Motivation

- Find meaningful low-dimensional structures hidden in high-dimensional observations.

The human brain confronts the same problem in perception:
- 30,000 auditory nerve fibers
- $10^6$ optic nerve fibers

$\Rightarrow$ extract small number of perceptually relevant features.

- Difficult to visualize data in dimensions greater than three.
**Informal definition:**
Manifold = any object which is nearly "flat" on small scales.

1dim manifolds:

2dim manifolds:
Manifold Learning

\[ \mathbb{R}^2 \]

\[ x_1 \]

\[ x_2 \]
Algorithms

- PCA (1901), kernel PCA
- Multi-dimensional Scaling (1952)
- Maximum Variance Unfolding, Colored MVU
- Mixture of PCA, Mixture of Factor Analyzers
- Local Linear Embedding (2000)
- Isomap (2000), C-Isomap
- Hessian Eigenmaps
- Local Tangent Space Alignment

- ... and many more
PCA is a linear method: it fails to find the nonlinear structure in the data
Issues with PCA

PCA uses the Euclidean distance

What is important is the geodesic distance

Unroll the manifold
Multi-dimensional Scaling
Multi-dimensional Scaling

- In PCA we are given a set of points

\[ X = [x_1, \ldots, x_n] \in \mathbb{R}^{l \times n} \]

- In MDS we are given pairwise distances instead of the actual data points.

- Question: If we only preserve the pairwise distances do we preserve the structure?
From Distances to Inner Products

$$d^2_{ij} = d^2_{ki} + d^2_{kj} - 2d_{ki}d_{kj} \cos(\alpha)$$

$$b_{ij} = d_{ki}d_{kj}\cos(\alpha)$$

$$b_{ij} = \frac{1}{2}(d^2_{ki} + d^2_{kj} - d^2_{ij}) = \langle X_i - X_k, X_j - X_k \rangle$$
From Distances to Inner Products

Similarly:

Center the data and then calculate $\langle x_i, x_j \rangle$

$$\langle x_i, x_j \rangle = G_{ij} = -\frac{1}{2} \left[ d_{ij}^2 - \frac{1}{n} \sum_{l=1}^{n} d_{il}^2 - \frac{1}{n} \sum_{m=1}^{n} d_{mj}^2 + \frac{1}{n^2} \sum_{o=1}^{n} \sum_{p=1}^{n} d_{op}^2 \right]$$

MDS cost function:

$$J_{MDS}(y_1, \ldots, y_n) = \sum_{i,j} \left( \langle x_i, x_j \rangle - \langle y_i, y_j \rangle \right)^2$$
MDS algorithm:

Step 1: Build a Gram matrix of inner products

\[ X = [x_1, \ldots, x_n] \in \mathbb{R}^{l \times n} \]
\[ G = \{ \langle x_i, x_j \rangle \}_{i,j} = X^T X \in \mathbb{R}^{n \times n} \]

Step 2: Find the top \( k \) eigenvectors of \( G \)

\[ [\psi_1, \ldots, \psi_k] \in \mathbb{R}^{n \times k} \]

with the top \( k \) eigenvalues: \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_k) \in \mathbb{R}^{k \times k} \)

Step 3:  
\[ [y_1, \ldots, y_n] = \Lambda^{1/2} [\psi_1, \ldots, \psi_k]^T \in \mathbb{R}^{k \times n} \]
Observation: If the data is centered, then the Gram matrix can be found this way:

\[ S_{ij} = \|x_i - x_j\|^2 \quad u = \frac{1}{\sqrt{n}}[1, \ldots, 1]^T \]

\[ G = -\frac{1}{2}(I - uu^T)S(I - uu^T) \]

PCA operates on \( \frac{1}{n}XX^T \in \mathbb{R}^{l \times l} \)

MMD operates on \( G = X^TX \in \mathbb{R}^{n \times n} \)

Though based on a somewhat different geometric intuition, metric MDS is closely related to PCA.

There are many different versions of MDS…
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NY</td>
<td>0</td>
<td>206</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>206</td>
<td>0</td>
<td>233</td>
<td>1308</td>
<td>802</td>
<td>2815</td>
<td>2934</td>
<td>2786</td>
<td>1771</td>
</tr>
<tr>
<td>MIA</td>
<td>429</td>
<td>233</td>
<td>0</td>
<td>1075</td>
<td>671</td>
<td>2684</td>
<td>2799</td>
<td>2631</td>
<td>1616</td>
</tr>
<tr>
<td>MIAMI</td>
<td>1504</td>
<td>1308</td>
<td>1075</td>
<td>0</td>
<td>1329</td>
<td>3273</td>
<td>3053</td>
<td>2687</td>
<td>2037</td>
</tr>
<tr>
<td>CHICAGO</td>
<td>963</td>
<td>802</td>
<td>671</td>
<td>1329</td>
<td>0</td>
<td>2013</td>
<td>2142</td>
<td>2054</td>
<td>996</td>
</tr>
<tr>
<td>SEATTLE</td>
<td>2976</td>
<td>2815</td>
<td>2684</td>
<td>3273</td>
<td>2013</td>
<td>0</td>
<td>808</td>
<td>1131</td>
<td>1307</td>
</tr>
<tr>
<td>SF</td>
<td>3095</td>
<td>2934</td>
<td>2799</td>
<td>3053</td>
<td>2142</td>
<td>808</td>
<td>0</td>
<td>379</td>
<td>1235</td>
</tr>
<tr>
<td>LA</td>
<td>2979</td>
<td>2786</td>
<td>2631</td>
<td>2687</td>
<td>2054</td>
<td>1131</td>
<td>379</td>
<td>0</td>
<td>1059</td>
</tr>
<tr>
<td>DENVER</td>
<td>1949</td>
<td>1771</td>
<td>1616</td>
<td>2037</td>
<td>996</td>
<td>1307</td>
<td>1235</td>
<td>1059</td>
<td>0</td>
</tr>
</tbody>
</table>

![Plot of MDS example](image)
Table 1  Flying Mileages Between 10 American Cities

<table>
<thead>
<tr>
<th>Atlanta</th>
<th>Chicago</th>
<th>Denver</th>
<th>Houston</th>
<th>Los Angeles</th>
<th>Miami</th>
<th>New York</th>
<th>San Francisco</th>
<th>Seattle</th>
<th>Washington, DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>587</td>
<td>1212</td>
<td>701</td>
<td>1936</td>
<td>604</td>
<td>748</td>
<td>2139</td>
<td>2182</td>
<td>543</td>
</tr>
<tr>
<td>587</td>
<td>0</td>
<td>920</td>
<td>940</td>
<td>1745</td>
<td>1188</td>
<td>713</td>
<td>1858</td>
<td>1737</td>
<td>597</td>
</tr>
<tr>
<td>1212</td>
<td>920</td>
<td>0</td>
<td>879</td>
<td>831</td>
<td>1726</td>
<td>1631</td>
<td>949</td>
<td>1021</td>
<td>1494</td>
</tr>
<tr>
<td>701</td>
<td>940</td>
<td>879</td>
<td>0</td>
<td>1374</td>
<td>968</td>
<td>1420</td>
<td>1645</td>
<td>1891</td>
<td>1220</td>
</tr>
<tr>
<td>1936</td>
<td>1745</td>
<td>831</td>
<td>1374</td>
<td>0</td>
<td>2339</td>
<td>2451</td>
<td>347</td>
<td>959</td>
<td>2300</td>
</tr>
<tr>
<td>604</td>
<td>1188</td>
<td>1726</td>
<td>968</td>
<td>2339</td>
<td>0</td>
<td>1092</td>
<td>2594</td>
<td>2734</td>
<td>923</td>
</tr>
<tr>
<td>748</td>
<td>713</td>
<td>1631</td>
<td>1420</td>
<td>2451</td>
<td>1092</td>
<td>0</td>
<td>2571</td>
<td>2408</td>
<td>205</td>
</tr>
<tr>
<td>2139</td>
<td>1858</td>
<td>949</td>
<td>1645</td>
<td>347</td>
<td>2594</td>
<td>2571</td>
<td>0</td>
<td>678</td>
<td>2442</td>
</tr>
<tr>
<td>2182</td>
<td>1737</td>
<td>1021</td>
<td>1891</td>
<td>959</td>
<td>2734</td>
<td>2408</td>
<td>678</td>
<td>0</td>
<td>2329</td>
</tr>
<tr>
<td>543</td>
<td>597</td>
<td>1494</td>
<td>1220</td>
<td>2300</td>
<td>923</td>
<td>205</td>
<td>2442</td>
<td>2329</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 1  CMDS of flying mileages between 10 American cities.
Isomap

A Global Geometric Framework for Nonlinear Dimensionality Reduction

J. B. Tenenbaum, V. de Silva and J. C. Langford
Science 290 (5500): 2319-2323, 22 December 2000
ISOMAP

Comes from Isometric feature mapping

Step1: Take a data matrix as input.

Step2: Estimate geodesic distance between any two points by “a chain of short paths”. Approximate the geodesic distance by Euclidean distances.

Step3: Perform MDS
Geodesic: the shortest curve on a manifold that connects two points on the manifold.

Example (3D sphere)
Euclidean distance needs not be a good measure between two points on a manifold. Length of geodesic is more appropriate.
The Swiss-role Dataset
Isomap
ISOMAP Interpolation
ISOMAP Interpolation
ISOMAP Interpolation
ISOMAP Summary

- Build graph from kNN or epsilon neighbors
- Run MDS

- Since MDS is slow, ISOMAP will be very slow.
- Need estimate of k or epsilon.
- Assumes data set is convex (no holes).
Local Linear Embedding

Nonlinear dimensionality reduction by locally linear embedding.
Sam Roweis & Lawrence Saul.
Local Linear Embedding

Assumption: manifold is approximately “linear” when viewed locally. Data: \( X = [x_1, \ldots, x_m] \in \mathbb{R}^{D \times m} \)

\[
W = \arg \min_W \sum_{i=1}^{m} \left\| x_i - \sum_{j=1}^{m} W_{ij} x_j \right\|^2
\]

1. select neighbors (epsilon or kNN)

2. reconstruct with linear weights
Local Linear Embedding

Step 1.

$$W = \arg \min_W \sum_{i=1}^{m} \| x_i - \sum_{j=1}^{m} W_{ij} x_j \|^2$$

Subject to $\sum_j W_{ij} = 1$, $\forall i$,

and $W_{ij} = 0$ if $x_j$ is not neighbor of $x_i$.

Without the constraints the weights that minimize the reconstruction errors are invariant to rotation, rescaling and translation of the data points.
Local Linear Embedding

- Step 2. Given the weights $W$, find the embedded points:

$$[z_1, \ldots, z_m] = \arg \min_{[z_1, \ldots, z_m]} \sum_{i=1}^{m} \| z_i - \sum_{j=1}^{m} W_{ij} z_j \|^2$$

Subject to $\sum_i z_i = 0$

and unit covariance matrix.

The same weights that reconstruct the data points in $D$ dimensions should also reconstruct the points in $d$ dimensions.

The weights characterize the intrinsic geometric properties of each neighborhood.
Locally Linear Embedding

Fit Locally, Think Globally
Unsupervised learning of image manifolds by semidefinite programming.
International Journal of Computer Vision, Volume 70 Issue 1, October 2006, Pages 77 - 90
Maximum Variance Unfolding

Build a graph from kNN or epsilon neighbors.

Given $x_1, \ldots, x_T$ find $y_1, \ldots, y_T$ such that

$$||x_i - x_j|| = ||y_i - y_j||$$ for all $(i, j) \in E$ neighborhood graph

and $\text{var}(y)$ is as large as possible.

Formally,

$$\max_y \text{tr}(\text{cov}(y))$$

s.t. $||x_i - x_j|| = ||y_i - y_j||$ for all $(i, j) \in E$ neighborhood graph

Here $\text{tr}(\text{cov}(y)) = \frac{1}{T} \sum_{i=1}^{T} ||y_i - \bar{y}||^2$, where $\bar{y} = \frac{1}{T} \sum_{i=1}^{T} y_i$
Consider the constraint \( ||x_i - x_j|| = ||y_i - y_j|| \)

From this, we have \( ||x_i - x_j||^2 = ||y_i - y_j||^2 \)

\[
x_i^T x_i - 2x_i^T x_j + x_j^T x_j = y_i^T y_i - 2y_i^T y_j + y_j^T y_j
\]

\[
Q_{ii} - 2Q_{ij} + Q_{jj} = P_{ii} - 2P_{ij} + P_{jj}
\]
Consider the cost function:

\[ \text{cov}(y) = \frac{1}{T} \sum_{i=1}^{T} (y_i - \bar{y})(y_i - \bar{y})^T \]

\[ = \frac{1}{T} \sum_{i=1}^{T} y_i y_i^T - \bar{y}\bar{y}^T \]

\[ = \frac{1}{T} YY^T - \frac{1}{T^2} Y11^T Y^T \]

\[ \text{tr}(\text{cov}(y)) = \frac{1}{T} \text{tr}(YY^T) - \frac{1}{T^2} \text{tr}(Y11^T Y^T) \]

\[ = \frac{1}{T} \text{tr}(YY^T) - \frac{1}{T^2} \text{tr}(Y^T Y11^T) \]

\[ = \frac{1}{T} \text{tr}(Q) - \frac{1}{T^2} \text{tr}(Q11^T) \]
The final problem is a semi-definite problem (SDP):

$$\max_Q \frac{1}{T} tr(Q) - \frac{1}{T^2} tr(Q11^T)$$

s.t. \( Q_{ii} - 2Q_{ij} + Q_{jj} = P_{ii} - 2P_{ij} + P_{jj} \) for all \((i, j) \in E\),

\( Q \succeq 0 \)
Maximum Variance Unfolding

D = 76*101*3
d = 3
N = 400 images
Maximum Variance Unfolding

Swiss roll “unfolded” by maximizing variance subject to constraints that preserve local distances and angles.

The middle snap-shots show various feasible (but non-optimal) intermediate solutions.
Maximum Variance Unfolding
Laplacian Eigenmap

Data: $X = [x_1, \ldots, x_n] \in \mathbb{R}^{l \times n}$

Step 1. Build graph from kNN or epsilon neighbors

Step 2. Choose weights:

$$W_{ij} = \exp\left(-\frac{1}{t} \|x_i - x_j\|^2\right) \text{ if } (i, j) \in E$$

$$W_{ij} = 0 \text{ Otherwise}$$

Special case: $t = \infty$, then $W_{ij} = 1$ if $(i, j) \in E$
Step 3. Assume the graph is connected, otherwise proceed with Step 3 for each connected component:

\[ D_{ii} = \sum_{j=1}^{n} W_{ij} \]

\[ L = D - W \in \mathbb{R}^{n \times n} \text{ Laplacian matrix} \]

**Lemma:** \( L \) is symmetric, positive semi-definite matrix.

Solve the eigenvector problem:

\[ Lf = \lambda Df \]
Laplacian Eigenmap

Solve the eigenvector problem:

\[ Lf = \lambda Df \quad f \in \mathbb{R}^n \]

The first \( m+1 \) smallest eigenvalues:

\[ Lf_0 = \lambda_0 Df_0 \quad 0 = \lambda_0, \quad f_0 = [1, \ldots, 1]^T \in \mathbb{R}^n \]
\[ Lf_1 = \lambda_1 Df_1 \]
\[ \vdots \]
\[ Lf_m = \lambda_m Df_m \quad 0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_m \]

The embedding:

\[ \mathbb{R}^l \ni x_i \rightarrow [f_1(i), \ldots f_m(i)]^T \in \mathbb{R}^m \]
Let us embed the neighborhood graph to 1 dim first.

A reasonable cost function is:

$$\min_{y_1, \ldots, y_n} \sum_{i,j=1}^{n} (y_i - y_j)^2 W_{ij}$$

subject to appropriate constraints to avoid \( y=0 \).

**Lemma**

$$\sum_{i,j=1}^{n} (y_i - y_j)^2 W_{ij} = 2y^T Ly$$

**Proof:**

$$\sum_{i,j=1}^{n} (y_i - y_j)^2 W_{ij} = \sum_{i,j=1}^{n} (y_i^2 + y_j^2 - 2y_iy_j)W_{ij}$$

$$= \sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2 \sum_{i,j} y_i y_j W_{ij} = 2y^T Ly$$
Therefore, our minimization problem is

\[
\min_{y= [y_1, \ldots, y_n]^T} \ y^T Ly
\]

Subject to:

\[y^T Dy = 1\] to fix the scaling.

\[y^T D1 = 0\] to avoid the trivial \(y = [1, \ldots, 1]^T\) solution.

Embedding the neighborhood graph to \(m\) dimension:

\[
\min_{Y^T = [y_1, \ldots, y_n] \in \mathbb{R}^{d \times n}} \ tr(Y^T Ly)
\]

Subject to: \(Y^T D Y = I\)

Solution: \(LY = \lambda D Y\)
Variational Variational Inference for Bayesian Mixtures of Factor Analysers

Zoubin Ghahramani, Matthew J. Beal, NIPS 1999
MANI Matlab demo

Todd Wittman: MANIfold learning demonstration GUI
Contains a couple of methods and examples.

http://www.math.ucla.edu/~wittman/mani

The following results are taken from Todd Wittman
How do we compare the methods?

- Speed
- Manifold Geometry
- Non-convexity
- Curvature
- Corners
- Noise
- Non-uniform Sampling
- Sparse Data
- Clustering

- High-Dimensional Data: Can the method process image manifolds?

- Sensitivity to Parameters
  - K Nearest Neighbors: Isomap, LLE, Hessian, Laplacian, KNN Diffusion
  - Sigma: Diffusion Map, KNN Diffusion
Testing Examples

- Swiss Roll
- Swiss Hole
- Punctured Sphere
- Corner Planes
- 3D Clusters
- Twin Peaks
- Toroidal Helix
- Gaussian
- Occluded Disks

We’ll compare the speed and sensitivity to parameters throughout.
Manifold Geometry

First, let’s try to unroll the Swiss Roll. We should see a plane.
Hessian LLE is pretty slow, MDS is very slow, and ISOMAP is extremely slow. MDS and PCA don’t can’t unroll Swiss Roll, use no manifold information. LLE and Laplacian can’t handle this data. Diffusion Maps could not unroll Swiss Roll for any value of Sigma.
Non-Convexity

Can we handle a data set with a hole?
Swiss Hole: Can we still unroll the Swiss Roll when it has a hole in the middle?
Only Hessian LLE can handle non-convexity. ISOMAP, LLE, and Laplacian find the hole but the set is distorted.
Manifold Geometry

Twin Peaks: fold up the corners of a plane. LLE will have trouble because it introduces curvature to plane.
PCA, LLE, and Hessian LLE distort the mapping the most.
Gaussian: We can randomly sample a Gaussian distribution. We increase the curvature by decreasing the standard deviation. Coloring on the z-axis, we should map to concentric circles.
For std = 1 (low curvature), MDS and PCA can project accurately. Laplacian Eigenmap cannot handle the change in sampling.
For \( \text{std} = 0.4 \) (higher curvature), PCA projects from the side rather than top-down. Laplacian looks even worse.
For std = 0.3 (high curvature), none of the methods can project correctly.
Corner Planes: We bend a plane with a lift angle $A$. We want to bend it back down to a plane.

If $A > 90$, we might see the data points written on top of each other.
For angle $A=75$, we see some distortions in PCA and Laplacian.
For $A = 135$, MDS, PCA, and Hessian LLE overwrite the data points. Diffusion Maps work very well for $\Sigma < 1$. LLE handles corners surprisingly well.
Clustering

A good mapping should preserve clusters in the original data set.

3D Clusters: Generate M non-overlapping clusters with random centers. Connect the clusters with a line.
For $M = 3$ clusters, MDS and PCA can project correctly. Diffusion Maps work well with large Sigma. LLE compresses each cluster into a single point. Hessian LLE has trouble with the sparse connecting lines.
For M=8 clusters, MDS and PCA can still recover. Diffusion Maps do quite well. LLE and ISOMAP are decent, but Hessian and Laplacian fail.
Can the method handle changes from dense to sparse regions? Toroidal Helix should be unraveled into a circle parametrized by t.

We can change the sampling rate along the helix by changing the exponent $R$ on the parameter $t$ and we can add some noise.
With no noise added, ISOMAP, LLE, Laplacian, and Diffusion Map are correct. MDS and PCA project to an asterisk. What’s up with Hessian and KNN Diffusion?
Adde noise to the Helix sampling.
LLE cannot recover the circle.
ISOMAP emphasizes outliers more than the other methods.
When the sampling rate is changed along the torus, Laplacian starts to mess up and Hessian is completely thrown off. Hessian LLE code crashed frequently on this example. Diffusion maps handle it quite well for carefully chosen Sigma=0.3.
Of course, we want as much data as possible. But can the method handle sparse regions in the data? Punctured Sphere: the sampling is very sparse at the bottom and dense at the top.
Only LLE and Laplacian get decent results.

PCA projects the sphere from the side. MDS turns it inside-out. Hessian and Diffusion Maps get correct shape, but give too much emphasis to the sparse region at the bottom of the sphere.
All of the examples so far have been 3D. But can the data handle high-dimensional data sets, like images?

Disks: Create 20x20 images with a disk of fixed radius and random center.

We should recover the centers of the circles.
LLE crashed on high-dimensional data set. Number of images was not high enough, but ISOMAP did a very good job.
Occluded Disks

We can add a second disk of radius $R$ in the center of every image.
Both LLE and Hessian crashed, possibly # points is too small. Laplacian failed completely. Is ISOMAP the best for high-dimensional data?
Sensitivity to Parameters

When the number of points is small or the data geometry is complex, it is important to set K appropriately, neither too big nor small.

But if the data set is dense enough, we expect K around 8 or 10 to suffice.

For Diffusion Maps, the method is very sensitive to the Sigma in Gaussian kernel. Varies from example to example.
Diffusion Map Sigma depends on manifold.

Helix

Clusters

Sigma = 10

Sigma = 0.2
<table>
<thead>
<tr>
<th></th>
<th>MDS</th>
<th>PCA</th>
<th>ISOMAP</th>
<th>LLE</th>
<th>Hessian</th>
<th>Laplacian</th>
<th>Diffusion Map</th>
<th>KNN Diffusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Very slow</td>
<td>Extremely fast</td>
<td>Extremely slow</td>
<td>Fast</td>
<td>Slow</td>
<td>Fast</td>
<td>Fast</td>
<td>Fast</td>
</tr>
<tr>
<td>Infers geometry?</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>MAYBE</td>
<td>MAYBE</td>
</tr>
<tr>
<td>Handles non-convex?</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>MAYBE</td>
<td>YES</td>
<td>MAYBE</td>
<td>MAYBE</td>
<td>MAYBE</td>
</tr>
<tr>
<td>Handles non-uniform sampling?</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>MAYBE</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Handles curvature?</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>MAYBE</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Handles corners?</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Clusters?</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Handles noise?</td>
<td>YES</td>
<td>YES</td>
<td>MAYBE</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Handles sparsity?</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>MAYBE</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Sensitive to parameters?</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>VERY</td>
</tr>
</tbody>
</table>

**So what have you learned, Dorothy?**
Some Notes on using MANI

- Hard to set $K$ and $\Sigma$ just right.

- MDS and ISOMAP are very slow.

- Hessian LLE is pretty slow. Since Hessian needs a dense data set, this means it takes even longer when the # points is large.

- Occluded Disks is 400-dimensional data, which takes a long time and a lot of data points to correctly map.

- Matlab GUIs seem to run better on PC than Linux.
Credits

M. Belkin,
P. Niyogi,
Todd Wittman
Thanks for your attention 😊