Minimum Volume Embedding

Blake Shaw and Tony Jebara
Columbia University

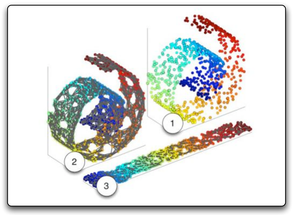
The Problem

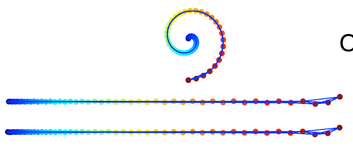
Visualizing data by *Unfolding*

How do we best visualize high-dimensional data? Given $\vec{x_i} \in \Re^D$ for $i = 1 \dots N$, can we find $\vec{y_i} \in \Re^d$ for $i = 1 \dots N$ such that $d \ll D$?

Standard PCA or Kernel PCA (Shoelkopf)
- Get matrix A, affinities between pairs $A_{ij} = k(x_i, x_j)$
- Get SVD of A and view top projections

Semidefinite Embedding (Weinberger, Saul)
- Get matrix A, compute k-nearest neighbor graph C
- Find K using a Semidefinite Program (SDP) which "pulls apart" A while preserving distances in C
- Get SVD of K and view top projections



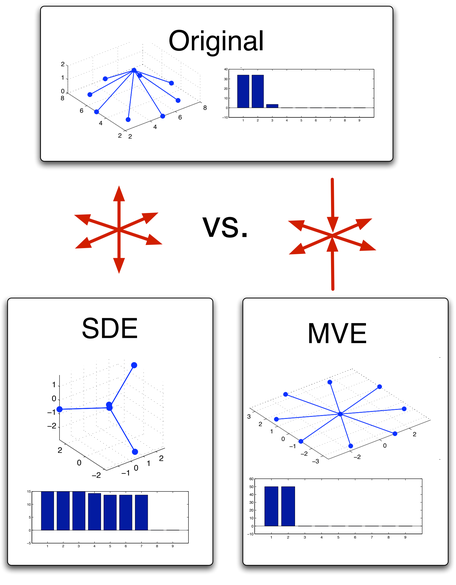


Semidefinite Embedding (SDE) pulls apart the swiss roll producing a 2d embedding. Minimum Volume Embedding (MVE) behaves similarly.

The Hubs and Spokes Problem
Unfolding and Flattening

Pulling apart the data in all directions could increase the dimensionality and *worsen* the visualization.

Instead, we want to pull apart the data only in the visualized dimensions and squash down the remaining ones.



The Algorithm

A Novel Cost Fuction

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.

SDE

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

The cost function of SDE stretches the data in all directions by maximizing the sum of the eigenvalues.

MVE

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

The cost function for MVE stretches only the top d dimensions and squashes the rest, directly optimizing the eigenspectrum of the data.

Constraints

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

An Iterated Monotonic SDP

The MVE cost function is not an SDP, but we can compute a variational upper bound on the the cost to get an Iterated Monotonic SDP.

$$\begin{aligned}
&\min_K -\sum_{i=1}^{d} \lambda_i + tr \sum_{i=d+1}^{D} \lambda_i \\
&\quad \text{s.t. } K \in \mathcal{K}, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij} \\
&= \min_K -\sum_{i=1}^{d} trKv_i v_i^T + tr \sum_{i=d+1}^{D} trKv_i v_i^T \\
&\quad \text{s.t. } K \in \mathcal{K}, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij} \\
&= \min_K \min_V -\sum_{i=1}^{d} trKv_i v_i^T + tr \sum_{i=d+1}^{D} trKv_i v_i^T \\
&\quad \text{s.t. } K \in \mathcal{K}, v_i^T v_j = \delta_{ij}
\end{aligned}$$

We lock V and solve an SDP for K, then lock K and solve an SVD for V. Note that finding the top eigenvectors of the current K is simply the Procrustes problem.

Discussion

Simply unfolding in every direction can go wrong. We optimize general spectral functions to minimize volume, improve eigengaps and reduce truncation error.

Algorithm Overview

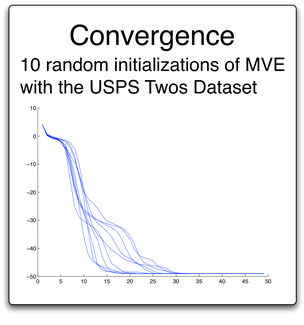
Input	$(\vec{x}_i)_{i=1}^N$, kernel κ, and parameters d, k.
Step 1	Form affinity matrix $A \in \Re^{N \times N}$ with pairwise entries $A_{ij} = \kappa(\vec{x}_i, \vec{x}_j)$.
Step 2	Use A to find a binary connectivity matrix C via k-nearest neighbors.
Step 3	Initialize $K = A$.
Step 4	Solve for the eigenvectors $\vec{v}_1, \dots, \vec{v}_N$ and eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$ 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 $\|K - \hat{K}\| \geq \epsilon$ set $K = \hat{K}$, go to Step 4.
Step 8	Perform kernel PCA on $\hat{K}$ to get d-dimensional output vectors $\vec{y}_1, \dots, \vec{y}_N$.

Table 1: Minimum Volume Embedding Algorithm.

Complexity and Convergence

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.

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. 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 which for the experiments on this poster was reached after approx. 10 iterations.



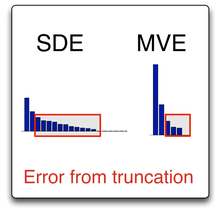
Extensions

We have found that MVE provides a much greater advantage over SDE when we extend each algorithm to use any specified connectivity structure C, as is shown in the social networks and phylogenetic trees examples. Note: 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.

Results

Experiments

The following experiments show that MVE more accurately represents the following datasets using only a few dimensions, as compared to SDE and KPCA. SDE sacrifices local distance accuracy when truncating extra dimensions. Because MVE explicitly maximizes the amount of energy in the top d dimensions, local distances are better preserved.

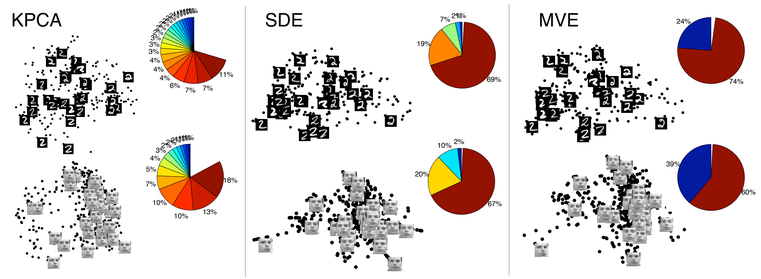


Summary of Results

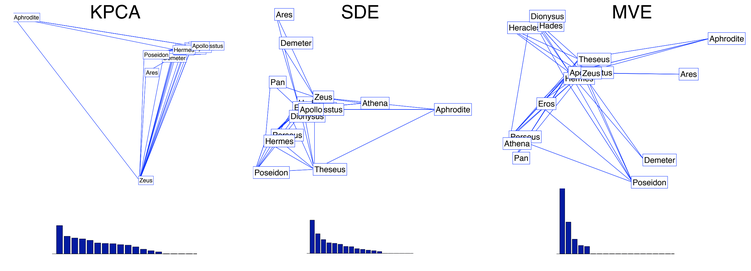
	MVE	SDE	KPCA
Hubs and Spokes	100%	29.9%	95.0%
Spiral (% in 1D)	99.9%	99.9%	45.8%
Twos	97.8%	88.4%	18.4%
Faces	99.2%	83.6%	31.4%
Social Networks	77.5%	41.7%	29.3%

Percentage of eigenvalue energy captured in 2D

Images of Twos and Faces



Social Networks



Phylogenetic Trees