nonlinear DR methods. We emphasize that FEALM is applicable to various nonlinear DR methods, such as t-SNE, as discussed in Sec. 5. FEALM can also be used for linear DR methods. For example, Eq. 1 can be applied to the linear DR method designed by Lehmann and Theisel [24]. Furthermore, we can extend Eq. 2 to recommend sets of graph-related hyperparameters of DR (e.g., the number of neighbors in UMAP), which produce significantly different DR results [4]. This can be achieved by replacing a linear projection matrix with hyperparameters for the optimization's search parameters.

10 Conclusion

We have presented FEALM, a feature learning framework that enables investigation of data patterns via a conjoint use with nonlinear DR. The derived exemplifying method and visual interface have demonstrated the utility of FEALM for analyses of real-world datasets. This work also exposes the limitations of conventional ways of data exploration using dimensionality reduction and thus contributes toward the maximal utilization of data.

ACKNOWLEDGMENTS

This work has been supported in part by the Knut and Alice Wallenberg Foundation through Grant KAW 2019.0024, the U.S. National Science Foundation through Grant ITE-2134901, and the National Institute of Health through Grant 1R01CA270454-01.

REFERENCES

[1] Supplementary materials. https://takanori-fujiwara.github.io/s/fealm/.
[2] A. Abid, M. J. Zhang, V. K. Bagaria, and J. Zou. Exploring patterns enriched in a dataset with contrastive principal component analysis. *Nat Commun*, 9(1):2134, 2018.
[3] J. Barth, D. Katumullage, C. Yang, and J. Cao. Classification of wines using principal component analysis. *J Wine Econ*, 16(1):56–67, 2021.
[4] A. Chatzimparmpas, R. M. Martins, and A. Kerren. t-viSNE: Interactive assessment and interpretation of t-sne projections. *IEEE Trans Vis Comput Graph*, 26(8):2696–2714, 2020.
[5] C. Collins, G. Penn, and S. Carpendale. Bubble Sets: Revealing set relations with isocontours over existing visualizations. *IEEE Trans Vis Comput Graph*, 15(6):1009–1016, 2009.
[6] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *J Mach Learn Res*, 16(1):2859–2900, 2015.
[7] M. W. Dorrity, L. M. Saunders, C. Queitsch, S. Fields, and C. Trapnell. Dimensionality reduction by UMAP to visualize physical and genetic interactions. *Nat Commun*, 11(1):1–6, 2020.
[8] D. Dua and C. Graff. UCI machine learning repository. https://archive.ics.uci.edu/ml, 2019.
[9] L. Ertöz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proc. SDM*, pp. 47–58, 2003.
[10] T. Fujiwara, O.-H. Kwon, and K.-L. Ma. Supporting analysis of dimensionality reduction results with contrastive learning. *IEEE Trans Vis Comput Graph*, 26(1):45–55, 2020.
[11] T. Fujiwara, Shilpika, N. Sakamoto, J. Nonaka, K. Yamamoto, and K.-L. Ma. A visual analytics framework for reviewing multivariate time-series data with dimensionality reduction. *IEEE Trans Vis Comput Graph*, 27(2):1601–1611, 2021.
[12] T. Fujiwara, X. Wei, J. Zhao, and K.-L. Ma. Interactive dimensionality reduction for comparative analysis. *IEEE Trans Vis Comput Graph*, 28(1):758–768, 2022.
[13] F. Gao and L. Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Comput Optim Appl*, 51(1):259–277, 2012.
[14] S. García, J. Luengo, and F. Herrera. *Data Preprocessing in Data Mining*, vol. 72. Springer, 2015.
[15] M. Gleicher. Explainers: Expert explorations with crafted projections. *IEEE Trans Vis Comput Graph*, 19(12):2042–2051, 2013.
[16] C. Goodall. Procrustes methods in the statistical analysis of shape. *J R Stat Soc Series B: Stat Methodol*, 53(2):285–321, 1991.
[17] Y. Guo, T. Hastie, and R. Tibshirani. Regularized linear discriminant analysis and its application in microarrays. *Biostat*, 8(1):86–100, 2007.
[18] C. Hare, T.-P. Liu, and R. N. Lupton. What Ordered Optimal Classification reveals about ideological structure, cleavages, and polarization in the American mass public. *Public Choice*, 176(1):57–78, 2018.
[19] D. Jäckle, M. Hund, M. Behrisch, D. A. Keim, and T. Schreck. Pattern Trails: Visual analysis of pattern transitions in subspaces. In *Proc. VAST*, pp. 1–12, 2017.
[20] H. Jeon, H.-K. Ko, J. Jo, Y. Kim, and J. Seo. Measuring and explaining the inter-cluster reliability of multidimensional projections. *IEEE Trans Vis Comput Graph*, 28(1):551–561, 2022.
[21] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. ICNN*, vol. 4, pp. 1942–1948, 1995.
[22] D. Kobak and G. C. Linderman. Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nat Biotechnol*, 39(2):156–157, 2021.
[23] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans Knowl Discov Data*, 3(1):1–58, 2009.
[24] D. J. Lehmann and H. Theisel. Optimal sets of projections of high-dimensional data. *IEEE Trans Vis Comput Graph*, 22(1):609–618, 2016.
[25] S. Liu, P.-T. Bremer, J. Jayaraman, B. Wang, B. Summa, and V. Pascucci. The Grassmannian Atlas: A general framework for exploring linear projections of high-dimensional data. *Comput Graph Forum*, 35(3):1–10, 2016.
[26] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Trans Vis Comput Graph*, 23(3):1249–1268, 2016.
[27] S. McCabe, L. Torres, T. LaRock, S. A. Haque, C.-H. Yang, H. Hartle, and B. Klein. netrd: A library for network reconstruction and graph distances. *J Open Source Softw*, 6(62):2990, 2021.
[28] L. McInnes, J.Healy, and J.Melville. UMAP:uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*, 2018.
[29] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Adv Neural Inf Process Syst*, 14, 2001.
[30] L. G. Nonato and M. Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Trans Vis Comput Graph*, 25(8):2650–2673, 2019.
[31] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explor*, 6(1):90–105, 2004.
[32] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in Python. *J Mach Learn Res*, 12:2825–2830, 2011.
[33] T. Plötz and S. Roth. Neural nearest neighbors networks. *Adv Neural Inf Process Syst*, 31, 2018.
[34] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, et al. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Trans Vis Comput Graph*, 23(1):241–250, 2017.
[35] B. Schaffner, S. Ansolabehere, and S. Luks. Cooperative Election Study Common Content, 2020. Harvard Dataverse, https://doi.org/10.7910/DVN/E9N6PH, 2021. Accessed: 2022-10-14.
[36] G. Sun, S. Zhu, Q. Jiang, W. Xia, and R. Liang. EvoSets: Tracking the sensitivity of dimensionality reduction results across subspaces. *IEEE Trans Big Data*, 8(6):1566–1579, 2022.
[37] A. Tatu, F. Maaß, I. Färber, E. Bertini, T. Schreck, T. Seidl, and D. Keim. Subspace search and visualization to make sense of alternative clusterings in high-dimensional data. In *Proc. VAST*, pp. 63–72, 2012.
[38] J. Townsend, N. Koep, and S. Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *J Mach Learn Res*, 17(137):1–5, 2016.
[39] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller. NetLSD: Hearing the shape of a graph. In *Proc. KDD*, pp. 2347–2356, 2018.
[40] L. Van Der Maaten. Accelerating t-SNE using tree-based algorithms. *J Mach Learn Res*, 15(1):3221–3245, 2014.
[41] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *J Mach Learn Res*, 9:2579–2605, 2008.
[42] L. van der Maaten, E. Postma, and J. van den Herik. Dimensionality reduction: A comparative review. *J Mach Learn Res*, 10:66–71, 2009.
[43] B. Wang and K. Mueller. The Subspace Voyager: Exploring high-dimensional data along a continuum of salient 3D subspaces. *IEEE Trans Vis Comput Graph*, 24(2):1204–1222, 2017.
[44] Y. Wang, J.Li, F.Nie, H.Theisel, M.Gong, and D. J. Lehmann. Linear discriminative star coordinates for exploring class and cluster separation of high dimensional data. *Comput Graph Forum*, 36(3):401–410, 2017.
[45] L. Wilkinson, A. Anand, and R. Grossman. High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. *IEEE Trans Vis Comput Graph*, 12(6):1363–1372, 2006.
[46] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Trans Pattern Anal Mach Intell*, 29(1):40–51, 2006.
[47] X. Yuan, D. Ren, Z. Wang, and C. Guo. Dimension Projection Matrix/Tree: interactive subspace visual exploration and analysis of high dimensional data. *IEEE Trans Vis Comput Graph*, 19(12):2625–2633, 2013.
[48] E. Zahara and Y.-T. Kao. Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Syst Appl*, 36(2):3880–3886, 2009.
[49] R. Zebari, A. Abdulazeez, D. Zeebaree, et al. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *J Appl Sci Technol Trends*, 1(2):56–70, 2020.
[50] F. Zhou, J. Li, W. Huang, Y. Zhao, X. Yuan, X. Liang, and Y. Shi. Dimension reconstruction for visual exploration of subspace clusters in high-dimensional data. In *Proc. PacificVis*, pp. 128–135, 2016.