Algorithms for manifold learning

Lawrence Cayton

lcayton@cs.ucsd.edu

Talk outline

1. Motivation: managing high-dimensional data.
2. Dimensionality reduction: a classic approach.
3. The manifold learning model.
4. Algorithms proposed for manifold learning.
5. Comparisons.
6. Open Problems.

High-dimensional data

Many recent machine learning applications require dealing with massive datasets.

- *Computer vision*: points $\leftrightarrow$ images; its dimensionality $=$ the number of pixels (often several thousand).
- *Document classification*: points $\leftrightarrow$ documents; dimensionality $=$ the number of distinct words in a corpus.

High-dimensional data

Many recent machine learning applications require dealing with massive datasets.

- *Computer vision*: points $\leftrightarrow$ images; its dimensionality $=$ the number of pixels (often several thousand).
- *Document classification*: points $\leftrightarrow$ documents; dimensionality $=$ the number of distinct words in a corpus.

Difficulties of performing function estimation or classification on high-dimensional data:

- computational complexity
- global optima are elusive
- some features will be irrelevant or misleading.

Dimensionality Reduction

We deal with these problems by using a dimensionality reduction procedure to preprocess data.

Dimensionality reduction is also useful for

- visualization
- revealing the underlying forces in the dataset.

Hopefully, the dimensionality reduction procedure will retain the “important” features in the data.

We will discuss methods for intrinsically low dimensional data.

Linear dimensionality reduction – PCA

Idea: Find the direction along which the data varies most:

$$\max_{\|v\|=1} \text{var}(Xv) = \cdots = \max_{\|v\|=1} v^\top X^\top X v$$

... and project the data onto this direction.

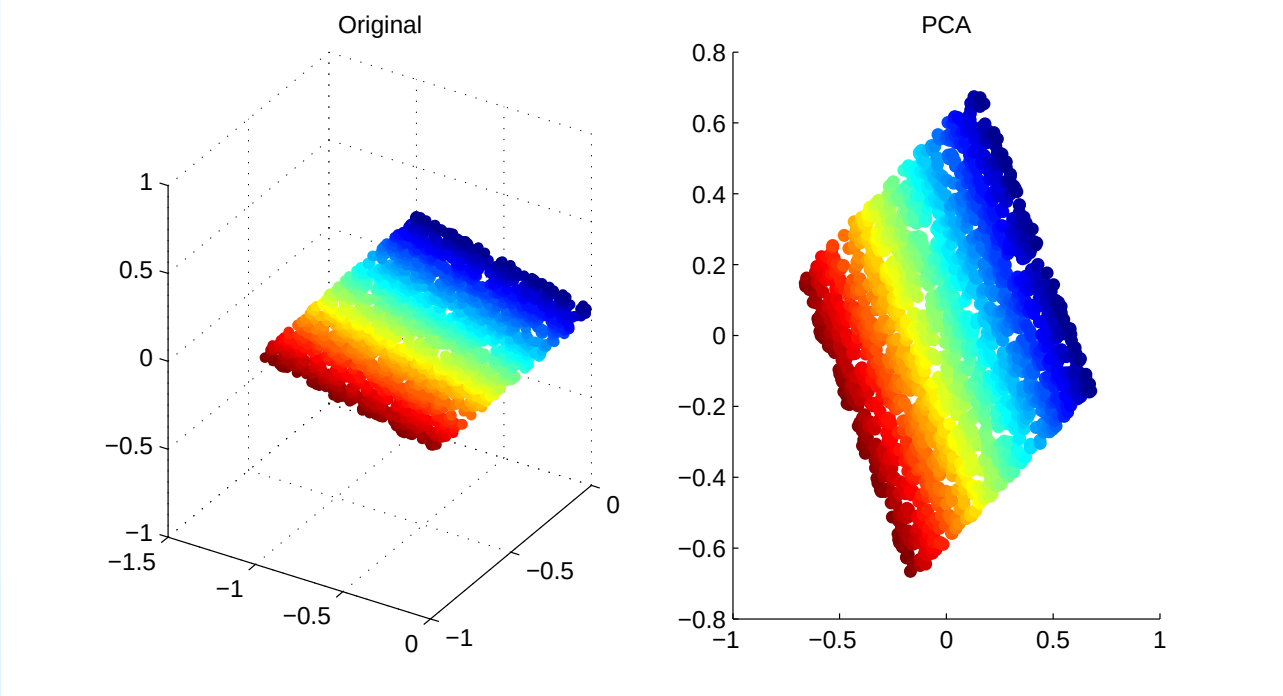
By Rayleigh's theorem, the maximizer is given by the top eigenvector of $X^\top X$.

Generally, the top d eigenvectors give the d-dimensional subspace along which the data varies the most and the eigenvalues reveal the relative weight of each direction.

Linear dimensionality reduction – PCA 2

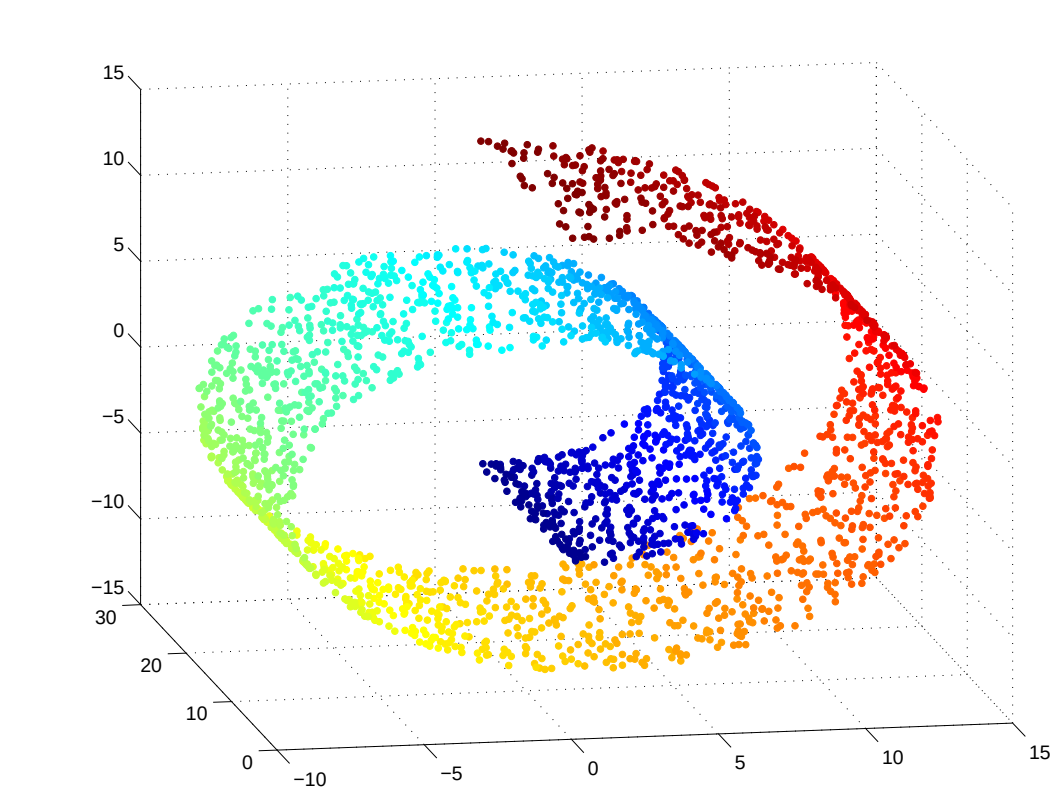
The result:

- If the data is distributed uniformly in $\Re^D$, PCA just rotates the data.
- But, if the data lies along a d-dimensional subspace of $\Re^D$, PCA will find that subspace.



Swiss Roll

Great, but what about this:



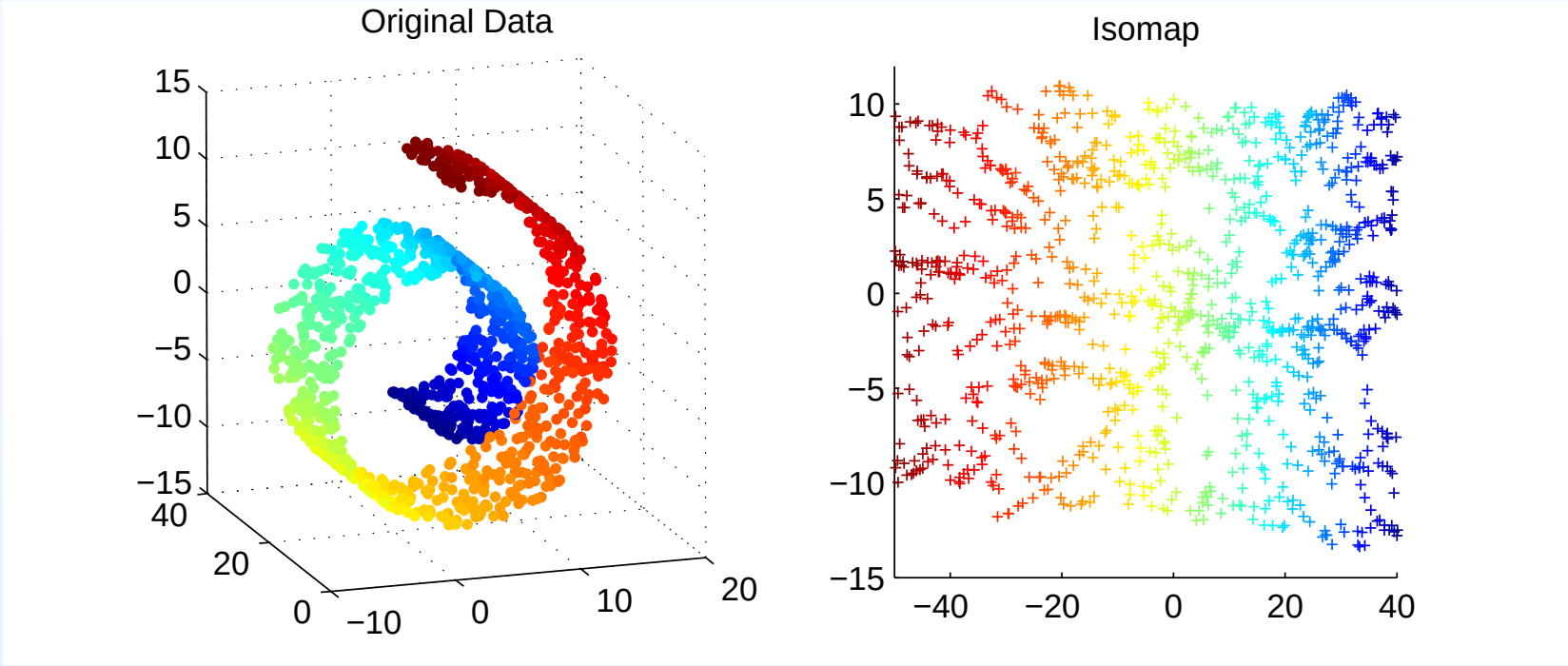
Still intuitively 2-d, but definitely not a 2-d subspace.

PCA can’t handle it.

Manifold Learning

PCA works on subspaces, but not on non-linear manifolds (*e.g* the swiss roll).

The goal of manifold learning: generalize the behavior of PCA on subspaces to manifolds.



Manifold Learning

We formalize the problem as follows.

Input: We are given points $x_1, \ldots, x_n \in \Re^D$.

- These points lie on a d-dimensional manifold M.
- M can be described by a single smooth coordinate chart $f : M \to \Re^d$.

Goal: Find $y_1, \ldots, y_n \in \Re^d$, where $y_i = f(x_i)$.

The image of f is referred to as the parameter space.

Overview of manifold learning algorithms

Lots of algorithms have been proposed – we'll discuss a few key ones.

There is no "best" algorithm.

A few similarities across algorithms:

- Like PCA, all are spectral algorithms that find global optima.
- Each algorithm looks at small neighborhoods of points to build up a model of the manifold.
- Accordingly, the neighborhood size is a parameter of each of these algorithms (k or ϵ).
- Each algorithm is based on leveraging some aspect of a manifold that is invariant across embeddings.

Isomap (isometric feature mapping)

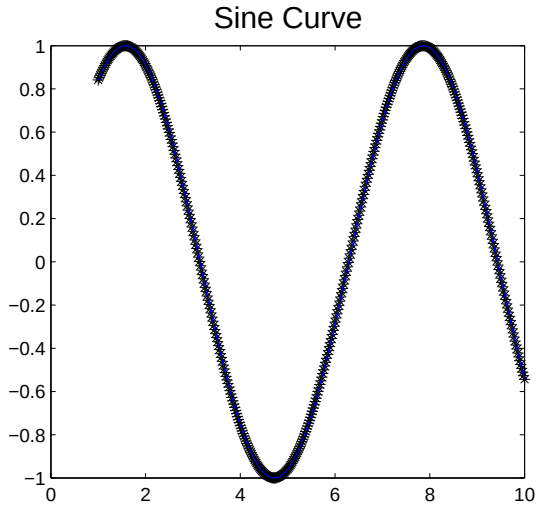
Insights:

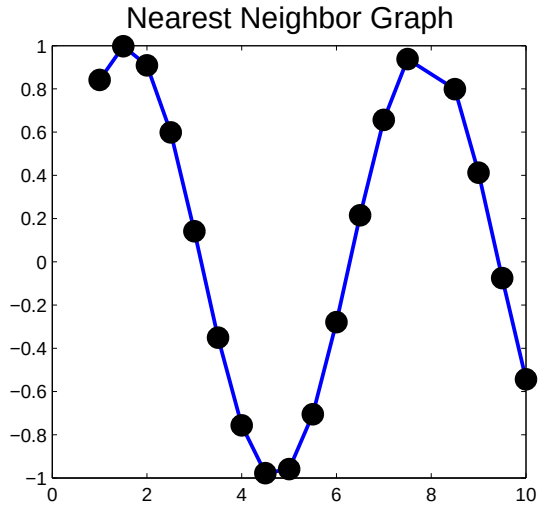
- If the manifold is isometrically embedded then

$$\|y_i - y_j\| = G(x_i, x_j)$$

(*i.e.* low-d Euclidean distance = Geodesic distance).

- Geodesic distances can be approximated using a nearest neighbor graph.





Isomap - algorithm

Algorithm:

1. Estimate geodesic distances:
   (a) Create the k-nearest neighbor graph, weighted by Euclidean distances.
   (b) Calculate shortest paths between all points.
2. Find a configuration of points whose *Euclidean* interpoint distances equal these geodesic distances.

Step 2 is handled by classical Multidimensional Scaling.

(Classical) Multidimensional Scaling

Problem: Given $G \in \Re^{n \times n}$, find a configuration of points such that $\|y_i - y_j\|^2 \approx G_{ij}$.

Fact: If G is Euclidean, then $B \stackrel{\text{def}}{=} -\frac{1}{2}HGH$ is a Gram (inner product) matrix ....

Moreover, $B = YY^\top$, where Y is a configuration with interpoint distances G.

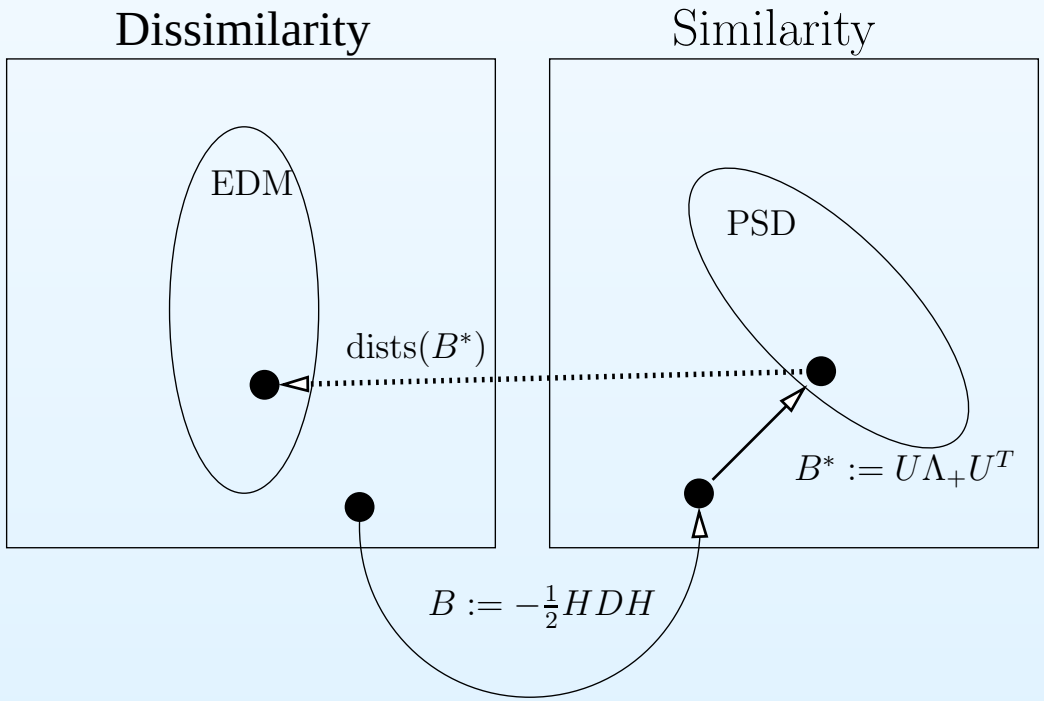
Algorithm:

1. Compute $B := -\frac{1}{2}HGH$.
2. Spectrally decompose $B = U\Lambda U^\top$.
3. Set $[\Lambda_+]_{ij} := \max\{\Lambda_{ij}, 0\}$.
4. Return $Y := U\Lambda_+^{1/2}$.

MDS - a different view

Algorithm:

1. Compute $B := -\frac{1}{2}HDH$.
2. Spectrally decompose $B = U\Lambda U^\top$.
3. Set $[\Lambda_+]_{ij} := \max\{\Lambda_{ij}, 0\}$.
4. Return $Y := U\Lambda_+^{1/2}$.



Isomap

In the continuum limit, Isomap will find the low-dimensional parameters of the manifold assuming:

- the manifold is isometrically embedded (smoothly), compact, and well sampled.
- The parameter space is convex. [questionable]

One of two provably “correct” algorithms for the task.

Isomap

In the continuum limit, Isomap will find the low-dimensional parameters of the manifold assuming:

- the manifold is isometrically embedded (smoothly), compact, and well sampled.
- The parameter space is convex. [questionable]

One of two provably "correct" algorithms for the task.

Notes:

- Automatically figures out the dimensionality of the manifold.
- Time complexity: $O(n^3)$ – prohibitive for large datasets.
- Variants:
  - C-Isomap learns conformal maps instead of isometric ones.
  - Landmark Isomap is a fast approximation to Isomap.

Local methods

Isomap computed an embedding using all interpoint geodesic distances.

Local methods use cost functions that only control the placement of points with respect to their neighbors.

We'll look at

- Locally Linear Embedding
- Laplacian Eigenmaps
- Hessian Locally Linear Embedding,

and see how they are related.

Locally Linear Embedding (LLE)

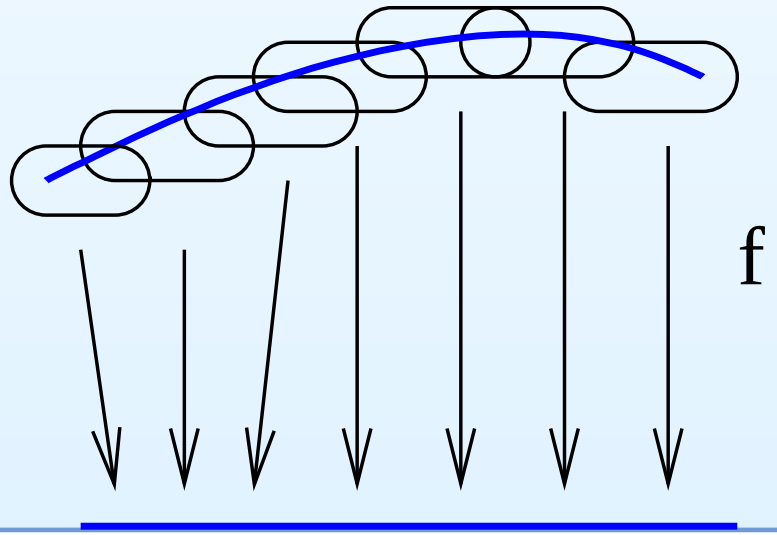
Intuition: Think of a manifold as a collection of small, overlapping linear coordinate patches. The coordinate chart will be roughly linear on these patches.

- Model these linear patches.
- Characterize the local geometry of these patches.
- Find a map from these patches to $\Re^d$ that maintains this characterization and is as linear as possible.

Locally Linear Embedding (LLE)

Intuition: Think of a manifold as a collection of small, overlapping linear coordinate patches. The coordinate chart will be roughly linear on these patches.

- Model these linear patches.
- Characterize the local geometry of these patches.
- Find a map from these patches to $\Re^d$ that maintains this characterization and is as linear as possible.



Locally Linear Embedding - details

1. Model each point as a convex combination of its neighbors:

$$\min_W \left\| \sum_i \left(x_i - \sum_{j \in N(i)} W_{ij} x_j \right) \right\|.$$

2. Construct a d-dimensional embedding Y minimizing

$$\min_{y_1,\ldots,y_n} \left\| \sum_i \left(y_i - \sum_{j \in N(i)} W_{ij} y_j \right) \right\|.$$

Locally Linear Embedding - details

1. Model each point as a convex combination of its neighbors:

$$\min_W \left\| \sum_i \left(x_i - \sum_{j \in N(i)} W_{ij} x_j \right) \right\|.$$

2. Construct a d-dimensional embedding Y minimizing

$$\min_{y_1,\ldots,y_n} \left\| \sum_i \left(y_i - \sum_{j \in N(i)} W_{ij} y_j \right) \right\|.$$

- W is invariant to rotations, translations, and scalings.
- Therefore, if the mapping is linear on x_i's coordinate patch, and x_i can be perfectly reconstructed from its neighbors, there will be zero error on y_i.

Locally Linear Embedding - wrapup

Both steps can be solved relatively efficiently; step 2 requires a spectral decomposition of $(I - W)^\top (I - W)$.

Conjectured to learn conformal maps.

Problems:

- No convergence results known.
- d must be known *a priori*.

Laplacian Eigenmaps

Idea:

- Define a local similarity measure on points.
- Find an embedding that preserves that local similarity measure –*i.e.* nearby points should be mapped to nearby points.

Laplacian Eigenmaps

Idea:

- Define a local similarity measure on points.
- Find an embedding that preserves that local similarity measure –*i.e.* nearby points should be mapped to nearby points.

W – the local similarity measure – is defined as:

$$W_{ij} = \begin{cases} 1 & \text{if } x_j \in N(i) \\ 0 & \text{otherwise.} \end{cases}$$

or alternatively

$$W_{ij} = \begin{cases} e^{-\|x_i - x_j\|/2\sigma^2} & \text{if } x_j \in N(i) \\ 0 & \text{otherwise} \end{cases}$$

Laplacian Eigenmaps

Cost function:

$$\min_{y_1,\ldots,y_n} \sum_{ij} W_{ij} \|y_i - y_j\|^2 = \cdots = \min_{Y} \mathrm{tr}(Y^\top L Y)$$

where $L \stackrel{\mathrm{def}}{=} D - W$ [graph Laplacian], $D_{ii} \stackrel{\mathrm{def}}{=} \sum_j W_{ij}$.

Constraint: $Y^\top D Y = I$ – prevents a rank-deficient solution.

Minimized by the bottom (non-constant) d eigenvectors of $Ly = \lambda D y$.

Laplacian Eigenmaps

Cost function:

$$\min_{y_1,\ldots,y_n} \sum_{ij} W_{ij} \|y_i - y_j\|^2 = \cdots = \min_{Y} \mathsf{tr}(Y^\top L Y)$$

where $L \stackrel{\text{def}}{=} D - W$ [graph Laplacian], $D_{ii} \stackrel{\text{def}}{=} \sum_j W_{ij}$.

Constraint: $Y^\top D Y = I$ – prevents a rank-deficient solution.

Minimized by the bottom (non-constant) d eigenvectors of $Ly = \lambda D y$.

Problems:

- No known convergence results.
- d must be known *a priori*.

Laplacian Eigenmaps

Another view: Laplacian Eigenmaps finds a mapping from local neighborhood patches to $\Re^d$ with small average Laplacian $\mathcal{L}$, where $\mathcal{L}$ is an operator defined as

$$\mathcal{L} \stackrel{\text{def}}{=} \sum_{i=1}^{d} \frac{\partial^2 f}{\partial x_i^2}$$

...this interpretation feels like LLE...

LapEig - LLE connection

It turns out that under certain assumptions,

LLE is finding a map minimizing $\frac{1}{2}\mathcal{L}^2$.

- LLE's map: the eigenfunctions of $\frac{1}{2}\mathcal{L}^2$
- LapEig's map: the eigenfunctions of $\mathcal{L}$

.. which are equal!

LapEig - LLE connection

It turns out that under certain assumptions,

LLE is finding a map minimizing $\frac{1}{2}\mathcal{L}^2$.

- LLE's map: the eigenfunctions of $\frac{1}{2}\mathcal{L}^2$
- LapEig's map: the eigenfunctions of $\mathcal{L}$

.. which are equal!

But, the algorithms are shown to be equivalent only in expectation and under some assumptions.

The equivalence has not been demonstrated empirically or exploited.

Bottom line: these algorithms are still considered different.

Hessian LLE

Idea: Replace the Laplacian with a Hessian.

Algorithm sketch:

1. Find a coordinate system for each linear patch.
2. Derive a discrete Hessian estimator for each patch.
3. Find the mapping of the input to d-dimensions that minimizes these Hessian estimators on average.

The third step is given by a sparse eigenvalue problem.

Hessian LLE

This algorithm is provably correct assuming that

- the manifold is locally isometrically embedded and
- the underlying parameter space is connected.

These assumptions are less restrictive than Isomap's – impressive!

But, Isomap has some finite-sample error bounds, Hessian LLE does not (they aren't known, anyways).

No one seems to be using it.

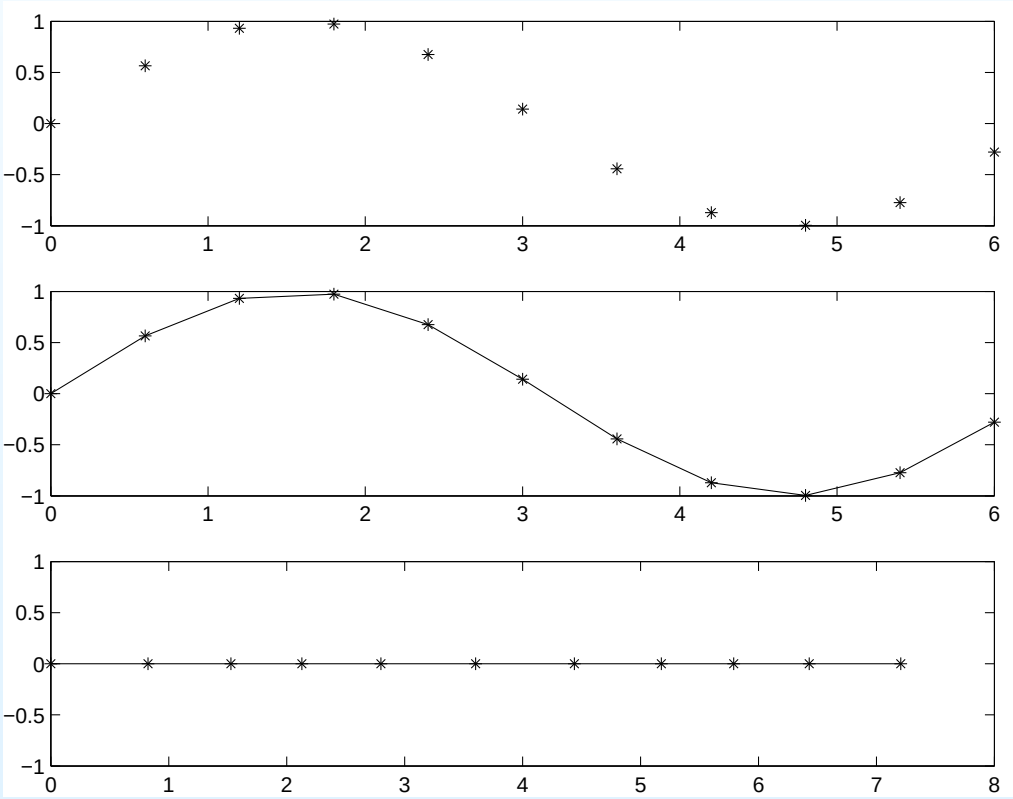
There are a few other LLE-related algorithms, as well.

Semidefinite Embedding

Based on a physical intuition:

1. Connect each point to its neighbors with a rigid rod.
2. Pull the manifold as far apart as possible.

For example:



Semidefinite Embedding - the details

Based on a semidefinite program (SDP) – essentially a linear program where the variables are constrained to form a semidefinite matrix.

Algorithm sketch

1. Use a SDP (specified later) to find a Gram matrix $B = YY^\top$ based on the input.
2. Spectrally decompose B: $B = U\Lambda U^\top$.
3. Return $Y := U\Lambda^{1/2}$.

Semidefinite Embedding - the SDP

Constraints:

- Want neighboring distances to be preserved in the embedding:

$$\begin{aligned} \forall i \forall j \in N(i) \ \ \|x_i - x_j\|^2 &= \|y_i - y_j\|^2. \\ &= B_{ii} + B_{jj} - 2B_{ij}. \end{aligned}$$

- Regularization: center the embedding ($\sum_{ij} B_{ij} = 0$).

Semidefinite Embedding - the SDP

Constraints:

- Want neighboring distances to be preserved in the embedding:

$$\begin{aligned} \forall i \forall j \in N(i) \;\; \|x_i - x_j\|^2 &= \|y_i - y_j\|^2. \\ &= B_{ii} + B_{jj} - 2B_{ij}. \end{aligned}$$

- Regularization: center the embedding ($\sum_{ij} B_{ij} = 0$).

Want to maximize the distance between non-neighboring points, giving objective:

$$\begin{aligned} \max \sum_{ij} \|y_i - y_j\|^2 &= \max \sum_{ij} B_{ii} + B_{jj} - 2B_{ij} \\ &= \max \mathsf{tr}(B). \end{aligned}$$

Semidefinite Embedding

The program:

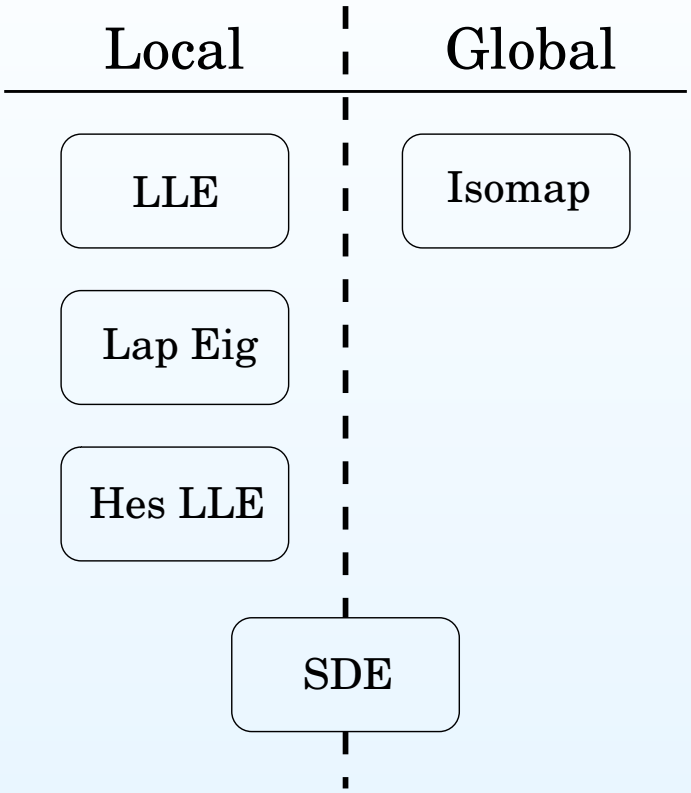
maximize	$\text{tr}(B)$
subject to	$\sum_{ij} B_{ij} = 0$;
	$B_{ii} + B_{jj} - 2B_{ij} = \|x_i - x_j\|^2$
	(for all neighboring x_i, x_j);
	$B \succeq 0$.

Semidefinite Embedding

Notes:

- Like Isomap, unlike local methods, provides automatic estimate of dimensionality.
- Solving a SDP takes a lot of time – this algorithm cannot be used for datasets with $n \geq 2000$ (or so) at all...
- but there is a fast approximation known as ℓSDE.
- No known convergence guarantees.
- Somewhat mysterious: why does maximizing interpoint distance reduce the dimensionality?

Comparisons: local vs. global



Local methods: embeddings tend to be incorrect for distant points.

Isomap: embeddings tend to be erroneous locally.

Comparisons: embedding type

Algorithm	**Mapping**
C-Isomap	conformal
Isomap	isometric
Hessian LLE	isometric
Laplacian Eigenmaps	unknown
Locally Linear Embedding	conformal
Semidefinite Embedding	isometric

Comparisons: time complexity

Local methods: scale well because of sparsity.

Isomap: spectral decomposition is not sparse, so does not scale well to large problem sizes.

Semidefinite embedding: Even worse – can't handle $n \geq 2000$.

Open Problems

1. Neighborhood sizes:
   - All of these algorithms require a neighborhood size parameter.
   - Different choices often lead to very different results.

   How can we pick neighborhoods appropriately?

Open Problems

1. Neighborhood sizes:
   - All of these algorithms require a neighborhood size parameter.
   - Different choices often lead to very different results.

   How can we pick neighborhoods appropriately?

2. A circle is a one-dimensional manifold, but cannot be handled by these methods.

   Intuitively, we need to cut it somewhere.

   What do we do with manifolds that are not homeomorphic to $\Re^d$?

Open Problems – fundamentals

1. Families of images have been found to lie along a manifold... what about everything else?

   To what extent can real world data be modeled as points on a manifold?

Open Problems – fundamentals

1. Families of images have been found to lie along a manifold... what about everything else?

   To what extent can real world data be modeled as points on a manifold?

2. Evaluation of these algorithms is difficult.
   - Theoretical evaluation.
     - Need finite sample size guarantees
     - Good learning algs are often bad in theory.
   - Experimental evaluation.
     - Figuring out if data actually lies on a manifold is as hard as learning the manifold.
     - What is a relevant cross-section of manifolds to learn?

   A sound evaluation methodology is needed.

Open Problems - the big one

With these fundamentals answered, we can address:

How successful are these algorithms at the dimensionality reduction task?