Minimum Volume Embedding

Blake Shaw
Computer Science Dept.
Columbia University
New York, NY 10027

Tony Jebara
Computer Science Dept.
Columbia University
New York, NY 10027

Abstract

Minimum Volume Embedding (MVE) is an algorithm for non-linear dimensionality reduction that uses semidefinite programming (SDP) and matrix factorization to find a low-dimensional embedding that preserves local distances between points while representing the dataset in many fewer dimensions. MVE follows an approach similar to algorithms such as Semidefinite Embedding (SDE), in that it learns a kernel matrix using an SDP before applying Kernel Principal Component Analysis (KPCA). However, the objective function for MVE directly optimizes the eigenspectrum of the data to preserve as much of its energy as possible within the few dimensions available to the embedding. Simultaneously, remaining eigenspectrum energy is minimized in directions orthogonal to the embedding thereby keeping data in a so-called minimum volume manifold. We show how MVE improves upon SDE in terms of the volume of the preserved embedding and the resulting eigenspectrum, producing better visualizations for a variety of synthetic and real-world datasets, including simple toy examples, face images, handwritten digits, phylogenetic trees, and social networks.

1 INTRODUCTION

In machine learning, computer vision, computational biology and other applied areas, datasets often involve high dimensional objects. While the extrinsic dimensionality of such datasets may be high, most of the dataset's variability can often be captured by far fewer dimensions. For example, data points may consist of a large number of features that are not independent because of underlying constraints in the data. In practice, such data points lie on a low-dimensional nonlinear manifold and only a few intrinsic coordinates are necessary to characterize their variation. Nonlinear dimensionality reduction methods recover such a manifold from data and provide low-dimensional representations of the original high-dimensional data. This can be useful for 2D or 3D visualization or for further processing by other algorithms.

Most nonlinear dimensionality reduction techniques begin with the following goal: given N points in a high-dimensional space $\vec{x_i} \in \Re^D$ for $i = 1 \dots N$ find a low-dimensional representation of corresponding points $\vec{y_i} \in \Re^d$ for $i = 1 \dots N$ such that $d \ll D$ which preserves local relationships or distances in the data. In kernel PCA [7], this is done by keeping the d dimensions that have the largest eigenvalues. However, there is always some corruption or loss when we drop dimensions with non-zero eigenvalues, and in turn pairwise distances are less perfectly preserved. Semidefinite embedding [11] instead unfolds the data manifold (without violating local relationships) prior to dimensionality reduction. This ensures that eigenvalues measure variation aligned with the manifold and perpendicular to it rather than across spurious directions in the original space. In practice, SDE drives some energy into the lower d dimensions such that less energy is lost after kernel PCA and local pairwise distances are more faithfully preserved. However, unfolding by maximizing variance does not explicitly target our goal of keeping energy in the first d components of kernel PCA. In this article, we explicitly unfold and increase energy in the top d dimensions while we simultaneously collapse and decrease the energy in the remaining $D - d$ dimensions. This ensures that as much energy as possible remains when we perform kernel PCA, and that the pairwise distances are as faithful as possible after we reduce dimensionality to d dimensions. We refer to this approach as minimum volume embedding (MVE) and provide a convergent and efficient algorithm. In driving away energy from the dimensions after the d'th one, we effectively minimize the volume the embedding occupies and flatten it out.

This paper is organized as follows. In Section 2 we review prior work in nonlinear dimensionality reduction and embedding. In Section 3, we argue that these algorithms should preserve as much eigenspectrum energy as possible in the reduced dimensionality. Section 4 shows how MVE attains this goal via a novel cost function, a variational upper bound, and an algorithm that minimizes it. Section 5 provides experiments on large datasets and shows favorable MVE performance compared to other manifold learning algorithms. We conclude with a discussion in Section 6.

2 CURRENT EMBEDDING ALGORITHMS

Dimensionality Reduction is a well studied problem, and there exist a number of algorithms in this field. At the foundation of many of these techniques is Principal Component Analysis (PCA) [7], an algorithm which linearly maps a set of data onto a new coordinate system, such that the 1st dimension captures the largest amount of the variance, the 2nd captures the next largest and so on. PCA can accurately recover a low-dimensional embedding if the data lies near a linear manifold. However, if the underlying manifold is not linear, a variety of extensions to PCA have been devised.

Locally Linear Embedding (LLE) [6], hLLE [2], Laplacian Eigenmaps [1], Isomap [8], and Semidefinite Embedding [11], all provide different techniques for capturing the non-linearity of the underlying manifold incorporating local distance information in different ways. LLE, for example, only considers local pairwise information between points, approximating the nonlinear manifold by a set of linear patches. Similarly, Laplacian Eigenmaps operates in this local regime. Isomap, on the other hand, operates globally on the set of all distances between points. It uses local information to construct a k-nearest neighbor graph and estimates distances between far away points by considering the shortest path on the graph.

Operating in both a local and global regime, Semidefinite Embedding, also known as Maximum Variance Unfolding, tries to preserve local distances as it "unfolds" the data by optimizing globally over all pairwise distances. SDE, for most cases, is arguably the leading embedding algorithm, able to more accurately recover the underlying non-linear manifold [11]. SDE works as follows. Given input data $\vec{x_i} \in \Re^D$ for $i = 1 \ldots N$, SDE forms an affinity matrix A using any choice of pairwise affinity metrics such as a linear kernel, or an RBF kernel. A is then used to generate a connectivity matrix C where typically each point is connected to its k-nearest neighbors, and k is a parameter of the algorithm. This connectivity structure enforces which local distances will be preserved. SDE then learns a kernel matrix K by maximizing $tr(K)$, while preserving the constraints that $K_{i,i}+K_{j,j}-K_{i,j}-K_{j,i} = A_{i,i}+A_{j,j}-A_{i,j}-A_{j,i}\forall_{i,j}$ where $C_{i,j} = 1$. K is then used in KPCA, to get a set of eigenvectors $\vec{v_i} \in \Re^d$ for $i = 1 \ldots N$, where typically the number of eigenvectors corresponding to large eigenvalues is much less than the dimensionality of the data.

At the core of many of these algorithms lies this PCA-like projection to select the d strongest eigenvectors, typically using the "kernel trick" to extend PCA. In doing so, these algorithms lose any information corresponding to dropped eigenvectors with non-zero eigenvalues. Furthermore, none of these algorithms explicitly aim to minimize the amount of information lost due to this truncation. Instead the process of learning a kernel is inherently separated from the step of truncating its low-energy eigenvectors.

3 EVALUATING EMBEDDINGS

How can we judge the quality of an embedding? This question is difficult to answer quantitatively. One proposed measure is to examine the spectrum of eigenvalues, and see if it correctly corresponds to the underlying dimensionality of the data [11]. If we know the underlying dimensionality for synthetic data sets, we can see if the number of large eigenvalues of the embedding corresponds to the intrinsic dimensionality of the data. Furthermore, if we are using the embedding algorithm as a visualization tool, by truncating the eigenspectrum, and only looking at the top 2 or 3 eigenvectors corresponding to strong eigenvalues, we should make sure that we capture a large percentage of the variance of the data in these top eigenvectors. Furthermore, the eigengap (the difference between the smallest eigenvalue preserved for embedding and the largest eigenvalue truncated during PCA) should be as big as possible. Capturing a large percentage of the variance in the top eigenvalues ensures that local distances are actually preserved when we view the data in only a few dimensions, and the large eigen-gap guarantees stability in the algorithm, making sure that subtle changes in the data don't cause eigenvectors to switch places, drastically altering the embedding [5].

Semidefinite Embedding has been shown to produce low rank kernel matrices for a variety of datasets [11]. However, the objective function that SDE maximizes is very much at odds with the intuition that we want: to reduce the dimensionality of the data. This is clear from a variety of datasets. In Figure 1, we see a synthetic data set consisting of a hub and spokes; the spokes are bent down from the hub. This simple dataset nicely illustrates an inherent limitation in SDE. The data exists in 3 dimensions, and it is clear that by simply unbending the spokes, we could represent this structure in 2D, preserving local relationships. However, because SDE is trying to maximize $tr(K)$, it is in essence trying to pull the data apart in every dimension. This maximizes the volume of the spokes into an N-dimensional spherical cloud. Figure 1 shows the resulting embedding and the eigenvalues before and after SDE which was actually detrimental to the original PCA spectrum in the top of the figure.

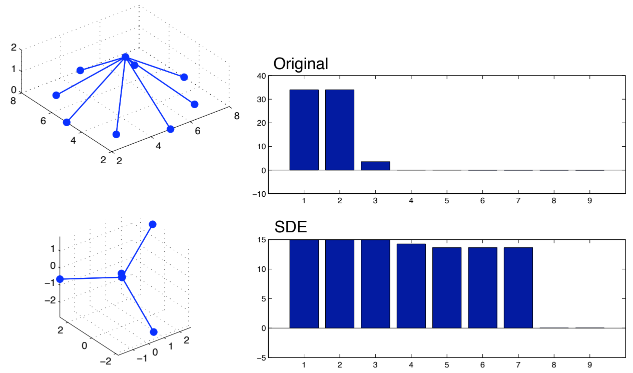

Figure 1: Top, the original data and its eigenspectrum (using PCA). Bottom, the unfolded data using SDE and its eigenspectrum. The embedding (left) generated for a synthetic hub and spokes dataset and then processed by Semidefinite Embedding. SDE pulls apart the data into an N-dimensional space, which is evident from the overpopulated eigenspectrum (right).

The original motivation for developing MVE was to visualize social networks which have a connectivity structure similar to that of this toy problem. However, as we experimented with more datasets, we saw that by using a more sophisticated objective function not only can we avoid the limitations of SDE as shown above, but we can consistently capture more of the variance of the data in the top eigenvectors for a variety of different kinds of datasets. This allows MVE to more accurately visualize the data in only a few dimensions.

4 THE MINIMUM VOLUME EMBEDDING ALGORITHM

To formulate MVE, we start from the SDE algorithm and make some important changes to its cost function. Recall that SDE maximizes the trace of a matrix K subject to positive definiteness, centering and distance-preserving constraints. It is a constrained minimization of a linear cost function of K:

$$\min_{K\in\mathcal{K}} f_{SDE}(K) \quad = \quad \min_{K\in\mathcal{K}} -tr(K) \;=\; \min_{K\in\mathcal{K}} -\sum_{i=1}^{N}\lambda_i$$

(where λ_i are the eigenvalues of K) subject to the constraint that K is in $\mathcal{K}$. Here, we use $\mathcal{K}$ to denote the convex set of matrices which satisfy the properties of SDE and define it as:

$$\mathcal{K} = \left\{ \forall K \in \Re^{N\times N} \;\middle|\; \begin{array}{l} K \succeq 0 \\ \sum_{ij} K_{ij} = 0 \\ K_{ii} + K_{jj} - K_{ij} - K_{ji} = \\ \qquad A_{ii} + A_{jj} - A_{ij} - A_{ji} \\ \qquad\quad \forall\, i,j \text{ when } C_{ij} = 1 \end{array} \right\}$$

While SDE sometimes works reasonably well, the intuition of pulling points apart (maximizing the trace of K) and unfolding by maximizing variance can create problems and use more dimensions than are necessary. Instead, we would like to pull points apart in the dimensions that we are interested in keeping for the embedding but reduce the variance in dimensions that will be removed. Thus, we would like to grow the top few eigenvalues of K while shrinking the remaining ones (to avoid the manifold puffing out in directions perpendicular to its surface). Ideally, if we knew the intrinsic dimensionality d of the manifold, MVE would therefore minimize the following cost function over the eigenvalues:

$$\min_{K\in\mathcal{K}} f(K) \quad = \quad \min_{K\in\mathcal{K}} -\sum_{i=1}^{d}\lambda_i + \sum_{i=d+1}^{N}\lambda_i \tag{1}$$

where λ are the eigenvalues of K in sorted order, $\lambda_i \geq \lambda_{i+1}$. We typically do not know d but treat it as a user-specified parameter for now. Typically, for visualization d is 2 or 3. Furthermore, our experiments show that using a value of d that is too small or too big will not significantly change results. Clearly, if we set $d = N$, this cost function f becomes the same as the cost function f_{SDE} used in SDE. We cannot directly minimize Equation 1, so we derive a variational upper bound on it, and iteratively minimize that bound.

Consider rewriting Equation 1 in the following way which makes the relationship between K, its eigenvalues λ_i and its eigenvectors $\vec{v}_i$ more explicit:

$$\min_{K\in\mathcal{K}} f(K) \;=\; \min_{K\in\mathcal{K}} -\sum_{i=1}^{d}\lambda_i + \sum_{i=d+1}^{N}\lambda_i$$
$$\text{s.t. } K\vec{v}_i = \lambda_i\vec{v}_i,\; \vec{v}_i^T\vec{v}_j = \delta_{ij},\; \lambda_i \geq \lambda_{i+1},\; \forall i,j. \tag{2}$$

We next manipulate the cost function algebraically (maintaining the above additional constraints) as follows:

$$\begin{aligned} f(K) &= -\sum_{i=1}^{d}\lambda_i + \sum_{i=d+1}^{N}\lambda_i \\ &= -\sum_{i=1}^{d} tr(\lambda_i \vec{v}_i\vec{v}_i^T) + \sum_{i=d+1}^{N} tr(\lambda_i \vec{v}_i\vec{v}_i^T) \\ &= -\sum_{i=1}^{d} tr(K \vec{v}_i\vec{v}_i^T) + \sum_{i=d+1}^{N} tr(K \vec{v}_i\vec{v}_i^T) \\ &= tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right] \end{aligned}$$

Therefore, the MVE problem can be rewritten as:

$$\min_{K\in\mathcal{K}} f(K) = \min_{K\in\mathcal{K}} tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right]$$
$$\text{s.t. } K\vec{v}_i = \lambda_i\vec{v}_i,\ \vec{v}_i^T\vec{v}_j = \delta_{ij},\ \lambda_i \geq \lambda_{i+1},\ \forall i,j.$$

Interestingly, $f(K)$ does not necessarily need all the above constraints. Consider minimizing $f(K)$ over an *arbitrary* set of eigenvectors $\vec{v}_1,\ldots,\vec{v}_N$ as follows:

$$\min_{\substack{\vec{v}_1,\ldots,\vec{v}_N \\ \vec{v}_i^T\vec{v}_j=\delta_{ij}}} tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right]$$

This is simply a *Procrustes* problem which is straightforward to solve. Clearly, to minimize the negative term, we set $\vec{v}_1,\ldots,\vec{v}_d$ to the top d eigenvectors of K. Similarly, to minimize the positive term, we set $\vec{v}_{d+1},\ldots,\vec{v}_N$ to the bottom $N-d$ eigenvectors of K. Therefore, minimizing f over eigenvectors:

$$\min_{\substack{\vec{v}_1,\ldots,\vec{v}_N \\ \vec{v}_i^T\vec{v}_j=\delta_{ij}}} tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right]$$

is *equivalent* to the value

$$tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right]$$
$$\text{s.t. } K\vec{v}_i = \lambda_i\vec{v}_i,\ \vec{v}_i^T\vec{v}_j = \delta_{ij},\ \lambda_i \geq \lambda_{i+1},\ \forall i,j.$$

under the more rigid set of constraints. In other words, minimization over arbitrary eigenvectors is equivalent to setting them to the eigenvectors of the matrix K.

These steps finally let us write a variational version of the MVE problem in Equation 2 which is straightforward to minimize. The variational version makes the choice of eigenvectors an additional parameter:

$$\min_{K\in\mathcal{K}} \min_{\substack{\vec{v}_1,\ldots,\vec{v}_N \\ \vec{v}_i^T\vec{v}_j=\delta_{ij}}} tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right] \quad (3)$$

When we minimize over the additional parameter, we get the MVE formulation in Equation 2. Thus, we can now approach the minimization problem for MVE as an alternating minimization found by iterating between solving for the best K while the set of eigenvectors is fixed and then solving for the best set of eigenvectors given K. We simply need to initialize the algorithm with either a starting K matrix or a starting set of eigenvectors $\vec{v}_1,\ldots,\vec{v}_N$. For instance, we may begin with $K = A$, the original affinity matrix between pairs of the input data-points or initialize K with the solution found by SDE.

Table 1 summarizes the MVE algorithm under the alternating minimization scheme.

Input	$(\vec{x}_i)_{i=1}^N$, kernel κ, and parameters d, k.
Step 1	Form affinity matrix $A \in \Re^{N\times N}$ with pairwise entries $A_{ij} = \kappa(\vec{x}_i, \vec{x}_j)$.
Step 2	Use A to find a binary connectivity matrix C via k-nearest neighbors.
Step 3	Initialize $K = A$.
Step 4	Solve for the eigenvectors $\vec{v}_1,\ldots,\vec{v}_N$ and eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ of K.
Step 5	Set $B = -\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T$.
Step 6	Using SDP find $\hat{K} = \arg\min_{K\in\mathcal{K}} tr(KB)$.
Step 7	If $\lVert K - \hat{K}\rVert \geq \epsilon$ set $K = \hat{K}$, go to Step 4.
Step 8	Perform kernel PCA on $\hat{K}$ to get d-dimensional output vectors $\vec{y}_1,\ldots,\vec{y}_N$.

Table 1: Minimum Volume Embedding Algorithm.

Although we have not proven that Equation 3 will converge to a global minimum, we can prove that it will converge to a local minimum, since it is a variational bound on the original problem in Equation 2. Furthermore, if we initialize the algorithm with the kernel PCA solution (as in Table 1 $K = A$), we are guaranteed to improve the cost function beyond the kernel PCA solution. Similarly, if we initialize the algorithm with the SDE solution, we are guaranteed to improve the cost function from that seed as well. We formalize the monotonicity of the MVE algorithm with the following theorem.

Theorem 1 *The iterative MVE algorithm is guaranteed to monotonically decrease the cost function* $f(K) = -\sum_{i=1}^{d}\lambda_i + \sum_{i=d+1}^{N}\lambda_i$.

Proof 1 *For any* $K \in \mathcal{K}$, *recall that* $f(K) =$

$$\min_{\substack{\vec{v}_1,\ldots,\vec{v}_N \\ \vec{v}_i^T\vec{v}_j=\delta_{ij}}} tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right].$$

Define the function:

$$g(K, \vec{v}_1, \ldots, \vec{v}_N) = tr\left[K\left(-\sum_{i=1}^{d}\vec{v}_i\vec{v}_i^T + \sum_{i=d+1}^{N}\vec{v}_i\vec{v}_i^T\right)\right].$$

Thus, $f(K)$ is the minimization of $g(K, \vec{v}_1, \ldots, \vec{v}_N)$ over eigenvectors. Assume we have a current setting of K denoted K^t and a current setting of the eigenvectors $\vec{v}_1^t, \ldots \vec{v}_N^t$. In general, we must have:

$$f(K^t) \quad \leq \quad g(K^t, \vec{v}_1^t, \ldots, \vec{v}_N^t)$$

After Step 4 of the MVE algorithm we obtain:

$$f(K^t) \quad = \quad g(K^t, \vec{v}_1^{t+1}, \ldots, \vec{v}_N^{t+1})$$

After Step 6 of the MVE algorithm, SDP ensures that:

$$g(K^{t+1}, \vec{v}_1^{t+1}, \ldots \vec{v}_N^{t+1}) \quad \leq \quad g(K^t, \vec{v}_1^{t+1}, \ldots \vec{v}_N^{t+1}).$$

Since $f(K)$ is a minimization of $g(K, \vec{v}_1, \ldots, \vec{v}_N)$,

$$\begin{aligned} f(K^{t+1}) &\leq g(K^{t+1}, \vec{v}_1^{t+1}, \ldots, \vec{v}_N^{t+1}) \\ &\leq g(K^t, \vec{v}_1^{t+1}, \ldots, \vec{v}_N^{t+1}) \\ &\leq f(K^t) \end{aligned}$$

showing that $f(K^t) \geq f(K^{t+1})$ after each loop of the MVE algorithm and $f(K)$ decreases monotonically.

It is interesting to note that spectral cost functions of the matrix K of the form $\sum_i \alpha_i \lambda_i$ are convex if $\alpha_i \geq \alpha_{i+1}$ and the eigenvalues of K are arranged in decreasing order $\lambda_i \geq \lambda_{i+1}$. Since MVE's cost function has $\alpha_i \leq \alpha_{i+1}$, it is concave. In practice, the MVE algorithm does not seem to have any significant local minima since it converges reliably to the same solution despite variations in initialization, as shown in Figure 2. In addition, the solution it obtains is consistently superior to ones found by SDE and kernel PCA. Another interesting property of MVE is that the selection of d does not need to be precise. For instance, setting $d = 2$ and computing a 3D visualization in Step 8 of the algorithm will still produce a better visualization than using traditional SDE or kernel PCA techniques. These empirical advantages are presented in further detail in the next section.

5 EXPERIMENTS

We present a variety of experiments using both synthetic and real-world data to highlight the performance of MVE, specifically in comparison to SDE and KPCA.

5.1 SYNTHETIC DATA

In Figure 3, we see MVE appropriately embedding the simple data set consisting of a hub and spokes. We saw in Figure 1 that SDE puffs out this dataset into an N-dimensional hub. Meanwhile, MVE recovers the correct minimum volume two-dimensional embedding. Although this example is very manufactured, it clearly highlights a deficiency in SDE, and one can imagine stringing together many of these hubs in a variety of different ways to create hierarchical hub-like structures that could cause SDE to use many more dimensions than necessary.

Figure 4 shows a synthetic dataset consisting of 50 points sampled from a 2 dimensional spiral, where each of the 50 points is connected to its 3 nearest neighbors. As expected, both MVE and SDE find nearly identical embeddings using the same parameters ($k = 3$) for both, and $d = 1$ for MVE. Both algorithms successfully reveal the underlying 1-dimensional structure in the data.

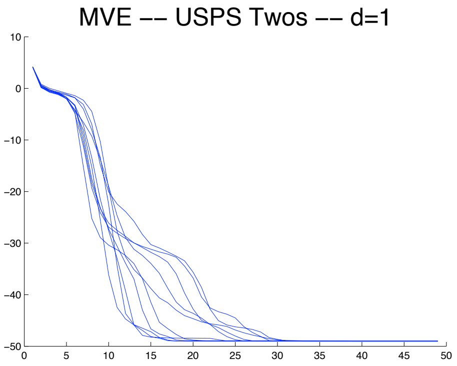

Figure 2: The results of using 10 random initializations for MVE, instead of initializing with the KPCA or SDE solution. We see that for $d = 1$, MVE converges each time to the same solution, showing that MVE is robust to local minima. We also note that seeding MVE with KPCA reduces the number of iterations by 5 fold in practice.

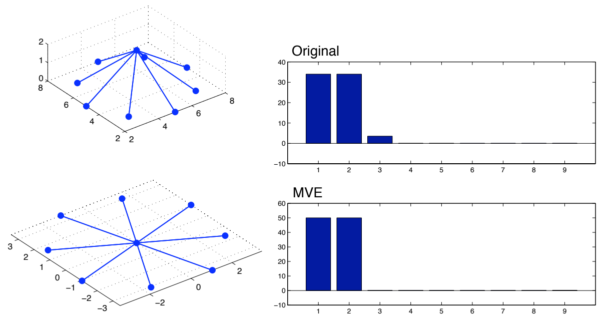

Figure 3: MVE correctly embeds the synthetic hub and spokes data in 2 dimensions (left). The eigenspectrum (right) of the embedding generated by MVE shows that the embedding is simply 2 dimensional and that MVE does not puff out the data into an N-dimensional space like SDE.

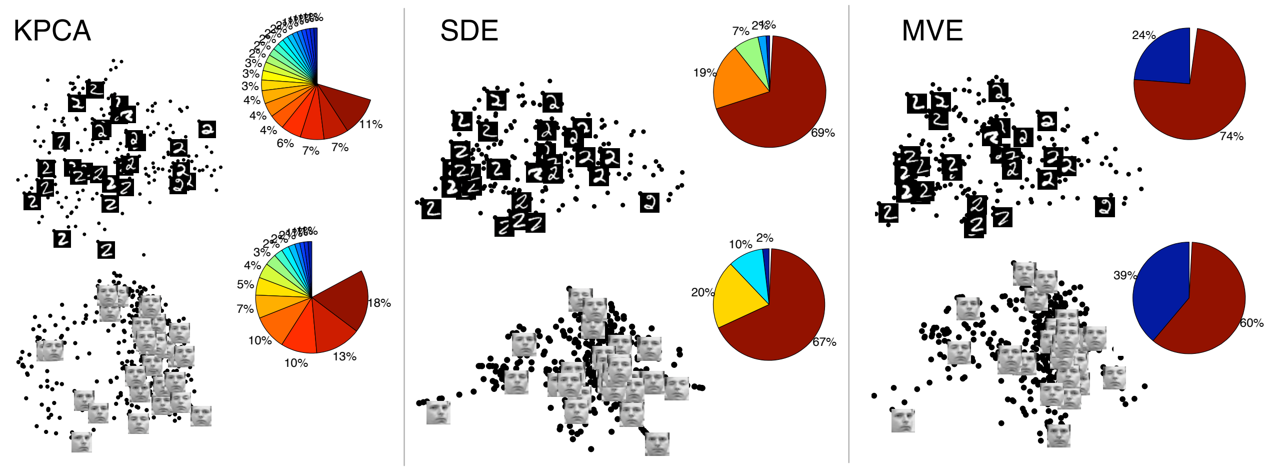

Figure 5: A comparison of the embeddings for the faces (top) and twos (bottom) image datasets. KPCA (left), SDE (middle), and MVE (right). The pie charts indicate the eigenvalue spectra corresponding to each embedding. It is clear that MVE captures more of the variance in the top 2 dimensions providing a more accurate 2D embedding.

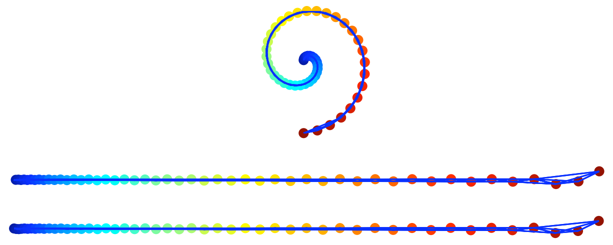

Figure 4: The embeddings of SDE (middle) and MVE (bottom) for a simple synthetic 2-dimensional spiral dataset (top). We see that SDE and MVE both capture the inherent low-dimensional manifold from which the data was sampled.

5.2 REAL-WORLD DATA

Dimensionality reduction algorithms such as MVE can provide great insight into datasets of images, allowing one to simply input each image as a vector of pixel intensity values, and automatically capture a small number of meaningful features that vary from image to image, such as position, rotation, pose, lighting, etc. [10].

Figure 5 shows 400 faces taken from the Frey Face dataset. Each image is vectorized as a 560 element vector of pixel intensity values. As in previous work on embedding images [10], SDE is capable of capturing an embedding which allows us to visualize the changes in the face somewhat accurately in only a few dimensions. Here we present a direct comparison of SDE vs. MVE on this dataset and show that we are able to more accurately represent the data in two dimensions using MVE because we capture significantly more of the variance of the data in the top two eigenvectors. This is shown by the pie charts in Figure 5 which present the eigenspectra for the three embedding methods. MVE seems to captures 10% more of the variance of the data in the top two dimensions, meaning local distances in MVE's 2D plot are more accurate, and more informative.

Figure 5 also shows 200 images of handwritten twos taken from the USPS handwritten digit dataset. As with the previous example with faces, we see that MVE is able to capture more of the variance in the top 2 eigenvectors, providing a more accurate lower dimensional embedding.

MVE was designed from the beginning to elegantly visualize social networks. In Figure 6, we see data mined from a web-based social network. The names have been anonymized. Connections between individuals represent whether or not those individuals have listed each other as friends. We use these friend connections as the C matrix for both MVE and SDE. The data consists of 19 points which exist in a 41839 dimensional space consisting of bag-of-words frequency vectors from the text in the person's profile. These 19 points are a subnetwork consisting only of the friends of the Zeus profile. For MVE d was set to 2.

Figure 7 shows a direct comparison between the eigenspectrum of the kernel produced by MVE (left), and SDE (right). We see from the plot that MVE captures a significantly greater amount of variance of the data in the top few eigenvectors. The long tail of eigenvalues in the SDE plot corresponds to information that will be lost if we truncate the eigenvectors at 2 or 3

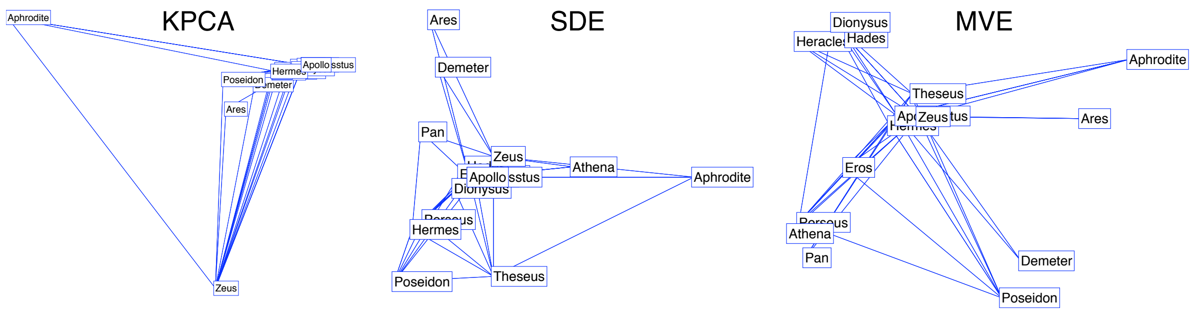

Figure 6: Embedding of social networks. KPCA (left), SDE (middle), and MVE (right).

dimensions. Furthermore, we can see in Figure 6 that KPCA alone does a poor job of embedding the data in two dimensions, allowing us to see only a small percentage of the variance in the data.

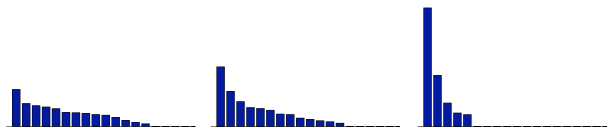

Figure 7: Eigenvalue spectra for the embeddings of social network data by KPCA (left), SDE (middle), and MVE (right).

We see from visualizing social networks that MVE provides a much greater advantage over SDE when the connectivity structure C is not a uniformly sampled mesh but takes on different structures, such as hubs and links. This advantage allows us to tackle visualization problems ordinarily not handled by algorithms such as SDE, for example phylogentic trees. For visualizing trees, it is helpful to modify MVE and SDE with added constraints that prevent all pairwise distances from shrinking: $K_{i,i} + K_{j,j} - K_{i,j} - K_{j,i} \geq A_{i,i} + A_{j,j} - A_{i,j} - A_{j,i} \forall_{i,j}$. This prevents the embedding from collapsing upon itself due to sparse connectivity. Figure 8 shows the result applied to the phylogenetic trees of salamanders and crustaceans. Here, connectivity is sparse and is given by a spanning tree. The figure shows MVE, SDE, and their modified versions with full constraints which we call MVE-Full and SDE-Full. Clearly, MVE-Full provides the best visualization of the tree structure.

5.3 SUMMARY

The following table summarizes the comparisons between MVE, SDE, and KPCA for the experiments in this paper in terms of accurately being able to represent the data in 2D. We see that in all cases MVE is able to capture more of the variance of the data in the first two eigenvectors, providing a more accurate 2-dimensional embedding. Although distances between points in local neighborhoods are preserved in the kernels learned by both MVE and SDE, as the sum of the first two normalized eigenvalues decreases from 100%, we can no longer say that local distances are being preserved in the resulting embedding. This is because the algorithm truncates the last $N-2$ eigenvectors to create the embedding, and therefore any distance information which comes from those eigenvectors is lost. MVE performs better than SDE in this regard, achieving near 100% accuracy for the twos and faces datasets, and for the difficult social network dataset achieving a significant improvement over the other algorithms.

Percentage of eigenvalue energy captured in 2D

	MVE	SDE	KPCA
Hubs and Spokes	100%	29.9%	95.0%
Spiral (% in 1D)	99.9%	99.9%	45.8%
Twos	97.8%	88.4%	18.4%
Faces	99.2%	83.6%	31.4%
Social Networks	77.5%	41.7%	29.3%

5.4 COMPUTATIONAL COMPLEXITY

The running time for one iteration of MVE is identical to that of running SDE: $O(N^3 + C^3)$ where N is the number of points, and C is the number of constraints in the SDP [10]. Therefore the running time of MVE is proportional to the running time of SDE by some constant factor corresponding to the number of iterations needed until convergence. A better analysis of the rate of convergence will be a matter of future work. However, for the experiments listed above, a reasonable convergence was reached after approximately 10 iterations.

Currently, running MVE with a standard SDP solver is slow (up to ten times slower than SDE). However, because of its similarity to SDE, MVE should be able to benefit from any speed-ups for SDE including SDE-customized SDP solvers and speed-ups from approximation schemes such as Landmark Semidefinite Embedding (lSDE) [10].

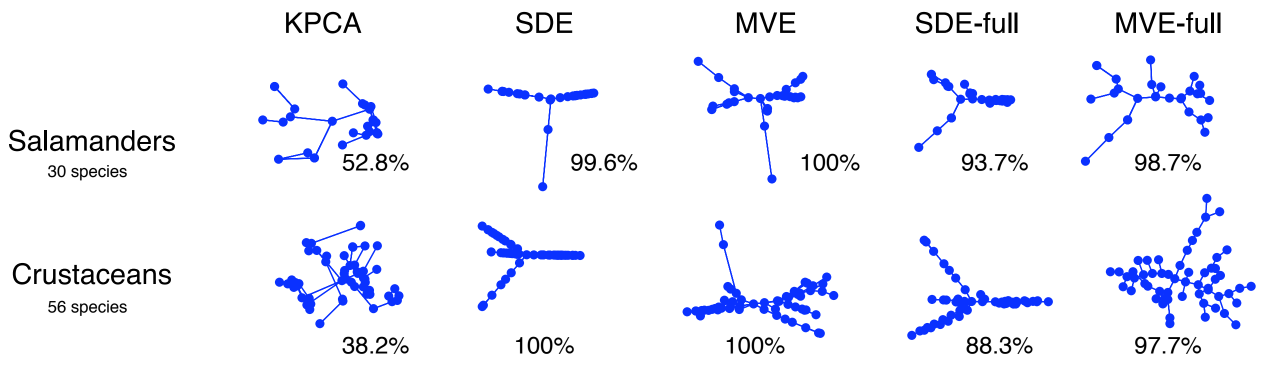

Figure 8: Embeddings of phylogenetic trees of Salamanders and Crustaceans using KPCA, SDE, MVE , SDE-Full, and MVE-Full. Also shown for each embedding is the percent of the variance which is captured in 2 dimensions. We see that MVE-Full provides the best visualization of the data.

6 CONCLUSION

The key motivation behind dimensionality reduction algorithms such as SDE is to accurately represent high-dimensional data in a lower dimensional space. In practice, algorithms such as SDE sacrifice some accuracy on the local distances when dropping dimensions with small but non-zero eigenvalues. Furthermore, in SDE, the optimization which unfolds the data in all directions is not complementary to the next step of truncating extra dimensions. So although local distances are preserved in the kernel learned by the optimization, local distances are not necessarily as well preserved in the low-dimensional embedding. MVE improves upon SDE and kernel PCA in this regard, by explicitly maximizing the amount of energy in the first d dimensions of the eigenspectrum, converging to a minimum volume solution which better preserves local distances in the resulting low-dimensional embedding. To highlight the advantages of MVE, we have shown experiments on a variety of synthetic and real-world datasets where MVE offered significant improvements over SDE in terms of capturing more of the variance of the data in the same few dimensions. Furthermore, we applied MVE to visualizing phylogenetic trees and social networks, showing its usefulness for high-dimensional datasets with unusual connectivity structures.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6), 2002.

[2] D. Donoho and C. Grimes. Hessian eigenmaps: locally linear embedding techniques for high dimensional data. *Proc. of National Academy of Sciences*, 100(10):5591–5596, 2003.

[3] T. Jebara. Convex invariance learning. In *Artificial Intelligence and Statistics (AISTAT)*, 2003.

[4] T. Jebara. Kernelizing sorting, permutation and alignment for minimum volume PCA. In *Conference on Learning Theory*, 2004.

[5] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems 14*, 2001.

[6] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2000.

[7] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Compuation*, 10, 1998.

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

[9] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[10] K. Q. Weinberger, B. D. Packer, and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Barbados, January 2005.

[11] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, Banff, Canada, 2004.

[1]This work was funded in part by NSF grant IIS-0347499