Fast, Visual and Interactive Semi-supervised Dimensionality Reduction

Dimitris Spathis[1,2], Nikolaos Passalis[1], Anastasios Tefas[1]

[1]Aristotle University of Thessaloniki, Greece
[2]University of Cambridge, UK

Abstract. Recent advances in machine learning allow us to analyze and describe the content of high-dimensional data ranging from images and video to text and audio data. In order to visualize that data in 2D or 3D, usually Dimensionality Reduction (DR) techniques are employed. Most of these techniques produce static projections without taking into account corrections from humans or other data exploration scenarios. In this work, we propose a novel interactive DR framework that is able to learn the optimal projection by exploiting the user interactions with the projected data. We evaluate the proposed method under a widely used interaction scenario in multidimensional projection literature, i.e., project a subset of the data, rearrange them better in classes, and then project the rest of the dataset, and we show that it greatly outperforms competitive baseline and state-of-the-art techniques, while also being able to readily adapt to the computational requirements of different applications.

Keywords: Interactive Dimensionality Reduction, Similarity-embeddings

1 Introduction

Recent advances in statistical machine learning allow us to tackle hard real-world problems such as machine translation, speech recognition, image captioning and developing self-driving cars [1]. In order to train machine learning models for the aforementioned tasks huge amounts of data are required. However, the process of data collection and processing is costly since human annotators must validate the *ground truth* of each dataset. This issue is further aggravated by the limited involvement of domain experts, who are also potential users of such systems, in the training process [2].

Interactive systems that learn by users have been proposed during the last decade for applications like image segmentation [3]. While these systems allowed to manipulate the input or some parameters of the model, they rarely offered ways to interact with the data points *per se*. To further increase the involvement of end-users in training machine learning systems, we also have to think about the interface and cognitive load we provide them with. In this work we research on dimensionality reduction techniques, which provide ways of projecting high-dimensional data in 2D. While dimensionality reduction is used for many purposes, such as to reduce storage space and preprocessing time, we focus on its usage for interactive visualization. By visualizing the data in a two-dimensional (2D) or three-dimensional (3D) space, we make it easier for humans to understand the structure of data in a manner that feels natural.

For a dataset that consists of N n-dimensional points $\mathbf{p}_i \in \mathbb{R}^n$, dimensionality reduction (DR) can be defined as a function which maps each point $\mathbf{p}_i \in \mathbb{R}^n$ to a low-dimensional point $\mathbf{q}_i \in \mathbb{R}^m$:

$$f_{\mathbf{W}} : \mathbb{R}^n \rightarrow \mathbb{R}^m. \tag{1}$$

Here, n is typically large (from tens to thousands of dimensions), m represents number of dimensions in the low-dimensional space, typically 2 or 3, while $\mathbf{W}$ denotes the parameters of the function used to perform the projection.

Even though many dimensionality reduction techniques leverage different concepts, most of them share a common property: the objective function used to optimize the final projection is a linear combination of pairwise distances between the data points. However, this renders most of the existing dimensionality reduction techniques prone to outliers, while increases the difficulty of interacting with the data since it is not always straightforward to manipulate the distance between different points (there is no universally small or large distance). The aforementioned limitations highlights the need for methods that use bounded similarity metrics instead of unbounded distance metrics. An illustrative example is the recent success of t-SNE [4], which transforms distances to probabilities using a non-linearity (Gaussian kernel). That way it effectively addresses the so-called *crowding problem* [4]. In this work, another recently proposed powerful similarity-based dimensionality method, the Similarity Embedding Framework (SEF) [5], is adapted towards interactive visualization tasks.

The main contribution of this paper is the extension of the Similarity Embedding Framework (SEF) towards efficiently handing interactive visualization tasks for semi-supervised learning. To this end, we adopt a common interaction scenario in multidimensional projection literature. First, a subset of the data, called control points, is projected, then the user rearrange them better in classes or clusters, and finally the rest of the dataset is projected based on that manipulation. The proposed method is evaluated on four datasets from a diverse range of domains and it is demonstrated that it outperforms the other evaluated baseline and state-of-the-art methods. The proposed approach can be combined with any differentiable projection function, ranging from fast and lightweight linear projection functions to more powerful kernel projections and deep neural networks. This allows the proposed method to readily adapt to different use cases, e.g., a linear projection function can be used for providing lightweight interactive projections on mobile devices with limited computing power or used on the client-side of a web application, while a more powerful and complex neural network can be used for scientific visualization tasks, e.g., visualizing and interacting with genome data [6, 7], when more powerful infrastructure is available.

The rest of the paper is structured as follows. First, the related work is briefly discussed and compared to the proposed approach in Section 2. Then the proposed approach is presented in detail (Section 3) and evaluated (Section 4). Finally, conclusions are drawn and future work is discussed in Section 5.

2 Related work

Popular dimensionality reduction techniques include linear methods, such as Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA) [8], and non-linear ones, such as Multidimensional Scaling (MDS) [9], t-Distributed Stochastic Neighbor Embedding (t-SNE) [4], and Uniform Manifold Approximation and Projection (UMAP) [10].

A group of literature (*multidimensional projection*) focuses on observation-level interaction [11] and uses some seeding direct manipulation points (also called *"control points"*), which are a subset of original data points. The intuition goes that by manipulating a subset of data points, a mapping function is (implicitly or explicitly) learned. Then, this function is used (or approximated) to project the rest of the dataset. Numerous approaches related to control-point manipulation have been suggested. The *Local Affine Multidimensional Projection* (LAMP) starts by projecting a subset of control points and then interpolates the remaining points through orthogonal affine mappings, using the Singular Value Decomposition (SVD) method [12]. The *Part-Linear Multidimensional Projection* (PLMP) [13] allows for scaling to big datasets by first constructing a linear map of the control points using *Force Scheme* [14]. Next, this mapping is used to project the the remaining points. Finally, the *Kernel-based Linear Projection* (KELP) [15], which is a recent state-of-the-art interactive visualization method, allows for visualizing how *kernel functions* project the data in high-dimensional spaces, while allowing for interacting with the learned projection.

To the best of our knowledge the proposed method is the first interactive visualization method that uses similarity measures to extract information for users' interactions with the data and then learns a projection function by exploiting this information. Building upon the SEF allows for better handling possible outliers and provides an intuitive way for users to perform direct manipulation of data points. Furthermore, the proposed method can be used with fast linear projection functions, that can be readily implemented on mobile devices or other systems with limited computational power, while at the same time providing high-quality and responsive interactive projections. Finally, the proposed method can be initialized by cloning any of the existing visualization or DR methods for providing the initial control points, significantly increasing its flexibility (note that even methods that do not provide out-of-sample extensions, such as t-SNE, can be readily used for the initialization).

3 Proposed Method

First we briefly review the Similarity Embedding Framework [5], upon which the proposed method is built. Then, we derive and discuss the proposed fast interactive data visualization method.

Let $S(\mathbf{x}_i, \mathbf{x}_j)$ be the pairwise similarity between the data points $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that a similarity metric S is a bounded function that ranges between 0 and 1 and expresses the proximity between two points. Then, the similarity matrix of the projected data is defined as $[\mathbf{P}]_{i,j} = S(f_{DR}(\mathbf{x}_i), f_{DR}(\mathbf{x}_j))$, where i is a row and j is a column of this matrix and f_{DR} is the projection function.

SEF's main goal is to learn a projection in which the similarities in the low-dimensional space, i.e., in our case the visual space, are as close as possible to a selected "target". The target similarity matrix $\mathbf{T}$ is a square matrix that can be the result of many methods, such as direct manipulation of data points (as in out case), or other DR techniques (if we want to mimic them), such as PCA, LDA, t-SNE, etc. In order to learn the projection function $f_{DR}(\mathbf{x})$ we optimize the following objective function:

$$J_s = \frac{1}{2 \parallel \mathbf{M} \parallel_1} \sum_{i \neq j}^{N} [\mathbf{M}]_{i,j} ([\mathbf{P}]_{i,j} - [\mathbf{T}]_{i,j})^2, \tag{2}$$

where $\mathbf{M}$ is a matrix acting as a weighting mask defining the importance of attaining the target similarity of the data and $\parallel \mathbf{M} \parallel_1$ is the l_1 norm of matrix $\mathbf{M}$. When each data point pair achieves its target similarity, the objective function (2) is minimized, while when a pair has different similarity from its target, it is getting penalized.

Although $\mathbf{T}$ can be any target that we want to achieve during the projection, we use the Gaussian kernel (also known as Heat kernel) to define the similarity between the projected points, i.e.:

$$S(\mathbf{x}_i, \mathbf{x}_j) = exp(- \parallel \mathbf{x}_i - \mathbf{x}_j \parallel_2^2 / \sigma_p), \tag{3}$$

where σ_p acts a scaling factor. Therefore, the final similarity matrix $\mathbf{P}$ is defined as:

$$[\mathbf{P}]_{i,j} = exp(- \parallel f_{DR}(\mathbf{x}_i) - f_{DR}(\mathbf{x}_j) \parallel_2^2 / \sigma_p), \tag{4}$$

Note that among the options that the SEF provides, is to clone existing DR techniques. Let $c(\mathbf{x})$ be a technique to be cloned. Then, SEF can mimic $c(\mathbf{x})$ by setting the target matrix as:

$$[\mathbf{T}]_{i,j} = exp(- \frac{\parallel c(\mathbf{x}_i) - c(\mathbf{x}_j) \parallel_2^2}{\sigma_{copy}}), \tag{5}$$

where σ_{copy} is the scaling factor used to calculate the similarities between the low-dimensional points, as projected using the techniques that is to be cloned. This approach can be used to initialize the employed projection function for providing the initial control points. Note the great flexibility of the proposed method that can clone virtually any visualization or DR method, including methods that do not provide out-of-sample-extensions, such as t-SNE.

The projection function f_{DR} could be defined in multiple ways, ranging from simple fast linear transformations to non-linear methods, such as kernel projections and deep neural networks. In order to minimize the loss in objective function (2), gradient descent is used. Therefore, it is required to calculate the derivative of the objective function J_s with respect to the parameters of each projection.

One of the simplest projection methods is the linear transformation of the input space, such that $f_{DR}(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$, where $\mathbf{W}$ is the projection matrix. Let the relationship between the original data $\mathbf{X}$ and the projected $\mathbf{Y}$ be that of $\mathbf{y}_i = f_{DR}(\mathbf{x}_i)$. The derivative of the objective function J_s when a linear transformation is used is calculated as:

$$\frac{\partial J_s}{\partial [\mathbf{W}]_{kt}} = \frac{1}{\parallel \mathbf{M} \parallel_1} \sum_{i=1}^{N} \sum_{j=1}^{N} [\mathbf{M}]_{i,j} ([\mathbf{P}]_{i,j} - [\mathbf{T}]_{i,j}) \frac{\partial [\mathbf{P}]_{i,j}}{\partial [\mathbf{W}]_{kt}}, \tag{6}$$

where

$$\frac{\partial [\mathbf{P}]_{i,j}}{\partial [\mathbf{W}]_{kt}} = -\frac{2}{\sigma_p} [\mathbf{P}]_{i,j} ([\mathbf{Y}]_{it} - [\mathbf{Y}]_{jt})([\mathbf{X}]_{ik} - [\mathbf{X}]_{jk}). \tag{7}$$

The objective function (2) is optimized using gradient descent:

$$\varDelta \mathbf{W} = -\eta \frac{\partial J}{\partial \mathbf{W}}, \tag{8}$$

where η is the learning rate. In this work the Adam algorithm is used for the optimization [16], since it has been proved to be fast and reliable.

On the other hand, kernel methods are known to provide superior solutions, since they transform the input space into a higher dimensional one in order to solve the problem in a linear manner there. Let $\boldsymbol{\Phi} = \phi(\mathbf{X})$ be the matrix of data in the high dimensional space (also known as Hilbert space), where $\phi(\mathbf{x})$ is a function that projects the data in a higher dimensional space. In a similar way with the linear version, we seek to learn a linear mapping from the Hilbert space into the visual space. We use the Representer theorem to express the matrix $\mathbf{W}$ as a linear combination of the training data $\mathbf{X}$:

$$\mathbf{W} = \phi(\mathbf{X})^T = \boldsymbol{\Phi}^T \mathbf{A}, \tag{9}$$

where $\mathbf{A}$ is the coefficient matrix of the linear combination. Therefore, the final projection is derived as:

$$\mathbf{Y}^T = \mathbf{W}^T \boldsymbol{\Phi}^T = \mathbf{A}^T \boldsymbol{\Phi}\boldsymbol{\Phi}^T = \mathbf{A}^T \mathbf{K}, \tag{10}$$

where $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ is the kernel matrix, i.e., it contains the inner products of data in the high-dimensional space. Similar to most kernel techniques, we exploit the so-called *kernel trick*, meaning that we can calculate the matrix $\mathbf{K}$ without explicitly calculating the inner products in Hilbert space. In this work, we use the popular RBF kernel:

$$[\mathbf{K}]_{i,j} = exp(\parallel \mathbf{x}_i - \mathbf{x}_j \parallel_2^2 / \gamma^2). \tag{11}$$

Compared to the linear version we have to learn the matrix $\mathbf{A}$ instead of the matrix $\mathbf{W}$. Regarding the optimization, the derivative $\frac{\partial [\mathbf{P}]_{i,j}}{\partial [\mathbf{A}]_{kt}}$ is similarly calculated:

$$\frac{\partial J}{\partial[\mathbf{A}]_{kt}} = \frac{1}{||\mathbf{M}||_1}\sum_{i=1}^{N}\sum_{j=1}^{N}[\mathbf{M}]_{ij}([\mathbf{P}]_{ij} - [\mathbf{T}]_{ij})\frac{\partial[\mathbf{P}]_{ij}}{\partial[\mathbf{A}]_{kt}}, \tag{12}$$

where

$$\frac{\partial[\mathbf{P}]_{ij}}{\partial[\mathbf{A}]_{kt}} = -\frac{2}{\sigma_P}[\mathbf{P}]_{ij}([\mathbf{Y}]_{it} - [\mathbf{Y}]_{jt})([\mathbf{K}]_{ik} - [\mathbf{K}]_{jk}).$$

Algorithm 1 Interactive Similarity-based Dimensionality Reduction Learning Algorithm

Input: A matrix $\mathbf{X} = [\mathbf{x}_i, ..., \mathbf{x}_N]$ of N data points, the subset of the control points $\mathbf{X}_s$, and a technique $s(\mathbf{x})$ that can be used for initializing the projection

Output: Projected dataset $\widetilde{\mathbf{Y}}$.

```
1: procedure InteractiveDRLearning
2:     $\boldsymbol{T} \leftarrow$ clone (s($\mathbf{X}_s$))  ▷ Clone Force Scheme, t-SNE, or any other DR technique using (5).
3:     $\boldsymbol{W} or \boldsymbol{A} \leftarrow$ SimilarityEmbeddingLearning($\boldsymbol{X}_s, \boldsymbol{T}$)  ▷ Learn projection using (6).
4:     $\boldsymbol{Y_s} \leftarrow$ project ($\boldsymbol{X}_s$)  ▷ Project control points using projection matrix $\mathbf{W}$ or $\mathbf{A}$.
5:     $\widetilde{\boldsymbol{Y}}_s \leftarrow$ manipulate ($\boldsymbol{Y}_s$)  ▷ Adjust control points on interactive scatter plot.
6:     $\widetilde{\boldsymbol{T}} \leftarrow$ clone ( $\widetilde{\boldsymbol{Y_s}}$)  ▷ Clone control points using (5).
7:     $\widetilde{\boldsymbol{W}} or \widetilde{\boldsymbol{A}} \leftarrow$ SimilarityEmbeddingLearning($\widetilde{Y_s}, \widetilde{T}$)  ▷ Learn projection using (6).
8:     $\widetilde{\boldsymbol{Y}} \leftarrow$ project ($\boldsymbol{X}$)  ▷ Project original dataset using projection matrix $\widetilde{\boldsymbol{W}}$ or $\widetilde{\boldsymbol{A}}$.
9: return the projected dataset $\widetilde{\boldsymbol{Y}}$
```

The complete proposed interactive dimensionality reduction learning algorithm is summarized in Algorithm 1. More formally, let $\mathbf{X} = [\mathbf{x}_i, ..., \mathbf{x}_N]$ be the data matrix (in the original feature space) and $\mathbf{X}_s = [\mathbf{x}_{s_i}, ..., \mathbf{x}_{s_{N_i}}]$ a subset of the data that is used as control points. Let $\mathbf{Y}_s$ be the projection of control points $\mathbf{X}_s$ in the visual space and $\mathbf{Y}$ the final projection of $\mathbf{X}$ in the visual space. First, the projection is initialized by cloning an existing technique (lines 2-3). The user interacts and manipulates the control points $\mathbf{Y}_s$ and their respective coordinates producing $\widetilde{\mathbf{Y}_s}$ (lines 4-5). Finally, the previous projection is optimized according to the user's interaction (lines 6-7) and the whole dataset is visualized by calculating $\widetilde{\mathbf{Y}}$ (line 8). Note that tilde'd characters are used to denote the results after the manipulation.

It worth noting the great flexibility of the proposed approach. The method can be initialized by cloning any of the existing visualization or DR methods, e.g., t-SNE, UMAP, etc. Then, any differentiable projection function can be used to project the data into the lower dimensional space. The complexity of the projection function can be adapted to the needs of each application, as it was already mentioned before. For example, a linear projection function can be used for mobile devices, while more powerful non-linear projection functions can be used when more computational resources are available. Note that the complexity of kernel methods can be also readily reduced using various methods, such as using low-rank Nyström approximations [17]. Furthermore, the optimization can be adapted to the available memory by using small batches that fit in memory instead of computing the whole similarity matrix described in (4). Note that in this case the samples must be appropriately shuffled before each optimization epoch to ensure that different data pairs are used for the optimization.

4 Experiments

For evaluating the proposed techniques, the following datasets from a wide range of domains are used: a handwritten digit recognition dataset (MNIST) [18], an outdoor image segmentation dataset (Segmentation) [19], a breast cancer diagnosis dataset (Cancer) [19] and a chemical–wine recognition dataset (Wine) [19]. The used datasets are summarized in Table 1.

Table 1. Description of the datasets used for the conducted experiments.

Dataset	Size	Dimensions
MNIST	60000	784
Segmentation	2100	19
Cancer	569	30
Wine	178	13

To ensure a fair comparison with other techniques proposed in the literature we use the same control points and the same manipulations for all the evaluated methods. That is, the same control points after the user's manipulations are fed into the proposed technique and the other evaluated techniques, i.e., the LAMP, the PLMP and the KELP. The public available implementation of each of these techniques was used in the conducted experiments, while the default parameter selection procedure was used. Also, the number of iterations of our optimization is fixed to 500. By inspecting the loss minimization curve during the experiments though, we saw that it converges way before the 500th iteration most of the times. With that in mind, our computation time could be even lower.

In the conducted experiments we estimate the class separation, clustering assignment, and neighbor error. The class separation is evaluated with the nearest centroid algorithm. Clustering assignment is measured with the Silhouette coefficient [20], that measures the cohesion and separation between grouped data. In order to visualize the neighbor error per data point, we calculate the 10 neighbors of each point in the visual space and we sum their Euclidean distance in the high-dimensional space. This score is then normalized to [0,1], so that a point that its 2D neighbors are far in the feature space, gets a high error close to 1.

The class separation evaluation results are shown in Table 2. For the MNIST dataset 1240 digits that belongs in four classes (2, 4, 7 and 9) were used. As proposed in the literature [13], $\sqrt{n}$ control points are used, where n is the dataset

Table 2. Mean classification precision of projected datasets after manipulation of control points. Best results marked bold. Standard deviation of 10 runs in parenthesis.

Data	Initialization	Nearest Centroid				
		KELP	LAMP	PLMP	Proposed Linear	Proposed Kernel
Wine	PCA	66.11 (6.71)	59.86 (10.42)	58.79 (9.93)	**81.98** (7.23)	71.56 (4.09)
	tSNE	61.42 (4.95)	63.52 (7.23)	52.63 (8.92)	62.50 (13.04)	**70.19 (4.44)**
	Force	62.76 (11.07)	61.10 (11.45)	52.45 (9.30)	57.97 (7.03)	**63.95 (6.33)**
Cncr	PCA	89.84 (2.21)	83.38 (3.22)	63.04 (6.12)	71.77 (7.83)	**90.68 (0.75)**
	tSNE	87.91 (5.01)	82.87 (2.54)	61.74 (5.35)	72.17 (4.94)	**90.95 (1.80)**
	Force	86.99 (5.31)	83.97 (1.78)	59.91 (4.89)	74.37 (5.34)	**87.84 (8.14)**
Segm.	PCA	58.37 (6.01)	50.32 (6.33)	44.20 (8.84)	50.45 (11.05)	**60.98 (5.52)**
	tSNE	62.43 (4.79)	53.91 (4.11)	44.24 (6.96)	55.59 (11.72)	**64.77 (3.50)**
	Force	59.70 (5.60)	53.44 (4.95)	39.43 (8.79)	54.71 (6.06)	**63.40 (4.22)**
MNIST	PCA	63.12 (5.59)	72.79 (5.84)	29.67 (4.30)	36.04 (3.10)	**77.92 (4.53)**
	tSNE	59.85 (11.84)	**74.00 (3.43)**	29.72 (3.62)	39.11 (4.06)	68.57 (5.73)
	Force	60.87 (10.52)	72.20 (5.32)	31.76 (2.31)	45.40 (6.32)	**75.52 (5.90)**

size. The proposed technique, especially when used with a kernel-based projection function, almost always outperforms all the other compared techniques, regardless the used initialization scheme and dataset.

Figure 1 illustrates the projected points along with their classes for all the evaluated techniques using the Segmentation dataset (the kernel variant is used for the proposed method), while in Figure 2 the Cancer dataset is visualized (again the kernel variant of the proposed method is used). The proposed method always leads to better class separation compared to the proposed methods, as well as it provides more stable behavior on previously unseen data (note that the PLMP and the KELP methods collapses in some cases).

Next, we evaluate the proposed method using a clustering setup (the number of clusters equals the number of classes) using the silhouette coefficient which attempts to count how tightly grouped all the data in each cluster are. Note that the silhouette coefficient normally ranges from -1 to 1. However, to improve the readability of results, values have been scaled to [-100, 100]. The results are reported in Table 3. The proposed technique achieves the highest score in every dataset, while the kernel version of the proposed technique still greatly outperforms the linear variant.

Table 4 shows the neighbor error, which evaluates how far the 10 neighbors of each point in the visual space are in the original high-dimensional space. As an error metric, lower-values indicate better projections. For the neighbor error metric, the results are more diverse than the previous two metrics. While the proposed kernel version performs considerably better than the linear and baselines in first two datasets (Wine and Cancer), it presents similar results with the linear and baselines in the other two datasets (Segmentation and MNIST). The neighbor error for the Cancer dataset (same setup as before) is visualized in Figure 3. Note that the proposed method better captures the manifold structure of the data, creating smooth low neighbor error regions.

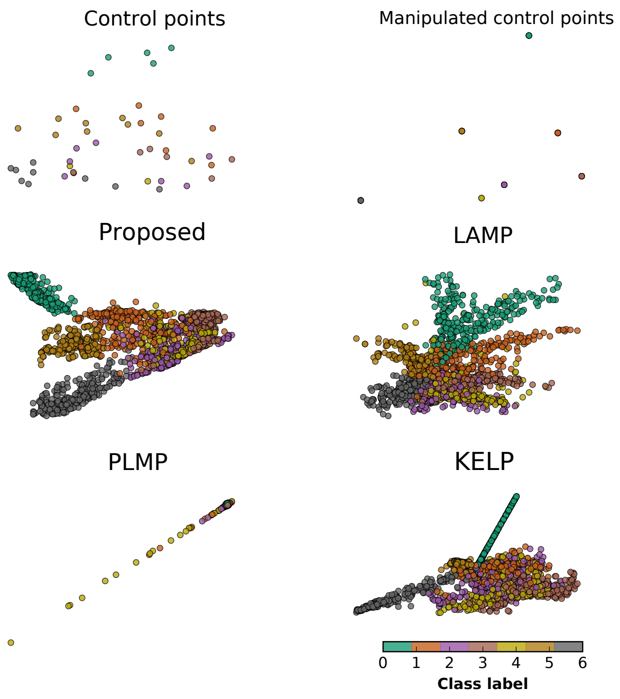

Fig. 1. Visualizing the Image Segmentation dataset. Classes correspond to the following labels: green – sky, orange – cement, purple – window, bordeaux – brickface, yellow – foliage, mustard – path, gray – grass. (figure best viewed in color)

Table 3. Average silhouette coefficient after manipulation of control points. Best results marked bold. Standard deviation of 10 runs in parenthesis.

Data	Initialization	Silhouette Coefficient				
		KELP	LAMP	PLMP	Proposed Linear	Proposed Kernel
Wine	PCA	12.52 (7.00)	-9.28 (3.69)	0.55 (4.63)	**28.13 (12.22)**	19.18 (4.23)
	tSNE	11.95 (6.58)	-14.70 (4.56)	-0.76 (6.05)	8.23 (8.58)	**14.49 (8.72)**
	Force	**15.96 (7.47)**	-11.41 (5.29)	1.60 (4.44)	4.71 (4.75)	12.11 (9.35)
Cncr	PCA	55.15 (13.60)	34.14 (4.26)	12.22 (5.49)	19.26 (7.63)	**57.93 (4.75)**
	tSNE	53.75 (13.78)	36.07 (4.22)	10.89 (3.13)	21.64 (5.60)	**60.35 (5.04)**
	Force	52.34 (15.51)	36.01 (5.09)	9.08 (3.56)	22.74 (6.93)	**56.15 (10.45)**
Segm.	PCA	6.02 (5.87)	-0.61 (5.26)	-11.66 (10.09)	-0.74 (10.87)	**16.00 (5.07)**
	tSNE	7.11 (2.95)	0.02 (3.52)	-10.58 (7.63)	5.54 (10.53)	**18.56 (4.49)**
	Force	6.35 (4.39)	-0.70 (3.68)	-15.48 (5.60)	1.55 (7.16)	**17.12 (3.54)**
MNIST	PCA	-24.39 (5.28)	17.28 (4.71)	-6.78 (0.93)	-6.38 (1.31)	**25.14 (4.65)**
	tSNE	-22.09 (9.28)	**19.26 (3.43)**	-6.05 (1.75)	-4.85 (2.00)	13.42 (6.81)
	Force	-28.30 (7.97)	17.65 (4.68)	-6.13 (0.95)	-3.02 (2.85)	**22.07 (6.98)**

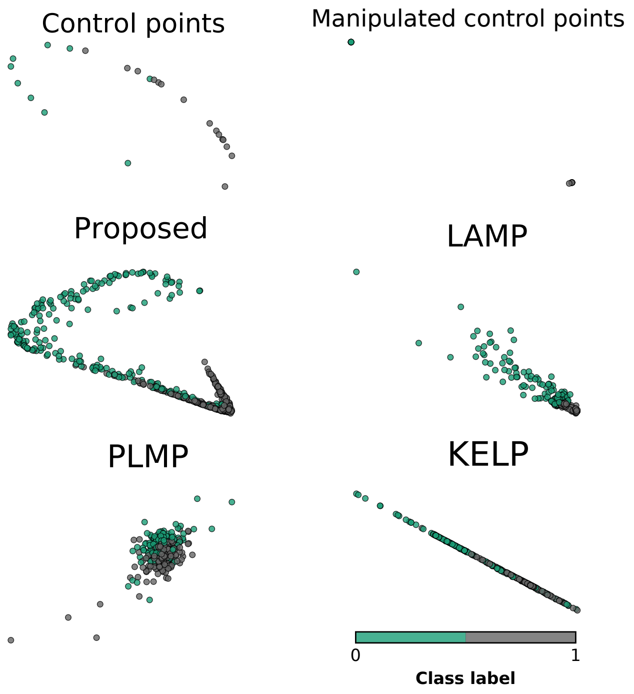

Fig. 2. Visualizing the Cancer dataset. (best viewed in color)

Table 4. Average Neighbor error after manipulation of control points. Best results marked bold. Standard deviation of 10 runs in parenthesis.

Data	Initialization	Neighbor Error				
		KELP	LAMP	PLMP	Proposed Linear	Proposed Kernel
Wine	PCA	20.77 (5.84)	26.24 (3.74)	27.73 (4.01)	24.63 (2.19)	**8.33 (4.87)**
	tSNE	18.56 (3.41)	27.75 (6.46)	27.86 (4.42)	25.92 (6.49)	**7.27 (2.34)**
	Force	19.03 (4.28)	26.36 (5.39)	25.34 (2.68)	25.81 (3.72)	**12.52 (5.60)**
Cncr	PCA	12.37 (3.56)	13.48 (1.48)	15.52 (2.30)	13.40 (2.90)	**5.23 (1.44)**
	tSNE	11.51 (2.39)	13.58 (2.31)	15.69 (2.47)	14.48 (2.15)	**8.44 (1.78)**
	Force	12.43 (2.66)	12.04 (1.66)	13.62 (0.84)	14.04 (2.23)	**7.49 (1.95)**
Segm.	PCA	8.19 (1.38)	7.00 (0.83)	9.11 (1.12)	8.83 (1.74)	**6.00 (0.83)**
	tSNE	8.52 (0.64)	**4.90 (0.65)**	8.02 (0.95)	7.51 (1.20)	5.84 (0.77)
	Force	7.69 (0.79)	**5.01 (0.90)**	7.83 (1.47)	7.27 (1.50)	5.57 (0.97)
MNIST	PCA	50.19 (2.72)	48.96 (3.20)	46.31 (3.23)	**44.50 (4.10)**	47.66 (2.88)
	tSNE	51.97 (3.20)	50.60 (1.63)	45.16 (4.94)	**45.10 (3.04)**	48.03 (2.40)
	Force	50.15 (3.76)	50.09 (3.34)	**46.57 (3.32)**	46.87 (2.55)	47.91 (2.54)

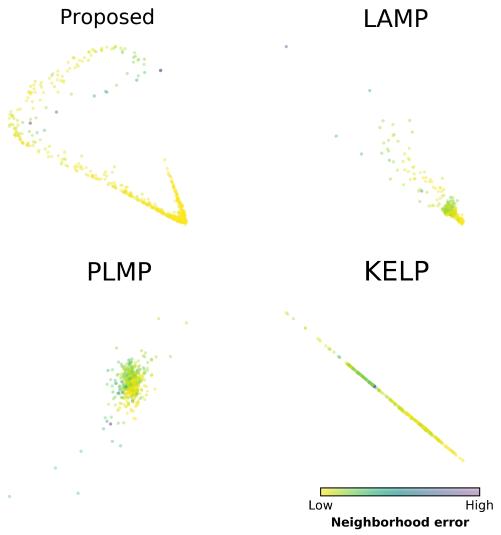

Fig. 3. Visualizing the neighbor error for the Cancer dataset. (best viewed in color)

Finally, we evaluate the effect of the number of used control points on the quality of the learned projection. The results are shown in Figure 4 using the proposed method and a linear projection function (PCA is used for the initialization). Even when a small number of control points is used, the proposed method outperforms the other methods. Qualitatively similar results were obtained using different initializations (e.g., tSNE or Force), as well as for other metrics and datasets.

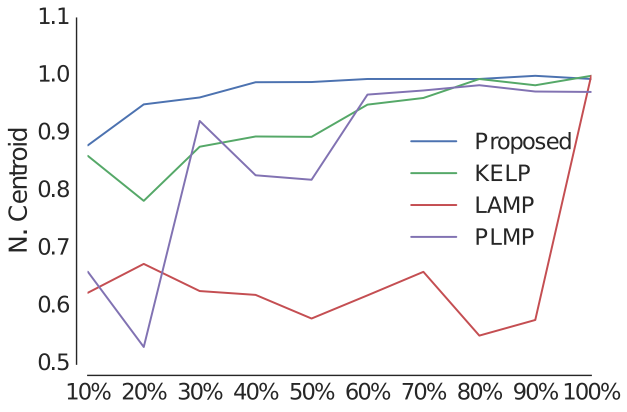

Fig. 4. Nearest Centroid precision in relation to control point increase (over the baseline proposed in the literature, i.e., $\sqrt{(n)}$) on the Wine dataset.

5 Conclusions

In this work, a novel interactive DR framework that is able to learn the optimal projection by exploiting the user interactions with the projected data was proposed. The proposed method can be combined with a number of different projection functions to readily adapt to the needs of each application, ranging from fast, lightweight linear projection functions to powerful deep neural networks. The user interaction was modeled as a target similarity matrix that was used to learn either a fast linear or a non-linear kernel projection using gradient descent. The proposed method was evaluated using a common interaction scenario in multidimensional projection literature, i.e., a subset of the data was projected, the data were rearranged in classes or clusters, and then a new projection function was learned based on that manipulation. The proposed method outperforms competitive baseline and state-of-the-art methods in the used benchmark datasets, while also being able to provide fast lightweight interactive projections. Namely, our methods improve the classification precision in Wine dataset by 16–29%, in Cancer by 1–31%, in Segmentation by 2–25% and in MNIST by 4–48%. Similarly, our methods improve the clustering coefficient in Wine dataset by 12–43, in Cancer by 5–51, in Segmentation by 11–30 and in MNIST by 6–53. There are several interesting future work directions. First, the target similarity matrix can be enriched with high-dimensional neighbor information using the rest of the dataset. Preliminary experiments using image and text datasets show promising results. It can be also used for interactive end-to-end-learning for manifold exploration [21, 22], domain adaptation [23] and transfer learning [24], as well as to interact with multimodal data [25, 26].

References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553) (2015) 436–444
2. Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T.: Power to the people: The role of humans in interactive machine learning. AI Magazine **35**(4) (2014) 105–120
3. Fails, J.A., Olsen Jr, D.R.: Interactive machine learning. In: Proceedings of the 8th international conference on Intelligent user interfaces, ACM (2003) 39–45
4. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(Nov) (2008) 2579–2605
5. Passalis, N., Tefas, A.: Dimensionality reduction using similarity-induced embeddings. IEEE transactions on neural networks and learning systems (99) (2017) 1–13
6. Derrien, T., André, C., Galibert, F., Hitte, C.: Autograph: an interactive web server for automating and visualizing comparative genome maps. Bioinformatics **23**(4) (2006) 498–499
7. Wang, Y., Zhang, B., Zhang, L., An, L., Xu, J., Li, D., Choudhary, M.N., Li, Y., Hu, M., Hardison, R., et al.: The 3d genome browser: a web-based browser for visualizing 3d genome organization and long-range chromatin interactions. BioRxiv (2017) 112268
8. Jolliffe, I.: Principal component analysis. Wiley Online Library (2002)

9. Kruskal, J.B., Wish, M.: Multidimensional scaling. Volume 11. (1978)
10. McInnes, L., Healy, J.: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv e-prints (February 2018)
11. Endert, A., Han, C., Maiti, D., House, L., North, C.: Observation-level interaction with statistical models for visual analytics. In: Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on, IEEE (2011) 121–130
12. Joia, P., Coimbra, D., Cuminato, J.A., Paulovich, F.V., Nonato, L.G.: Local affine multidimensional projection. IEEE Transactions on Visualization and Computer Graphics **17**(12) (2011) 2563–2571
13. Paulovich, F.V., Silva, C.T., Nonato, L.G.: Two-phase mapping for projecting massive data sets. IEEE Transactions on Visualization and Computer Graphics **16**(6) (2010) 1281–1290
14. Tejada, E., Minghim, R., Nonato, L.G.: On improved projection techniques to support visual exploration of multi-dimensional data sets. Information Visualization **2**(4) (2003) 218–231
15. Barbosa, A., Paulovich, F.V., Paiva, A., Goldenstein, S., Petronetto, F., Nonato, L.G.: Visualizing and interacting with kernelized data. IEEE transactions on visualization and computer graphics **22**(3) (2016) 1314–1325
16. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Drineas, P., Mahoney, M.W.: On the nyström method for approximating a gram matrix for improved kernel-based learning. Journal of Machine Learning Research **6**(Dec) (2005) 2153–2175
18. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
19. Lichman, M.: UCI machine learning repository (2013)
20. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20** (1987) 53–65
21. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research **7** (2006) 2399–2434
22. Zhang, Y., Zhang, Z., Qin, J., Zhang, L., Li, B., Li, F.: Semi-supervised local multi-manifold isomap by linear embedding for feature extraction. Pattern Recognition **76** (2018) 662–678
23. Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: Proceedings of the International Conference on Machine Learning. (2011) 513–520
24. Pan, S.J., Yang, Q., et al.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10) (2010) 1345–1359
25. Kim, J., Chung, H.J., Park, C.H., Park, W.Y., Kim, J.H.: Chromoviz: multimodal visualization of gene expression data onto chromosomes using scalable vector graphics. Bioinformatics **20**(7) (2004) 1191–1192
26. Liu, Y., Liu, L., Guo, Y., Lew, M.S.: Learning visual and textual representations for multimodal matching and classification. Pattern Recognition (2018)