Large-Scale Manifold Learning by Semidefinite Facial Reduction

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

The problem of nonlinear dimensionality reduction is often formulated as a semidefinite programming (SDP) problem. However, only SDP problems of limited size can be directly solved directly using current SDP solvers. To overcome this difficulty, we propose a novel SDP formulation for dimensionality reduction based on semidefinite facial reduction that significantly reduces the number of variables and constraints of the SDP problem, allowing us to solve very large manifold learning problems. Moreover, our reduction is exact, so we obtain high quality solutions without the need for post-processing by local gradient descent search methods, as is often required by other SDP-based methods for manifold learning.

Keywords: dimensionality reduction, unsupervised learning, semidefinite programming

1. Introduction

The problem of nonlinear dimensionality reduction has received a great deal of attention recently. A number of authors have formulated this problem as a semidefinite programming (SDP) problem, such as: Weinberger and Saul (2004); Weinberger et al. (2006); Song et al. (2007); Shaw and Jebara (2007, 2009). All of these methods preserve the local structure of the data by fixing the Euclidean distance between data points that are 'similar'. Using various different intuitions, these methods determine the distances between all the data points, while preserving distances between points that are close to each other. Maximum Variance Unfolding (MVU) (Weinberger and Saul, 2004) tries to flatten the nonlinear data manifold by pulling all points as far apart as possible, while fixing the distances between similar points. Colored MVU (Song et al., 2007) follows an approach similar to MVU, but incorporates side-information (such as class labels).

A significant problem with all of these algorithms is the high computational cost of solving SDP problems. Even after exploiting the sparsity of the constraint matrix, each iteration of a typical SDP solver takes $\mathcal{O}(n^3 + m^3)$ time and $\mathcal{O}(n^2 + m^2)$ space, where n is the size of the matrix variable and m is the number of constraints (Borchers and Young, 2007). On the other hand, SDP solvers usually converge in less than one hundred iterations, regardless of the problem size. In the MVU method, we have $m \in \mathcal{O}(n)$; as a result, an SDP solver requires $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space to solve this problem. In practice, solving such SDP problems is only efficient when n is at most a few hundred.

Due to this limitation in the size of SDP problems that can be solved efficiently, Fast-MVU (Weinberger et al., 2006) has been proposed. This is a scalable variation of MVU. However, FastMVU uses an approximation to reduce the size of the SDP, resulting in sub-optimal solutions that require post-processing by a local gradient descent search method.

In this paper, we propose a novel SDP formulation for nonlinear dimensionality reduction algorithm called 'semidefinite facial reduction unfolding' or SFU for short. SFU is stable, fast, scalable, and has no tuning parameters. Unlike existing large-scale variations of SDP-based methods, SFU does not need post-processing steps because the reduction we propose is *exact*.

The SFU method introduced in this paper, dramatically reduces the complexity of SDP problems by restricting the search space of the SDP problem to a small face of the semidefinite cone—this is not an approximation, meaning that very large SDP problems can be solved *exactly* and *efficiently* by this technique. Although we only show the details of this method when applied to the problem of dimensionality reduction (a reformulation of MVU), this technique is general and could be applied to many existing machine learning algorithms that are cast as an instance of SDP but where their effectiveness is limited by the computational complexity of SDP solvers. In addition, this 'semidefinite facial reduction' technique has been recently used with great success in the related problems of *protein structure determination* (see Alipanahi et al. (2012)) and *sensor network localization* (see Krislock and Wolkowicz (2010)). In this paper we show the theoretical validity of the proposed 'semidefinite facial reduction' procedure by presenting a new shorter proof of the variable reduction theorem from Krislock and Wolkowicz (2010) extend this theoretical result by establishing an important and practical constraint reduction theorem.

The rest of this paper is organized as follows. Sections 2 and 3 introduce the notation and background material, including MVU, and FastMVU. Section 4 develops our proposed method, which is conceptually divided into three major steps. Section 5 presents the experimental results of our algorithm and we draw our conclusions in Section 6.

2. Notation

Scalars, vectors, sets, and matrices are shown in small, small bold italic, script, and capital letters, respectively. We let $\mathbb{R}^p$ be the space of real p-dimensional vectors, $\mathbb{R}^{p\times q}$ be the space of real $p \times q$ matrices, $\mathcal{S}^p$ be the space of symmetric $p \times p$ matrices, $\mathcal{S}^p_+$ be the set of symmetric positive semidefinite $p \times p$ matrices, and $\mathcal{S}^p_{++}$ be the set of symmetric positive definite $p \times p$ matrices. Furthermore, we let I_p be the $p \times p$ identity (subscript omitted when dimension is clear), $\boldsymbol{e}_p$ be the $p \times 1$, all-ones vector (subscript omitted when dimension is clear), and $|\mathcal{B}|$ be the cardinality of a set $\mathcal{B}$. For a matrix $A \in \mathbb{R}^{p\times q}$ we let A_{ij} be the (i, j)-th entry of A, $A[\mathcal{B}, :]$ be the submatrix of A consisting of rows indexed by $\mathcal{B} \subseteq \{1, \ldots, p\}$ and all columns $\{1, \ldots, q\}$, $A^\top$ be the transpose of A, $\mathbf{rank}(A)$ be the rank of A, and $\mathbf{range}(A)$ be the range-space of A. For a square matrix $A \in \mathbb{R}^{p\times p}$, we let $A[\mathcal{B}]$ be the principal submatrix of A indexed by $\mathcal{B} \subseteq \{1, \ldots, p\}$, $\mathbf{trace}(A)$ be the sum of diagonal elements of A, and $\mathbf{diag}(A)$ be the vector made from diagonal elements of A. Finally, for a vector $v \in \mathbb{R}^p$, we let $\mathbf{Diag}(v)$ be the $p \times p$ diagonal matrix with the vector v along its diagonal.

3. Background

In order to motivate our algorithm, we provide a brief overview of some related methods. Many nonlinear dimensionality reduction techniques have been proposed in the last decade, including kernel PCA (Mika et al., 1999), locally linear embedding (LLE) (Roweis and Saul,

2000), Laplacian Eigenmaps (Belkin and Niyogi, 2001), and Isomap (Tenenbaum, 1998). It has been shown that all of these algorithms can be formulated as kernel PCA (Ham et al., 2004). The difference lies mainly in the choice of the kernel. Common kernels such as radial basis functions and polynomial kernels generally perform poorly at manifold learning, which is perhaps what motivated the development of algorithms such as LLE and Isomap.

The problem of choosing an appropriate kernel remained a crucial problem until more recently, when Weinberger and Saul (2004) introduced MVU. The MVU method learns a kernel matrix K instead of choosing a kernel function *a priori*. It formulates the problem of learning a kernel K from the data as an SDP problem. A low-dimensional embedding of the data can then be determined by running kernel PCA on the learned kernel K.

The connection between a kernel matrix K and the distances between the data points is as follows. Let $D \in \mathcal{S}^n$ be the Euclidean distance matrix (EDM) of the data points and $K \in \mathcal{S}^n_+$ the kernel matrix (or Gram matrix) of the same set of points; that is, given a collection of data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, we have

$$D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \qquad \text{and} \qquad K_{ij} = \mathbf{x}_i^\top \mathbf{x}_j, \qquad \text{for all } i, j = 1, \ldots, n.$$

Using the linear operator $\mathcal{K}(\cdot)$ defined by

$$\mathcal{K}(K)_{ij} = K_{ii} + K_{jj} - 2K_{ij}, \qquad \text{for all } i, j = 1, \ldots, n, \tag{1}$$

we have $D = \mathcal{K}(K)$. Moreover, given an EDM $D \in \mathcal{S}^n$, we can determine the unique kernel matrix $K \in \mathcal{S}^n_+$ of a set of centered points whose EDM is D using the $\mathcal{T}(\cdot)$ operator,

$$K = \mathcal{T}(D) = -\frac{1}{2} HDH, \quad D \in \mathcal{S}^n, \tag{2}$$

where $H = I - \frac{1}{n}\boldsymbol{e}\boldsymbol{e}^\top$. It can be shown that a matrix D with all zeros on its diagonal is an EDM if and only if $K = \mathcal{T}(D)$ is positive semidefinite (Schoenberg, 1935; Cox and Cox, 2001, p. 397).

In MVU, in addition to the semidefinite constraint, neighborhood constraints are used to preserve the local distances of neighboring points $i \sim j$ in a k-nearest neighbor graph. By adding an objective function to maximize $\mathbf{trace}(K)$, which ensures maximal variance in the resulting embedding, MVU constructs an SDP problem for learning the kernel matrix K. In its last step, MVU performs an eigendecomposition on the kernel K to extract the coordinates of the low-dimensional points in the embedded space. This algorithm is summarized in Algorithm 1.

FastMVU reduces the size of the SDP problem by decomposing the kernel matrix K as $K \approx QYQ^\top$, where $Q \in \mathbb{R}^{n \times v}$ is computed from the v nontrivial bottom eigenvectors of the Laplacian matrix of the k-nearest neighbor graph (Weinberger et al., 2006). However, since the decomposition $K \approx QYQ^\top$ is only approximate, it is not possible to preserve the local distance constraints exactly, so FastMVU adopts a least-squares approach to satisfy the local distance constraints. FastMVU is summarized in Algorithm 2. The objective function in this optimization problem is quadratic, so problem (4) is a quadratic SDP, or QSDP. FastMVU uses the well-known Schur complement technique to obtain an equivalent SDP problem with a linear objective function. As a result, although the size of the Y matrix in the QSDP in problem (4) is only $v \times v$, the size of the semidefinite variable in the equivalent

Alg. 1 Maximum Variance Unfolding

1: Construct the neighborhood graph, using k-nearest neighbors.
2: Solve the SDP problem,

$$\begin{aligned} \textbf{maximize} \quad & \textbf{trace}(K) \\ \textbf{subject to} \quad & K_{ii} + K_{jj} - 2K_{ij} = D_{ij}, \text{ if } i \sim j \\ & \textstyle\sum_{ij} K_{ij} = 0 \\ & K \succeq 0 \end{aligned} \tag{3}$$

3: Run Kernel PCA on the learned kernel, K.

Alg. 2 Fast Maximum Variance Unfolding

1: Construct the neighborhood graph, using k-nearest neighbors.
2: Compute graph Laplacian matrix and find its v nontrivial bottom eigenvectors $Q \in \mathbb{R}^{n \times v}$.
3: Solve the following SDP problem

$$\begin{aligned} \textbf{maximize} \quad & \textbf{trace}(Y) - \omega \textstyle\sum_{i \sim j} [(QYQ^\top)_{ii} + (QYQ^\top)_{jj} - 2(QYQ^\top)_{ij} - D_{ij}]^2 \\ \textbf{subject to} \quad & Y \succeq 0 \end{aligned} \tag{4}$$

4: Run Kernel PCA on the learned kernel, $K = QYQ^\top$.
5: Perform gradient descent post-processing on the embedded points.

linear SDP problem is $v^2 + v + 1$. Solving this SDP problem has complexity $\mathcal{O}(v^6)$ and is inefficient to solve when $v \geq 30$. Since the optimal K is only an approximate solution, gradient descent post-processing is performed on the embedded points in order to improve the quality of the embedding.

An improved version of FastMVU was formulated by Wu et al. (2009). Using a well-known technique (see, e.g., (Ben-Tal and Nemirovski, 2001, Lecture 1?)) Wu et al. (2009) observed that the QSDP in problem (4) is in fact equivalent to a semidefinite-quadratic-linear programming (SQLP) problem having a linear objective function, linear equality constraints, a quadratic (or second-order cone) constraint, and a semidefinite constraint. Furthermore, Wu et al. (2009) demonstrate that the SQLP formulation of problem (4) can be solved much more efficiently than the QSDP formulation used by Weinberger et al. (2006).

4. Large-scale manifold learning by semidefinite facial reduction (SFU)

We follow an intuition similar to MVU—we preserve the local structure of a manifold when maximizing the total variance in the embedding space. However, our proposed method differs from the FastMVU approach in two key areas.

The first key point of our proposed method is that the structure of a reasonably large cluster of the data should be preserved as a whole. This idea is motivated by the following:

1. the data lies on a low-dimensional nonlinear manifold;

2. manifolds are locally approximately linear.

Therefore, we propose using what we call the *cluster graph*, $G = (V, E)$, where the set of vertices $V = \{1, \ldots, n\}$ is partitioned into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_q$ such that the data in each cluster is approximately linear. Points within clusters are all connected to each other, and additional edges are added to the edge set E that connect neighbouring clusters.

The next key step in our proposed method is, given a clustering of the data, we prove that an exact decomposition $K = UZU^\top$ is possible and give an explicit expression for U; note that U is a known matrix and Z is the unknown matrix variable. Since the decomposition of K is exact, the final result of the algorithm is optimal for the original SDP problem and thus does not require a post-processing step to improve the quality of the solution. This exact decomposition also implies that a least-squares approach, as is used in FastMVU, is not necessary in our method.

4.1. Overview of the SFU method

In this section, we will first give the intuition and a sketch of the proposed algorithm; secondly, we will give the formal justification and describe the algorithm in detail. Suppose a given data set has been divided into clusters. SFU takes an approach similar to MVU but treats each cluster as a single point. That is, we construct a neighborhood graph between clusters, and preserve only the distances of the neighboring clusters while pulling the non-neighbor clusters as far apart as possible.

More formally, suppose a given data set is divided into q clusters $\{\mathcal{C}_\ell\}_{\ell=1}^q$ of size $n_\ell = |\mathcal{C}_\ell| > d$, where d is the dimension of the embedding. Let $D_\ell \in \mathcal{S}^{n_\ell}$ denote the EDM of the points in $\mathcal{C}_\ell$. Clearly the EDM of all data points $\{\boldsymbol{x}_i\}_{i=1}^n$ can be represented as

$$D = \begin{bmatrix} D_1 & . & \cdots & . \\ . & D_2 & \cdots & . \\ \vdots & \vdots & \ddots & \vdots \\ . & . & \cdots & D_q \end{bmatrix} \in \mathcal{S}^n \ , \tag{5}$$

where only the block-diagonal elements are known. The goal is to determine the non-block-diagonal elements of D. This problem can be cast as an SDP problem.

Given clusters of a given data set, SFU has three major steps: (I) form the cluster graph $G = (V, E)$ from the clusters $\{\mathcal{C}_\ell\}_{\ell=1}^q$; (II) decompose the kernel matrix K as $K = UZU^\top$, where the matrix U is computed from clusters $\{\mathcal{C}_\ell\}_{\ell=1}^q$; (III) the positive semidefinite matrix Z is computed by solving a reduced SDP problem. A sketch of this algorithm is given in Algorithm 3; the algorithm will be given in detail in Algorithm 4 after discussing the formal justification.

4.2. Determine clusters and form the cluster graph (Step I)

Most prominent dimensionality reduction methods assume that a sensible neighborhood graph is given as input for learning. They usually compute the neighborhood graph by finding the k-nearest neighbors of each point. The number of neighbors k is also an input to these algorithms. This is in contrast with our proposed SFU method that requires a partitioning of the given data as the input.

Alg. 3 Sketch of the proposed method

1: Determine the clusters $\{\mathcal{C}_\ell\}_{\ell=1}^q$ and form the cluster graph $G = (V, E)$.
2: Compute the matrix U from the clusters $\{\mathcal{C}_\ell\}_{\ell=1}^q$.
3: Determine Z by maximizing $\mathbf{trace}(UZU^\top)$ subject to distance constraints for preserving structure of the clusters and constraints for preserving distances between neighboring clusters.
4: Run Kernel PCA on the learned kernel $K = UZU^\top$.

In this paper, and in all of the experiments, we used affinity propagation clustering (APC) (Frey and Dueck, 2007) to cluster a given data set. The input to APC is a set of similarities between data points, which is usually the negative of the Euclidean distance between them, and a 'preference' value. APC does not require the user to predetermine the number of clusters. Instead the number of clusters will be implicitly determined by the preference value. In all of the experiments of this paper, we set the preference value as the median of similarities, removing the need for tuning the clustering parameters. The run time of APC is $\mathcal{O}(n^2)$.

After clustering, we define a cluster as a set of indices, $\mathcal{C}_\ell$, such that $\{\boldsymbol{x}_i\}_{i\in\mathcal{C}_\ell}$ are the points in the ℓ-th cluster. We assume that there are q clusters $\{\mathcal{C}_\ell\}_{\ell=1}^q$ of size $n_\ell = |\mathcal{C}_\ell| > d$, such that $n = \sum_\ell n_\ell$, that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_1}$ denote the points of the first cluster, $\boldsymbol{x}_{n_1+1}, \ldots, \boldsymbol{x}_{n_2}$ denote the points of the second cluster, and so on. We also assume that each cluster is nearly on an affine space with dimensionality close to the intrinsic dimensionality of the manifold. Note that it is always possible to provide such clusters for a given data set if the size of the clusters are small enough.

4.2.1. Finding the neighbors in the cluster graph

In order to preserve the local structure of each cluster, we need to fix the distances between each pair of points in the same cluster. The corresponding (i, j) pairs will be put in the constraint set E_W, or the set of "*within*" cluster constraints. That is, we have

$$E_W = \{(i, j) : i, j \in \mathcal{C}_\ell, \text{for some } \ell = 1, \ldots, q\}.$$

Next, to keep the neighboring clusters close to each other, we link the clusters together, by first finding the extreme points of the *convex hull* of each cluster. Then, for each extreme point, we find its closest neighbor among all the extreme points of the convex hulls of the other clusters. We then say that two extreme points from different clusters are neighbors if they are mutually closest to each other. We put all such cluster linking edges in the constraint set E_B, or the set of "*between*" cluster constraints.

4.2.2. Unfolding the cluster graph

Now that we have constructed the cluster graph $G = (V, E)$, with $V = \{1, \ldots, n\}$ and $E = E_W \cup E_B$, we get the following (unreduced) SDP problem whose solution will give us

the maximum variance unfolding of the cluster graph:

$$\begin{aligned} \textbf{maximize} \quad & \textbf{trace}(K) \\ \textbf{subject to} \quad & K_{ii} + K_{jj} - 2K_{ij} = D_{ij}, \quad (i,j) \in E_W, \\ & K_{ii} + K_{jj} - 2K_{ij} \leq D_{ij}, \quad (i,j) \in E_B, \\ & \textstyle\sum_{ij} K_{ij} = 0, \\ & K \succeq 0. \end{aligned} \tag{6}$$

Note that we allow the cluster linking distances to become shorter when unfolding the manifold since we need to prevent the graph from "locking-up" as we unfold it.

We could solve the SDP problem (6) directly, but we are limited in the size of problems we can solve. For example, if we have a data set with $15,000$ points that has been grouped into 100 clusters, each of size 150, and there are 500 cluster links, then the resulting SDP problem would have a matrix variable of size $n = 15,000$ and $|E| = |E_W| + |E_B| = 100\binom{150}{2} + 200 = 1,118,000$ constraints—this is far beyond the limit of current SDP solvers. However, as we show next, we are able to significantly reduce the size of the manifold unfolding SDP problem (6) and solve many such large scale problems, as we will show in the computational results in Section 5.

4.3. Decomposing the matrix K (Step II)

We now provide the details on the how we can obtain an SDP problem that is *equivalent* to problem (6), but with much fewer variables and constraints. In particular, we show that if K is a kernel matrix that satisfies all the constraints in the SDP problem (6), then $K \in U\mathcal{S}_+^{q(d+1)}U^\top$; that is, $K = UZU^\top$ for some $Z \in \mathcal{S}_+^{q(d+1)}$. This technique is known as *semidefinite facial reduction* because the set of matrices $U\mathcal{S}_+^{q(d+1)}U^\top$ is geometrically a *face* of the semidefinite matrix cone $\mathcal{S}_+^n$; for more information, see (Borwein and Wolkowicz, 1981; Ramana et al., 1997).

4.3.1. Semidefinite facial reduction for manifold unfolding

We begin by stating a known result that gives us the needed reduction in the number of variables. Although Theorem 1 is known, in this paper we contribute a new shorter proof of the main part of this theorem.

Theorem 1 ((Krislock and Wolkowicz, 2010, Theorem 2.3)) *Let $D \in \mathcal{S}^p$ be a Euclidean distance matrix with embedding dimension d and let $\mathcal{F}$ be the set of $n \times n$ centered Gram matrices K whose first p points agree with the distances given in D; that is,*

$$\mathcal{F} = \left\{ K \in \mathcal{S}_+^n \cap \mathcal{S}_C^n : K_{ii} + K_{jj} - 2K_{ij} = D_{ij}, \forall i,j = 1,\ldots,p \right\}.$$

Let G be the centered Gram matrix corresponding to D; that is, $G = \mathcal{T}(D)$, where $\mathcal{T}(\cdot)$ is defined in equation (2). Let $P \in \mathbb{R}^{d\times p}$ be centered points whose Gram matrix is G; that is, $P\boldsymbol{e} = 0$ and $G = P^\top P$. Let $\bar{U} \in \mathbb{R}^{p\times(d+1)}$ be a matrix with orthonormal columns such that $\textbf{range}(\bar{U}) = \textbf{range}(\begin{bmatrix} P^\top & \boldsymbol{e} \end{bmatrix})$ *and let* $U = \begin{bmatrix} \bar{U} & 0 \\ 0 & I_{n-p} \end{bmatrix} \in \mathbb{R}^{n\times(n-p+d+1)}$. *Let* $\begin{bmatrix} V & \frac{U^\top \boldsymbol{e}}{\|U^\top \boldsymbol{e}\|} \end{bmatrix}$ *be orthogonal. Then:*

1. *for each* $K \in \mathcal{F}$, *there exists* $Z \in \mathcal{S}_+^{n-p+d+1}$ *such that* $K = UZU^\top$;

2. *for each* $K \in \mathcal{F}$, *there exists* $\bar{Z} \in \mathcal{S}_+^{n-p+d}$ *such that* $K = (UV)\bar{Z}(UV)^\top$;

3. *there exists* $\bar{Z} \in \mathcal{S}_{++}^{n-p+d}$ *such that* $(UV)\bar{Z}(UV)^\top \in \mathcal{F}$.

Proof We now give a new proof of item 1. For this proof, we need to define some notation. The inner-product of matrices $A, B \in \mathbb{R}^{m\times n}$ is defined as $\langle A, B\rangle = \sum_{i=1}^m \sum_{j=1}^n A_{ij}B_{ij} = \mathbf{trace}(A^\top B)$. We extend the definition of the linear map $\mathcal{T}$ to accept matrices without a zero diagonal as follows:

$$\mathcal{T}(B) = -\frac{1}{2}H\mathbf{offDiag}(B)H \ ;$$

here we have $\mathbf{offDiag}(B) = B - \mathbf{Diag}(\mathbf{diag}(B))$, for $B \in \mathcal{S}^n$. The adjoint of $\mathcal{T}$ is the linear map $\mathcal{T}^*$ which satisfies, $\langle \mathcal{T}^*(A), B\rangle = \langle A, \mathcal{T}(B)\rangle$, for all $A, B \in \mathcal{S}^n$, and is given by $\mathcal{T}^*(A) = -\frac{1}{2}\mathbf{offDiag}(HAH)$. Similarly, the adjoint of $\mathcal{K}$, defined in equation (1), is given by $\mathcal{K}^*(A) = 2(\mathbf{Diag}(A\boldsymbol{e}) - A)$.

We begin the proof by letting $\bar{V}$ such that $\begin{bmatrix}\bar{U} & \bar{V}\end{bmatrix}$ is orthogonal. Then, by the definition of $\bar{U}$, we have that $\bar{V}^\top P^\top = 0$ and $\bar{V}^\top \boldsymbol{e} = 0$. Next we let $\Lambda = \mathcal{T}^*(\bar{V}\bar{V}^\top)$. Since $\bar{V}^\top \boldsymbol{e} = 0$, we have that $H\bar{V}\bar{V}^\top H = \bar{V}\bar{V}^\top$. Thus, $\Lambda = -\frac{1}{2}\mathbf{offDiag}(\bar{V}\bar{V}^\top)$ and

$$\mathcal{K}^*(\Lambda) = 2(\mathbf{Diag}(\Lambda\boldsymbol{e}) - \Lambda) = \mathbf{offDiag}(\bar{V}\bar{V}^\top) - \mathbf{Diag}(\mathbf{offDiag}(\bar{V}\bar{V}^\top)\boldsymbol{e}) = \bar{V}\bar{V}^\top,$$

since $\bar{V}\bar{V}^T\boldsymbol{e} = 0$. Let $K \in \mathcal{F}$. Then

$$\left\langle K, \begin{bmatrix}\bar{V}\bar{V}^\top & 0\\ 0 & 0\end{bmatrix}\right\rangle = \left\langle K[1{:}p], \bar{V}\bar{V}^\top\right\rangle = \langle K[1{:}p], \mathcal{K}^*(\Lambda)\rangle = \langle \mathcal{K}(K[1{:}p]), \Lambda\rangle$$
$$= \left\langle D, \mathcal{T}^*(\bar{V}\bar{V}^\top)\right\rangle = \left\langle \mathcal{T}(D), \bar{V}\bar{V}^\top\right\rangle = \left\langle G, \bar{V}\bar{V}^\top\right\rangle = \left\langle P^\top P, \bar{V}\bar{V}^\top\right\rangle = 0,$$

which implies that $K\begin{bmatrix}\bar{V}\\ 0\end{bmatrix} = 0$. Therefore, all the eigenvectors of K corresponding to its nonzero eigenvalues must be linear combinations of the columns of the matrix U. Therefore, $K = UZU^\top$ for some $Z \in \mathcal{S}_+^{n-p+d+1}$, which proves item 1. See Krislock and Wolkowicz (2010) for the proofs of item 2 and item 3. ■

We now extend this result to include a reduction in the number of distance constraints.

Theorem 2 *Let* $D \in \mathcal{S}^p$ *be an EDM with embedding dimension* d *and let* $\mathcal{F}$ *be the set of* $n \times n$ *centered Gram matrices* K *whose first* p *points agree with the distances given in* D; *that is,*

$$\mathcal{F} = \left\{K \in \mathcal{S}_+^n \cap \mathcal{S}_C^n : K_{ii} + K_{jj} - 2K_{ij} = D_{ij}, \forall i, j = 1, \ldots, p\right\}.$$

Let $U \in \mathbb{R}^{n\times(n-p+d+1)}$ *be defined as in Theorem 1. Let* $\mathcal{B} \subseteq \{1, \ldots, p\}$ *for which the principal submatrix* $D[\mathcal{B}]$ *has embedding dimension* d. *Then*

$$\mathcal{F} = \left\{K \in \left(U\mathcal{S}_+^{n-p+d+1}U^\top\right) \cap \mathcal{S}_C^n : K_{ii} + K_{jj} - 2K_{ij} = D_{ij}, \forall i, j \in \mathcal{B}\right\}. \tag{7}$$

Proof Let $K \in \mathcal{F}$. Then by Theorem 1, $K \in \left(U\mathcal{S}_+^{n-p+d+1}U^\top\right) \cap \mathcal{S}_C^n$. Moreover, since $K \in \mathcal{F}$ and $\mathcal{B} \subseteq \{1,\ldots,p\}$, we clearly have that $K_{ii} + K_{jj} - 2K_{ij} = D_{ij}$, for all $i,j \in \mathcal{B}$.

We now prove the other direction. Let $K \in \left(U\mathcal{S}_+^{n-p+d+1}U^\top\right) \cap \mathcal{S}_C^n$ such that

$$K_{ii} + K_{jj} - 2K_{ij} = D_{ij}, \quad \text{for all } i,j \in \mathcal{B}\ . \tag{8}$$

Then there exists $Z \in \mathcal{S}_+^{n-p+d+1}$ such that

$$K = UZU^\top = \begin{bmatrix} \bar{U} & 0 \\ 0 & I \end{bmatrix} Z \begin{bmatrix} \bar{U}^\top & 0 \\ 0 & I \end{bmatrix},$$

so $K[1{:}p] = \bar{U}\bar{Z}\bar{U}^T$, for some $\bar{Z} \in \mathcal{S}_+^{d+1}$. Let $\bar{K} = HK[1{:}p]H = (H\bar{U})\bar{Z}(H\bar{U})^\top$. Then $\bar{K}$ is a centered Gram matrix that satisfies

$$\bar{K}_{ii} + \bar{K}_{jj} - 2\bar{K}_{ij} = K_{ii} + K_{jj} - 2K_{ij}, \quad \text{for all } i,j = 1,\ldots,p. \tag{9}$$

Note that

$$H\bar{U} = \left(I - \frac{1}{p}\boldsymbol{e}\boldsymbol{e}^\top\right)\bar{U} = \bar{U} - \frac{1}{p}\boldsymbol{e}\left(\boldsymbol{e}^\top\bar{U}\right)\ .$$

Thus, since $\boldsymbol{e} \in \mathbf{range}(\bar{U})$, we have that $\mathbf{range}(H\bar{U}) \subseteq \mathbf{range}(\bar{U})$. Therefore,

$$\mathbf{range}(\bar{K}) \subseteq \mathbf{range}(H\bar{U}) \subseteq \mathbf{range}(\bar{U})\ ,$$

so we have that $\bar{K} \in \bar{U}\mathcal{S}_+^{d+1}\bar{U}^\top$. Now let $\bar{V}$ be a $(d+1)\times d$ matrix that satisfies

$$\mathbf{range}(\bar{V}) = \left\{\bar{U}^T\boldsymbol{e}\right\}^\perp \tag{10}$$

Then $\bar{K} = (\bar{U}\bar{V})\Phi(\bar{U}\bar{V})^\top$, for some $\Phi \in \mathcal{S}_+^d$. Next we note that (9) and (10) imply that

$$\bar{K}_{ii} + \bar{K}_{jj} - 2\bar{K}_{ij} = D_{ij}, \quad \text{for all } i,j \in \mathcal{B}\ . \tag{11}$$

Moreover, (12) tells us that $\bar{K}[\mathcal{B}]$ is a Gram matrix for the EDM $D[\mathcal{B}]$. Therefore, $H\bar{K}[\mathcal{B}]H$ is the centered Gram matrix for $D[\mathcal{B}]$; that is, $H\bar{K}[\mathcal{B}]H = \mathcal{T}(D[\mathcal{B}])$. Since $\bar{K}[\mathcal{B}] = (\bar{U}_\mathcal{B}\bar{V})\Phi(\bar{U}_\mathcal{B}\bar{V})^\top$, where $\bar{U}_\mathcal{B} := \bar{U}[\mathcal{B},:]$, we have that

$$(H\bar{U}_\mathcal{B}\bar{V})\Phi(H\bar{U}_\mathcal{B}\bar{V})^\top = \mathcal{T}(D[\mathcal{B}])\ . \tag{12}$$

Since $D[\mathcal{B}]$ has embedding dimension d, we have that $\mathbf{rank}(\mathcal{T}(D[\mathcal{B}])) = d$; therefore, $\mathbf{rank}(H\bar{U}_\mathcal{B}\bar{V}) \geq d$. Notice that the matrix $H\bar{U}_\mathcal{B}\bar{V}$ is $|\mathcal{B}| \times d$; therefore, $H\bar{U}_\mathcal{B}\bar{V}$ has full-column rank. Thus, Φ is the unique solution of equation (13), implying that there is a unique centered Gram matrix $\bar{K} \in \bar{U}\mathcal{S}_+^{d+1}\bar{U}^\top$ that satisfies (12). Therefore, $\bar{K} = G$, where $G = \mathcal{T}(D)$. Since

$$G_{ii} + G_{jj} - 2G_{ij} = D_{ij}, \quad \text{for all } i,j = 1,\ldots,p\ , \tag{13}$$

we have by equation (10) that $K \in \mathcal{F}$. ■

Theorem 1 implies that we can replace a large $n \times n$ semidefinite variable K with $UZU^\top$, where Z is a much smaller $(n-p+d+1) \times (n-p+d+1)$ semidefinite variable. In addition, Theorem 2 implies we also get a drastic reduction in the number of constraints—in fact, we only need distance constraints between $|\mathcal{B}| = d+1$ noncoplanar points. This amounts to a major reduction in both variables and constraints. Most importantly, this reduction is exact and loses no information.

We now provide the intuition behind the reduction in the number of distance constraints. Suppose you have p points in $\mathbb{R}^d$ with known distances between all of them. By fixing $d+1$ points which are noncoplanar, the positions of all the other points are now uniquely determined. The matrix U encodes enough information that specifying the distances between just $d+1$ non-coplanar points is enough to fully describe all possible Gram matrices corresponding to the EDM D.

For each cluster $\mathcal{C}_\ell$, we compute the matrix U_ℓ as in Theorem 1. After computing the matrices U_ℓ, for $\ell = 1, \ldots, q$, we form the block-diagonal matrix

$$U = \begin{bmatrix} U_1 & 0 & \ldots & 0 \\ 0 & U_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & U_q \end{bmatrix} \in \mathbb{R}^{n \times q(d+1)} \,. \tag{14}$$

Since we can choose U_ℓ such that $U_\ell^\top U_\ell = I$, for $\ell = 1, \ldots, q$, we can also have $U^\top U = I$. It then follows from the disjoint cluster result (Krislock and Wolkowicz, 2010, Corollary 2.6), and from Theorem 1 and Theorem 2, that $K \in \mathcal{S}_+^n$ is a Gram matrix for the partial EDM D in (5) if and only if

$$K = UZU^\top, \quad \text{for some } Z \in \mathcal{S}_+^{q(d+1)} \,, \tag{15}$$

and K satisfies the distances between at least $d+1$ noncoplanar points in each cluster.

Recall now the example where we have a data set with $n = 15,000$ points that have been grouped into $q = 100$ clusters, each of size 150, and an additional $|E_B| = 500$ cluster links. Suppose the embedding dimension of our data is $d = 2$. Then our reduced SDP problem would have a matrix variable of size $q(d+1) = 300$ and the number of constraints would be $q\binom{d+1}{2} + |E_B| = 800$. This is in contrast to the original SDP problem which had a matrix variable of size $15,000$ and $1,118,000$ constraints.

4.4. Solving the SDP (Step III)

In this section, we present the reduced formulation the SDP problem (6) that unfolds the manifold. From the results in the previous section, we know that we have the exact decomposition of any feasible kernel matrix as $K = UZU^\top$, where U is given by equation (15). Since $U^\top U = I$, we have that $\mathbf{trace}(K) = \mathbf{trace}(Z)$. Therefore, in order to stretch the manifold, we maximize $\mathbf{trace}(Z)$.

For the constraint reduction, recall that we originally had

$$E_W = \{(i,j) : i,j \in \mathcal{C}_\ell, \text{for some } \ell = 1, \ldots, q\} \,.$$

Alg. 4 Semidefinite Facial reduction Unfolding (SFU)

1: Determine the clusters using the APC method.
2: Determine the neighborhood relation between clusters.
3: Compute the matrix U in equation (15) for the decomposition, $K = UZU^\top$.
4: Solve the SDP problem in equation (17).
5: Run Kernel PCA on the learned kernel, $K = UZU^\top$.

However, after making the substitution $K = UZU^\top$, Theorem 2 implies that only $\binom{d+1}{2}$ constraints are required for each cluster. Suppose we choose the distance constraints between the nodes in $\mathcal{B}_\ell$ for cluster $\mathcal{C}_\ell$, for $\ell = 1, \ldots, q$. Then we only need to have the distance constraints in the set

$$\bar{E}_W = \{(i,j) : i,j \in \mathcal{B}_\ell, \text{for some } \ell = 1, \ldots, q\} .$$

Adding these distance constraints, the SDP problem (6) is equivalent to the reduced SDP problem

$$\begin{aligned} \textbf{maximize} \quad & \textbf{trace}(Z) \\ \textbf{subject to} \quad & (UZU^\top)_{ii} + (UZU^\top)_{jj} - 2(UZU^\top)_{ij} = D_{ij} , \quad (i,j) \in \bar{E}_W , \\ & (UZU^\top)_{ii} + (UZU^\top)_{jj} - 2(UZU^\top)_{ij} \leq D_{ij} , \quad (i,j) \in E_B , \\ & \textstyle\sum_{ij} (UZU^\top)_{ij} = 0 , \\ & Z \succeq 0 . \end{aligned} \tag{16}$$

After the SDP problem (17) has been solved, we need to compute the eigendecomposition of K to find the coordinates of the points. However, since $K = UZU^\top$, we only need to solve for the eigenvectors V_Z of Z, and then compute the eigenvectors of K as $V_K = UV_Z$. Since Z is much smaller than K, this last step will also be much faster.

The overall number of constraints is $m = q(a+(d+1)d/2)$; where a, the average number of neighbors per cluster, is $\mathcal{O}(1)$, and is usually around $5 \sim 10$. Moreover, the embedding dimensionality, d, is usually small and fixed, so $m \in \mathcal{O}(q)$. This is in contrast with the other SDP-based methods (e.g., MVU and MVE) where $m \in \mathcal{O}(n)$.

In terms of computational complexity, size of the reduced matrix Z is $q(d+1) \times q(d+1)$; considering that number of constraints is $m \in \mathcal{O}(q)$, the overall complexity of solving the SDP problem boils down to $\mathcal{O}(q^3)$. In our experiments we found $q \in \mathcal{O}(\sqrt{n})$, giving rise to a significant reduction in the problem size (see Table 1).

5. Experiments

In order to evaluate the performance of SFU and compare to other methods, we have conducted several experiments on real and synthetic data sets. All of these experiments except one were run on a 2.8 GHz quad-core Windows computer with 8 GB of memory. SFU is implemented in Matlab. We used SDPT3 Tütüncü et al. (2003); Toh et al. (2006) as the SDP solver in all experiments. All of the SDP run times are listed in Table 1.

Data Set	n	SFU run time	SFU q	FastMVU $v = 10(111)$	FastMVU $v = 20(421)$	FastMVU $v = 30(931)$
Swiss Roll	15,000	18	82 (246)	47	442	8497
World Map	15,040	11	84 (252)	128	351	-
Frey Faces	1,965	5	70 (210)	59	489	-

Table 1: Run times of SFU and FastMVU for different data sets in seconds. The numbers in parenthesis show the actual size of the SDP matrix: $q(d+1)$ for SFU (where q is the number of clusters), and v^2+v+1 for FastMVU (where v is the number of Laplacian eigenvectors).

5.1. Synthetic Data Sets

5.1.1. Swiss Roll

We first consider the famous Swiss Roll data set, depicted in Fig. 1. Although it only has three dimensions, it tends to be one of the more challenging data sets due to its complex global structure. We applied SFU to a randomly generated Swiss roll data set of size 15,000 and used APC to cluster the data, which found 82 clusters ($q = 82$). The size of the reduced matrix, Z, is less than *two percent* of the original matrix K. We also applied FastMVU to this data set with ten eigenvectors of the graph Laplacian matrix ($v = 10$), on the same data set. The unfolded manifolds are depicted in Fig. 1; it can be seen that the SFU produces a conformal embedding.

It should be noted that increasing the number of Laplacian eigenvectors (v) in FastMVU does not affect the results noticeably, while the run time grows dramatically. In fact, $v = 10$ is mentioned as the "sweet spot" of FastMVU by Weinberger et al. (2006). We also tried different values of v, however increasing v does not noticeably affect the output results. It is also interesting to note that for $v = 30$, the run times are a couple of hours.

5.2. Real Data Sets

5.2.1. Map of Cities

The first real data set we have considered was the map of randomly selected cities from Europe, Asia, and Africa. We downloaded the "Worldcities" data set[1], which includes the lattitude and longitude of nearly 2.7 million cities. We selected the cities in Europe, Asia, and Africa; randomly sampled 15,040 cities and mapped their spherical coordinates to the three dimensional Cartesian coordinates.

We applied SFU to the three dimensional data set, APC found 84 clusters and SDP was solved in 11 seconds. FastMVU was also applied to the same data set with $k = 30$ and $v = 10$. FastMVU finished in 128 seconds. The two dimensional embeddings are depicted in Figure 2. SFU produces a better result, which is loyal to the original map. But the two-dimensional map generated by FastMVU is compressed.

1. Available from http://geolite.maxmind.com/download/worldcities/cities.txt.gz.

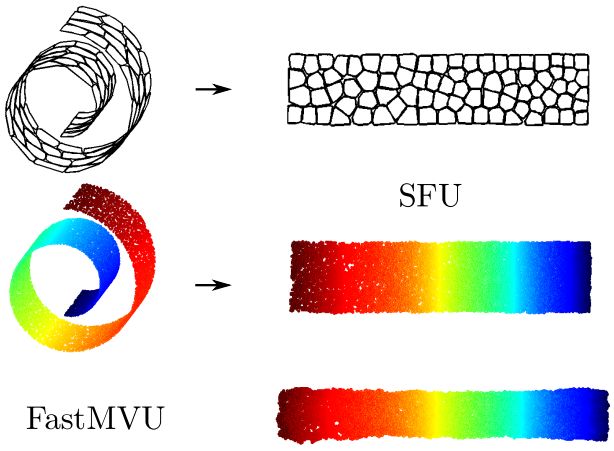

Figure 1: Left: manifold divided into small clusters, right: two-dimensional embedding. Bottom row: embedding of FastMVU.

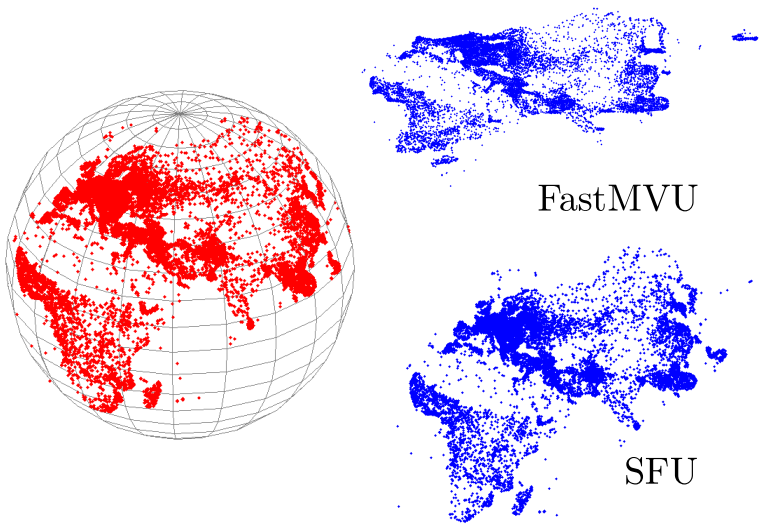

Figure 2: Left: The original data set consisting of 15,040 randomly selected cities in Europe, Africa, and Asia. Right: FastMVU (top) and SFU (bottom) embedding in two dimensions.

5.2.2. Images of Faces

In this experiment, we have used 1965 28×20 pixel images of Brendan Frey's face taken from a short video[2]. These images show his face in different moods: happy, angry, frowning,

2. Available from http://cs.nyu.edu/~roweis/data/frey_rawface.mat.

and so on. We applied SFU to this data set. APC found 70 clusters and SDP was solved in 5 seconds. We also applied FastMVU with $k = 30$ and $v = 10$, which terminated in 59 seconds. The embeddings are depicted in Fig. 3. It can be observed that in SFU's embedding the images on the lower left show him in a happy mood, while the images on the upper right show him in a frowning/unhappy mood. Moreover, similar facial expressions are all clustered together. On the other hand, FastMVU produces an embedding, where no clear pattern can be observed.

6. Conclusions

In this paper, we have proposed a new large-scale nonlinear dimensionality reduction method based on semidefinite facial reduction. While there are several well-known SDP-based dimensionality reduction methods, their effectiveness is limited by the computational complexity of the SDP solvers. Our experimental results have demonstrated the effectiveness of SFU on a variety of very large data sets.

References

Babak Alipanahi, Nathan Krislock, Ali Ghodsi, Henry Wolkowicz, Logan Donaldson, and Ming Li. Protein structure by semidefinite facial reduction. In Benny Chor, editor, *Research in Computational Molecular Biology*, volume 7262 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin / Heidelberg, 2012.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings NIPS*, 2001.

A. Ben-Tal and A.S. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, volume 2. Society for Industrial Mathematics, 2001.

Brian Borchers and Joseph G. Young. Implementation of a primal-dual method for SDP on a shared memory parallel architecture. *Comput. Optim. Appl.*, 37(3):355–369, 2007.

Jonathan M. Borwein and Henry Wolkowicz. Facial reduction for a cone-convex programming problem. *J. Austral. Math. Soc. Ser. A*, 30(3):369–380, 1981.

T. Cox and M. Cox. *Multidimensional Scaling*. Chapman Hall, Boca Raton, 2nd edition, 2001.

Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007. URL www.psi.toronto.edu/affinitypropagation.

J. Ham, D. Lee, S. Mika, and Schölkopf B. A kernel view of the dimensionality reduction of manifolds. In *International Conference on Machine Learning*, 2004.

Nathan Krislock and Henry Wolkowicz. Explicit sensor network localization using semidefinite representations and facial reductions. *SIAM Journal on Optimization*, 20(5):2679–2708, 2010.

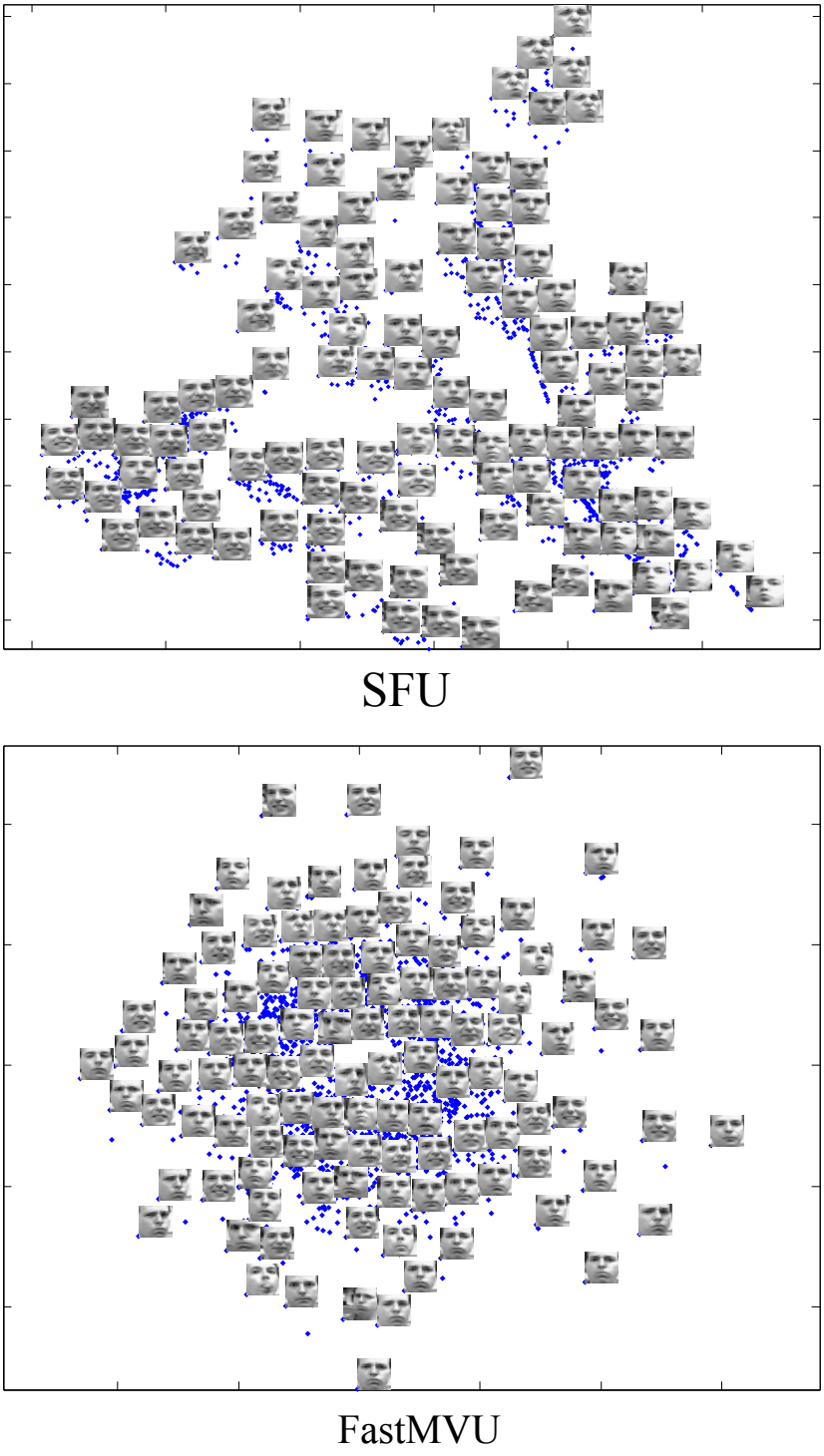

Figure 3: Visulization of Frey faces in two dimensions using SFU and FastMVU.

S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Proceedings NIPS 11*. MIT Press, 1999.

Motakuri V. Ramana, Levent Tunçel, and Henry Wolkowicz. Strong duality for semidefinite programming. *SIAM Journal on Optimization*, 7(3):641–662, 1997.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

I. J. Schoenberg. Remarks to Maurice Fréchet's article "Sur la définition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de Hilbert". *Ann. of Math. (2)*, 36(3):724–732, 1935.

B. Shaw and T. Jebara. Minimum volume embedding. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics March 21-24, 2007, San Juan, Puerto Rico*, volume 2 of JMLR: W&CP, pages 460–467, March 2007.

Blake Shaw and Tony Jebara. Structure preserving embedding. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 937–944, Montreal, June 2009. Omnipress.

Le Song, Alex Smola, Karsten M. Borgwardt, and Arthur Gretton. Colored maximum variance unfolding. In *NIPS*. MIT Press, 2007.

J. Tenenbaum. Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems 10*, pages 682–687, 1998.

K.-C. Toh, R. H. Tütüncü, and M. J. Todd. On the implementation and usage of SDPT3 — a MATLAB software package for semidefinite–quadratic–linear programming, version 4.0. user's guide. 2006.

R. H. Tütüncü, K.-C. Toh, and M. J. Todd. Solving semidefinite–quadratic–linear programs using SDPT3. *Mathematical Programming*, 95:189–217, 2003.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-04)*, volume II, pages 988–995, 2004.

K.Q. Weinberger, F. Sha, Q. Zhu, and L.K. Saul. Graph Laplacian regularization for large-scale semidefinite programming. *Advances in neural information processing systems*, 19: 1489–1496, 2006.

Xiao-Ming Wu, Anthony Man-Cho So, Zhenguo Li, and Shuo-Yen Robert Li. Fast graph Laplacian regularized kernel learning via semidefinite-quadratic-linear programming. In *Neural Information Processing Systems (NIPS)*, pages 1964–1972, 2009.