Non-Linear Dimension Reduction (in L_2)

Non-Linear Dimension Reduction: $R^D \to R^d$

Goal: Find a low-dim. Non-linear map that preserves the relevant information

- Application dependent
- *Different definitions yield different techniques*

Some canonical techniques…

- IsoMap (Isometric Mapping)
- LLE (Locally Linear Embedding)
- LE (Laplacian Eigenmaps)
- MVU (Maximum Variance Unfolding)
- kPCA (kernel PCA)
- RP (random projections – *linear!*)
- NE (Nash Embedding)
- t-SNE (t-distributed Stochastic Neighbor Embedding)
- VAE (Variational AutoEncoders)

Motivation

While the data is represented in high dimensions, it usually only has a few degrees freedom.

Examples:

- Scanned images of handwritten characters

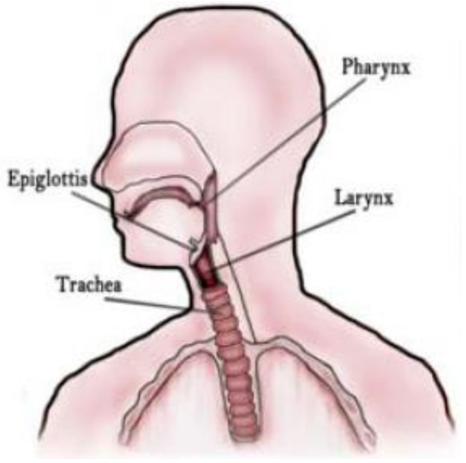
  Representation: 28x28 pixels (= 784 dim)

  Few attributes such as: shape, tilt and cursiveness

  govern the actual written character.

- Natural processes with physical constraints - speech

  Few anatomical characteristics, such as size of the vocal chords, pressure applied, etc. govern the speech signal.



The Manifold Hypothesis

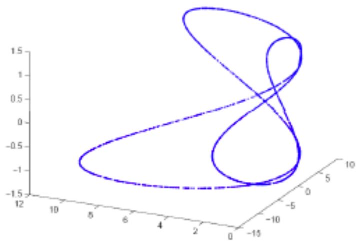
So how can we model such data? (high representation dim., but few degrees)

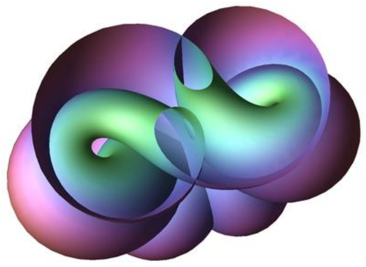
properties we want:

- Small variations in the latent degrees of freedom doesn't change the representation too much
- Globally, the data doesn't conform to any assumed structure

A mathematical model: **Manifold**

- Geometric objects that are constrained to be locally flat
- Since there is no global constraint, they are can be thought of as smooth non-linear objects





Modelling data Manifolds

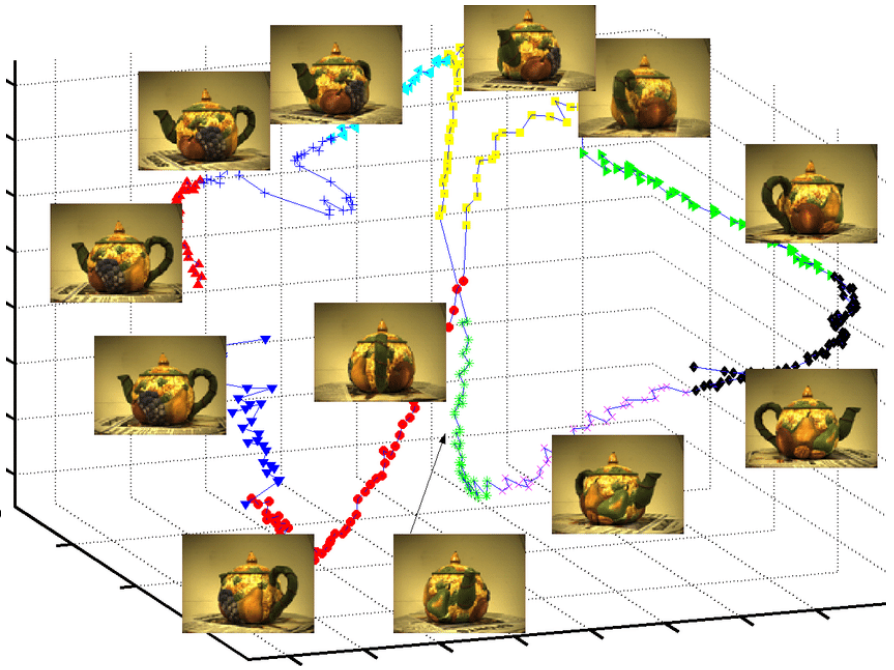
We can use the manifolds to such data!

An example dataset:

Image of a teapot from different angles

How data is distributed in the pixel space

- Closed 1D loop (manifold)



Key Questions:

Given samples from this manifold,

- How can we represent data low-dim?
- How can we do prediction effectively?

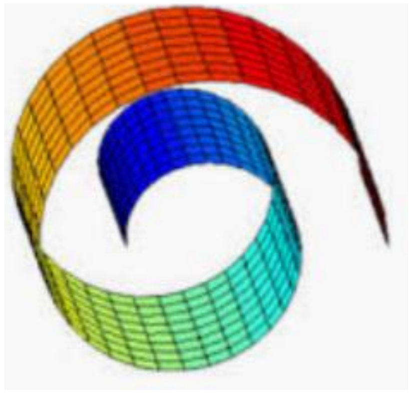
IsoMap (Isometric Mapping)

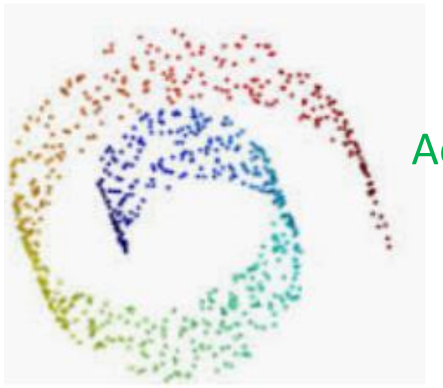
Goal: Find a low-dim. nonlinear map that improves that preserves the interpoint geodesic distances!

Idea:

- if somehow geodesics are known, we can use something like MDS to do the embedding!
- Key question then: how to estimate the geodesics from data samples

Underlying data manifold (unobserved)





Actual, perhaps noisy, data samples (observed)

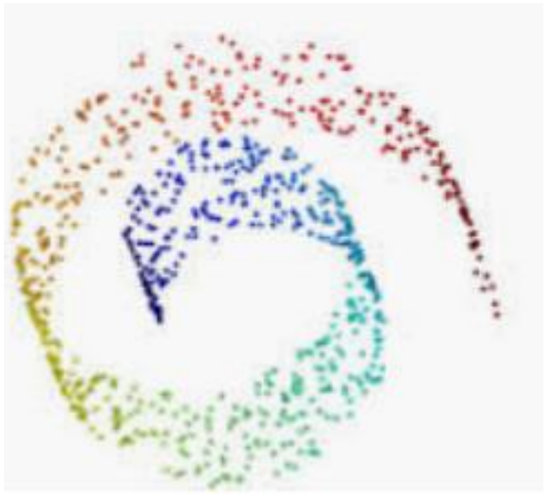
How to estimate the geodesics from RHS?

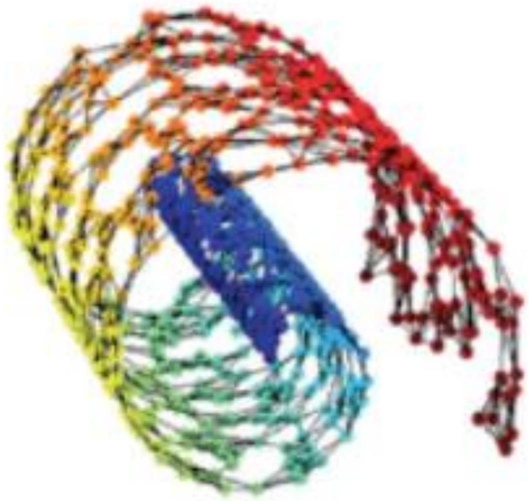
IsoMap (Isometric Mapping)

Estimating geodesics from data samples

Idea:

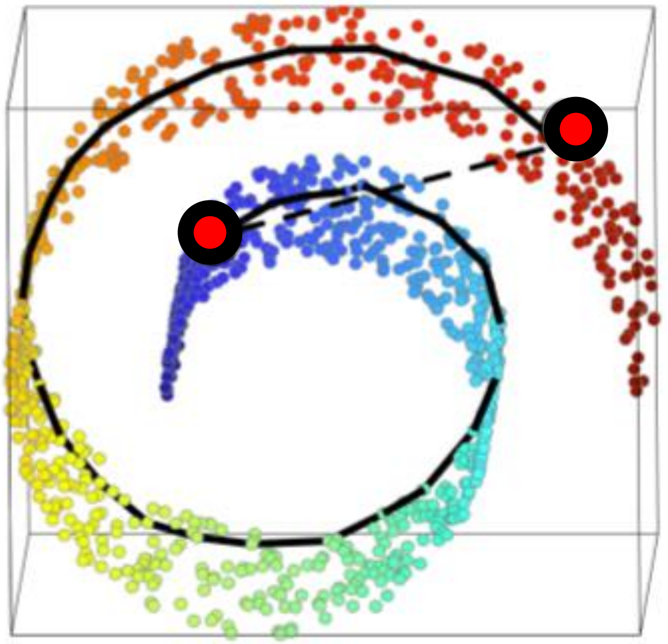
- Create a k-nearest neighbor graph, and compute (all-pairs) shortest paths between the data samples!



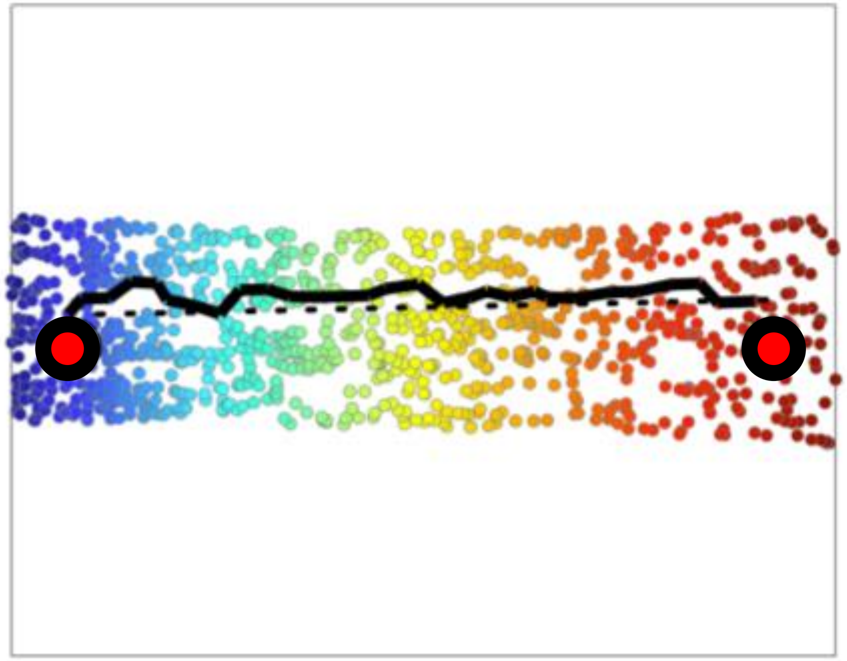


Now simply run MDS on the estimated geodesic distance matrix!!

IsoMap (Isometric Mapping)



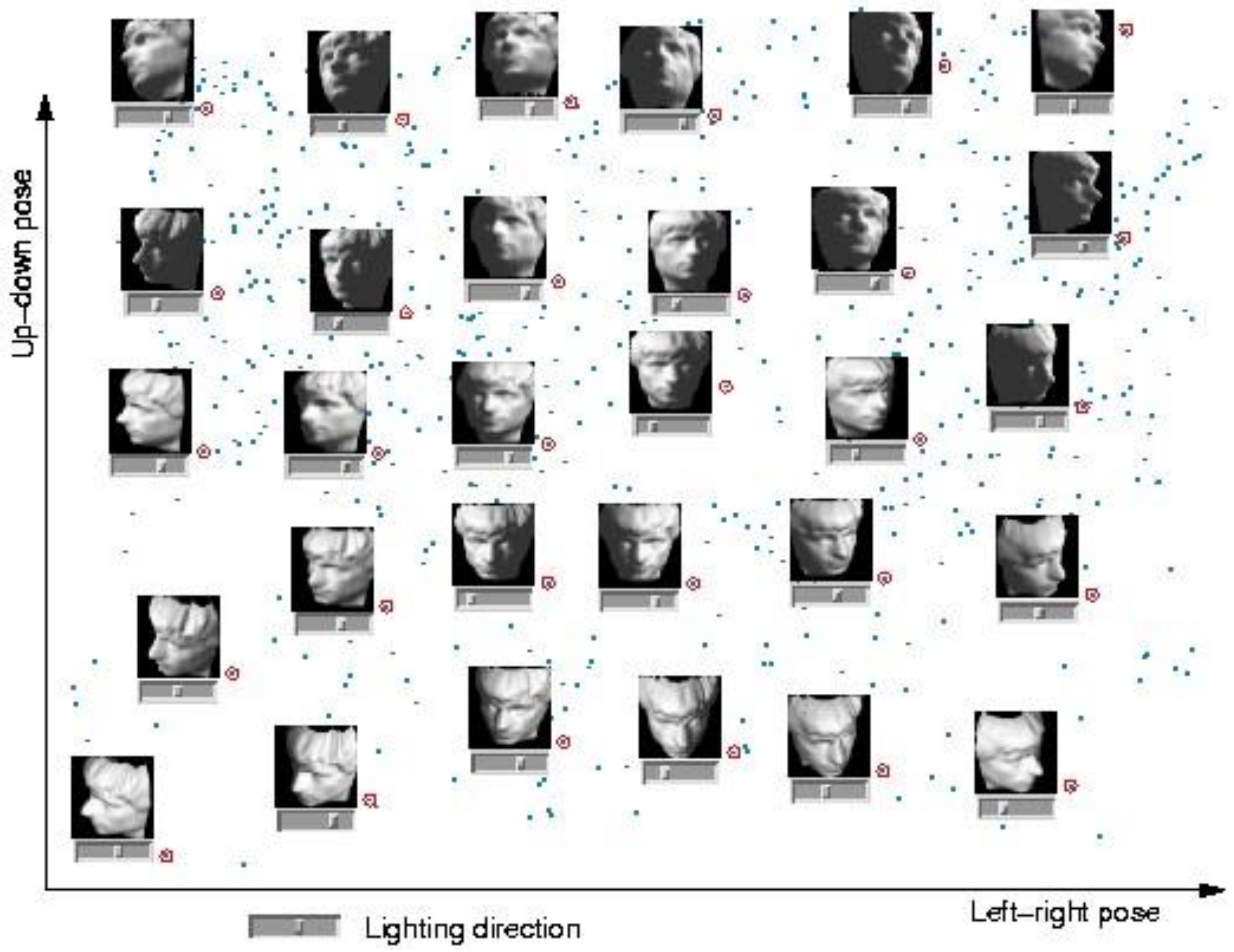
An estimated geodesic via shortest path



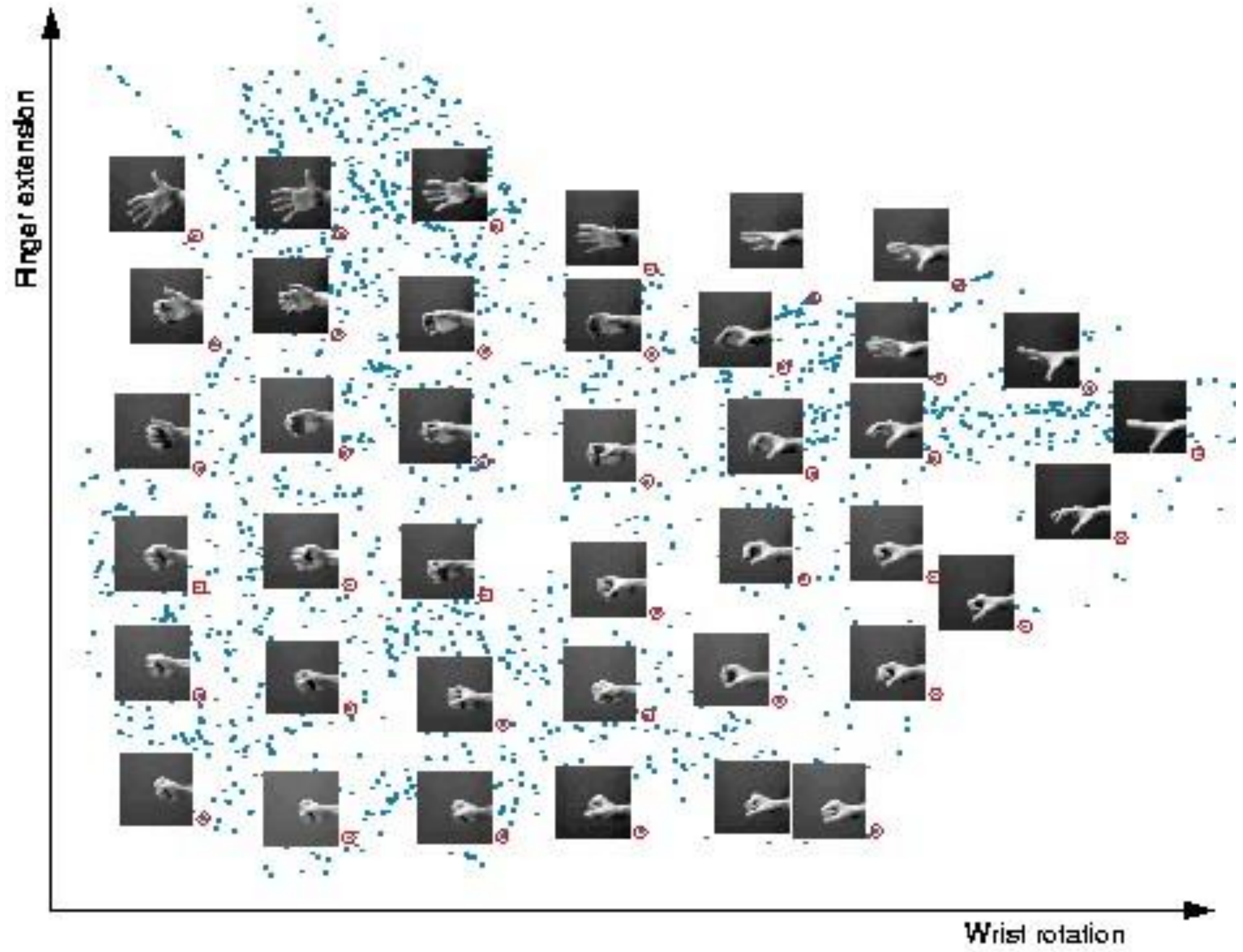
Embedding in the low-dimensional space via MDS (which preserves the input geodesic distances)

Theorem: Under suitable distributions over the underlying manifold, as neighbor-hood size r → 0, number of samples n → ∞, nr → ∞, can show that the shortest path → geodesic path.

IsoMap (Isometric Mapping)



IsoMap (Isometric Mapping)

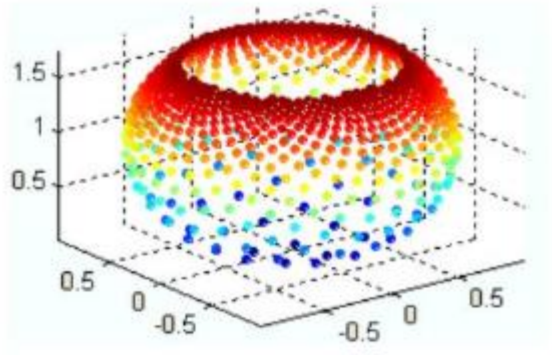


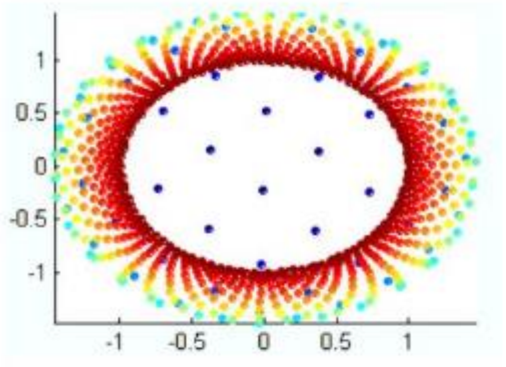
IsoMap (Isometric Mapping)

Seems like IsoMap does an excellent job, isn't then manifold embedding problem solved?

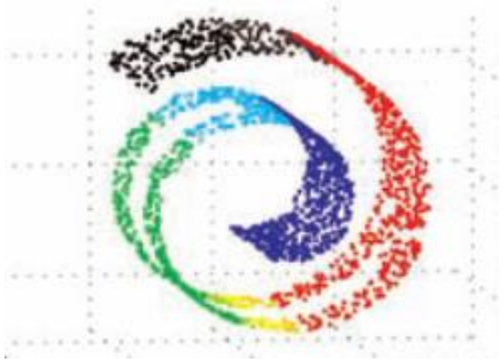
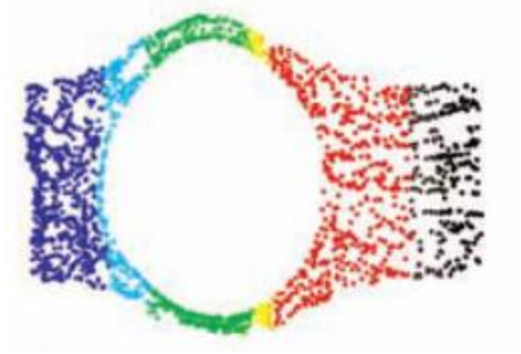
Nope!

The manifold needs to be globally isometric to some low-dim Euclidean space





The manifold needs to be 'geodesically convex'

Locally Linear Embedding (LLE)

Goal: Find a low-dim. map that preserves the "local geometry"

umm… what is "local geometry"???

Since a manifold is locally linear, one can define "local geometry" as how a specific data point is linearly related to its neighbors.

Then, one can compute an embedding Y of the given input X such that the locally linear relationships between neighbors is approximately preserved!

How to formalize this?

Locally Linear Embedding (LLE)

Given: Input data $X \in R^{D \times n}$, embedding dimension d

Step 1.

$$\min_W \Phi(W) = \sum_{i=1}^{n} \left|\left| x_i - \sum_{j \in N(i)} W_{ij} x_j \right|\right|^2$$

find linear relationship between neighbors

$$\text{s.t. } \forall i \sum_j W_{ij} = 1. \qquad w_{ij} = 0 \text{ where } j \notin N(i)$$

Step 2.

$$\min_Y \Psi(Y) = \sum_{i=1}^{n} ||y_i - \sum_{j \in N(i)} W_{ij} y_i||^2$$

find embedded data that respects the local linear relationships

$$\text{s.t. } YY^T = I$$

How do you do these optimizations efficiently??

Locally Linear Embedding (LLE)

$$\min_W \Phi(W) = \sum_{i=1}^{n} \left|\left| x_i - \sum_{j \in N(i)} W_{ij} x_j \right|\right|^2$$

$$\text{s.t. } \forall i \ \sum_j W_{ij} = 1.$$

$$w_{ij} = 0 \text{ where } j \notin N(i)$$

Step 1.
Cost of the ith point:

$$\Phi(W_{i:}) = ||x_i - \textstyle\sum_{j \in N(i)} W_{ij} x_j||^2.$$

We can re-write it as…

$$\begin{bmatrix} x_i & x_i & \dots & x_i \end{bmatrix} = x_i e^T$$

$$\Rightarrow x_i = x_i e^T w_i \text{ where } \sum_j w_{ij} = 1$$

$$\sum_{j \in N(i)} W_{ij} x_j = N_i W_i$$

Therefore:

$$\min_{W_i} \Phi(W_{i:}) = \min_{W_i} ||X_i e^T W_i - N_i W_i||^2$$

Notation:

- D x k Neighbor matrix

$$N_i = \begin{bmatrix} x_{j_1} & x_{j_2} & \dots & x_{j_k} \end{bmatrix}$$

- k x 1 weight vector

$$W_i = \begin{bmatrix} W_{ij_1} \\ W_{ij_2} \\ \vdots \\ W_{ij_k} \end{bmatrix}$$

- K x 1 ones vector

$$e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Locally Linear Embedding (LLE)

$$\min_{W_i} \Phi(W_{i:}) = \min_{W_i} ||X_i e^T W_i - N_i W_i||^2$$

$$= \min_{W_i} W_i^T \underbrace{(X_i e^T - N_i)^T (X_i e^T - N_i)}_{\text{known matrix } =: G} W_i$$

$$\min_{W_i} W_i^T G W_i$$

$$\text{s.t. } e^T W_i = 1$$

Can be solved as a constrained optimization via Lagrange multipliers!

$$L(W_i, \lambda) = W_i^T G W_i - \lambda(e^T W_i - 1)$$

$$\frac{dL}{dW_i} = 2GW_i - \lambda e = 0$$

$$2Gw_i = \lambda e$$

Known λ $\quad W_i = G^{-1}\frac{\lambda}{2}e$

Pick any λ which $\Sigma W_i = 1$

Locally Linear Embedding (LLE)

Step 2.
We can re-write it as…

$$y_i = yI_{:i} \qquad \sum_{j \in N(i)} W_{:j} y_i = YW_{:i}$$

$$\min_Y \Psi(Y) = \sum_{i=1}^{n} ||y_i - \sum_{j \in N(i)} W_{ij} y_i||^2$$

$$\text{s.t. } YY^T = I$$

Notation:

$$Y = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}$$

$$W = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & W_{ij} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

$$M = (I - w)(I - W)^T$$

Therefore, the optimization becomes:

$$\min_Y \sum_{i=1}^{n} ||YI_{:i} - YW_{:i}||^2$$

$$= \min_Y ||YI - YW||_F^2$$

$$= \min_Y tr((I - W)^T Y^T Y (I - W))$$

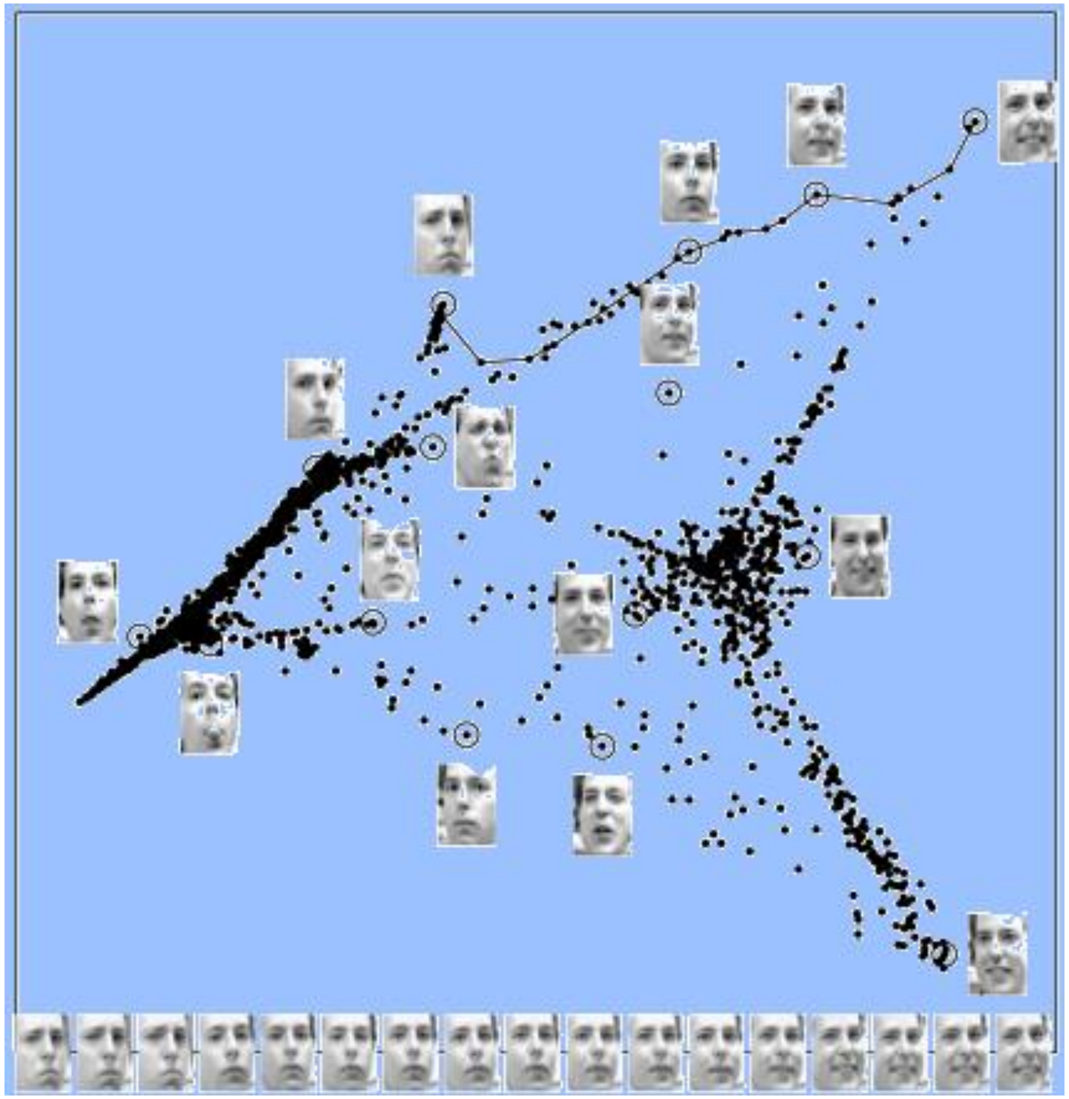
$$= \min_Y tr(Y(I - w)(I - W)^T Y^T)$$

$$\min_Y \mathrm{tr}(YMY^T)$$

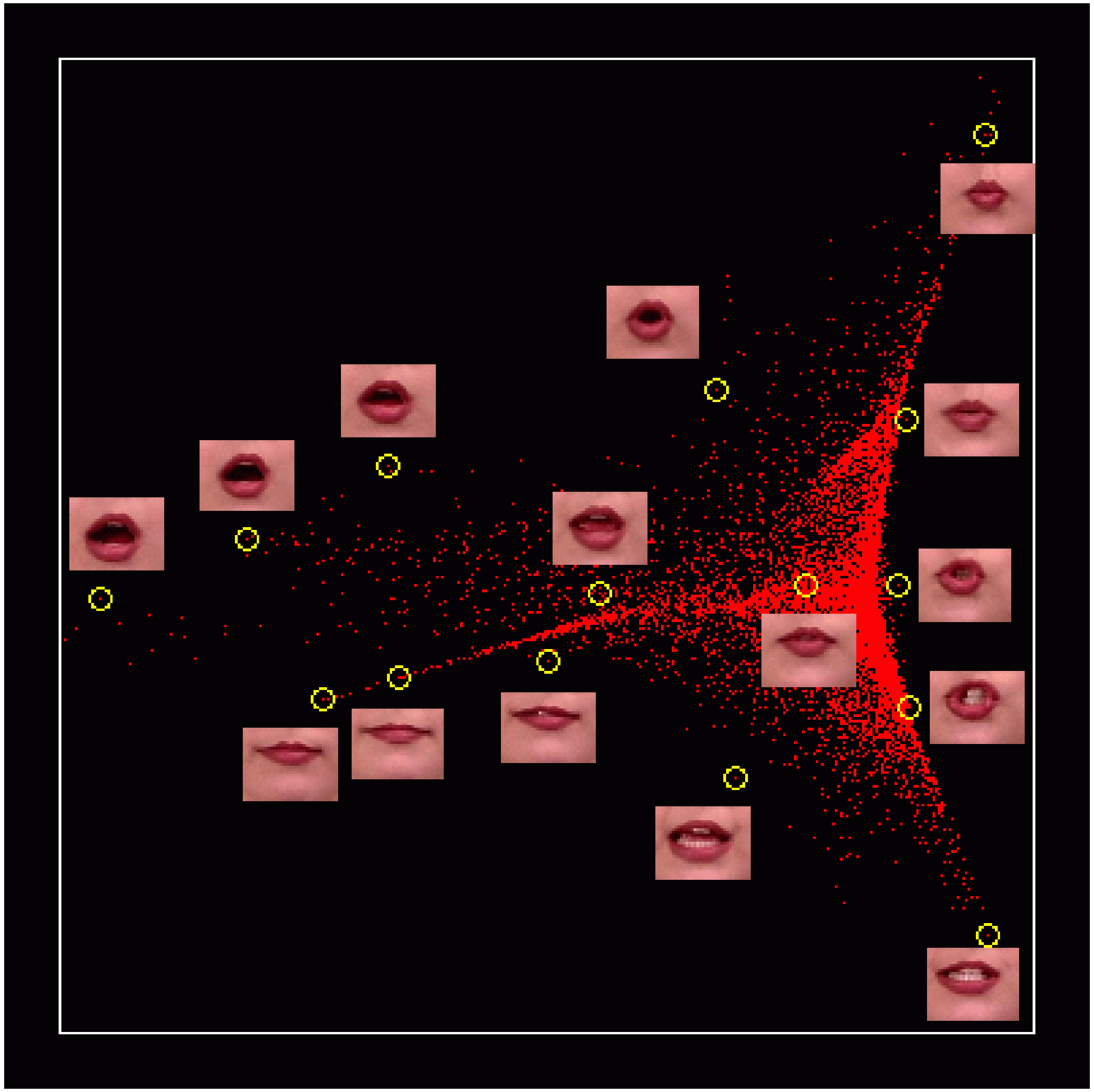
$$\text{s.t. } Y^T Y = I$$

Eigendecomposition of M, top d eigenvectors!

Locally Linear Embedding (LLE)



Locally Linear Embedding (LLE)



Laplacian Eigenmaps (LE)

[Belkin and Niyogi '03]

Goal: Find a low-dim. map that preserves the "local geometry"

Can define local geometry as similarity between points in local neighborhoods!

Define similarity between points x_i, x_j as $W_{ij} := e^{-\|x_i - x_j\|^2/2\sigma^2}$

Can optimize for embedding points $y_1,\ldots,y_n$ as

graph Laplacian

$$\min_Y \sum_{i,j} W_{ij}||y_i - y_j||^2 \quad \text{s.t. } Y^TY = I$$

Equivalent to

$$\min_Y \operatorname{tr}(Y^T \bigcirc\!\!\!\!L\, Y) \quad \text{s.t. } Y^TY = I$$

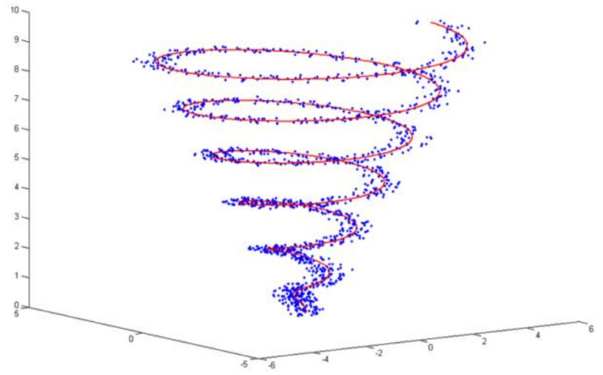
Observation:
if x_i x_j are far, W_{ij} is close to zero and y_i and y_j can map anywhere

Eigendecomposition of L, bottom d eigenvectors!

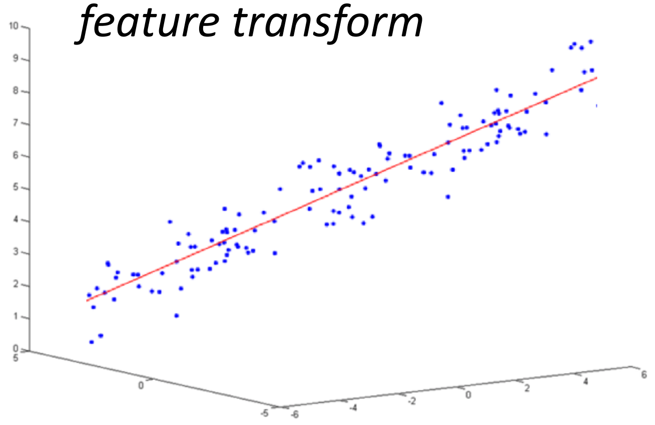
Kernel-PCA (k-PCA)

Can we kernelize PCA to implicitly do a linear dimension reduction (ie PCA) in a non-linear feature space?

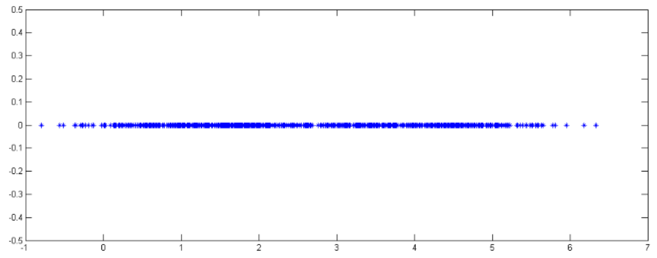
High dimensional manifold data



Explicit non-linear feature transform



PCA



Kernel-PCA
(done implicitly)

Kernel-PCA (k-PCA)

How to kernelize PCA?

Observation:
PCA directions are the principal eigenvectors of the (centered) matrix XX^T
But… we want the inner product X^TX

Let $X = USV^T$

- Since $XX^T = US^2U^T$, we are interested in finding how any datapoint x is projected onto the vectors U, ie the projection U^tx

If data is in a feature space $\phi(X)$

- Problem: cannot compute U (no efficient way to compute outer product)
- If dot product can be computed efficiently, then easy to compute V and S
- Since $U = \phi(X)\, V\, S^{-1}$
- For any datapoint $\phi(x)$, its projection $U^T \phi(x) = \boxed{S^{-1} V \phi(X) . \phi(x)}$

can be computed efficiently!

Kernel-PCA (k-PCA)

We can view many of the manifold embedding methods as a special case of kernel PCA (with specific choice of the kernel)!

- PCA: $K = X^T X$ (linear kernel)

- Classical-MDS: $K = -\frac{1}{2} H D^{eucl} H$, where H is the centering matrix
  (Euclidean) distance gets converted into inner product matrix K

- Isomap: $K = -\frac{1}{2} H D^{geodesic} H$
  Basically MDS with geodesic distances

- Locally Linear Embedding: $A := (I - W)(I - W)^T$; $K = L^{-1}$ or $K = (\lambda_{max} I - A)$
  W are the learned locally linear weights from LLE

- Laplacian Eigenmap: $K = L^{-1}$ or $K = (\lambda_{max} I - L)$
  L is the graph Laplacian used in LE

Kernel-PCA (k-PCA)

Observation

- Many manifold embedding methods assume a specific kernel form

Idea: why don't we directly ***learn the kernel*** which yields a good embedding?

Maximum Variance Unfolding (MVU) (aka Semi-Definite Embedding (SDE))

Maximum Variance Unfolding (MVU)

In order to find an optimal kernel for non-linear embedding, we need to define what we want the embedding to do.

Goal: Find a low-dim. map that preserves the "local geometry"

local geometry = distances between neighboring points!

So, how to learn a kernel that preserve the local distances?

For points x_i and x_j in any neighborhood

$$\begin{aligned} ||x_i - x_j||^2 &= ||\phi(x_i) - \phi(x_j)||^2 \\ &= K_{ii} + K_{jj} - 2K_{ij} \end{aligned}$$

- K needs to be PSD
- $\Sigma_{ij} K_{ij} = 0$

Can formulate the optimization as follows…

Maximum Variance Unfolding (MVU)

[Weinberger and Saul ’04]

Maximize$_K$ tr(K) *Maximize the spread of points*

Constraints:

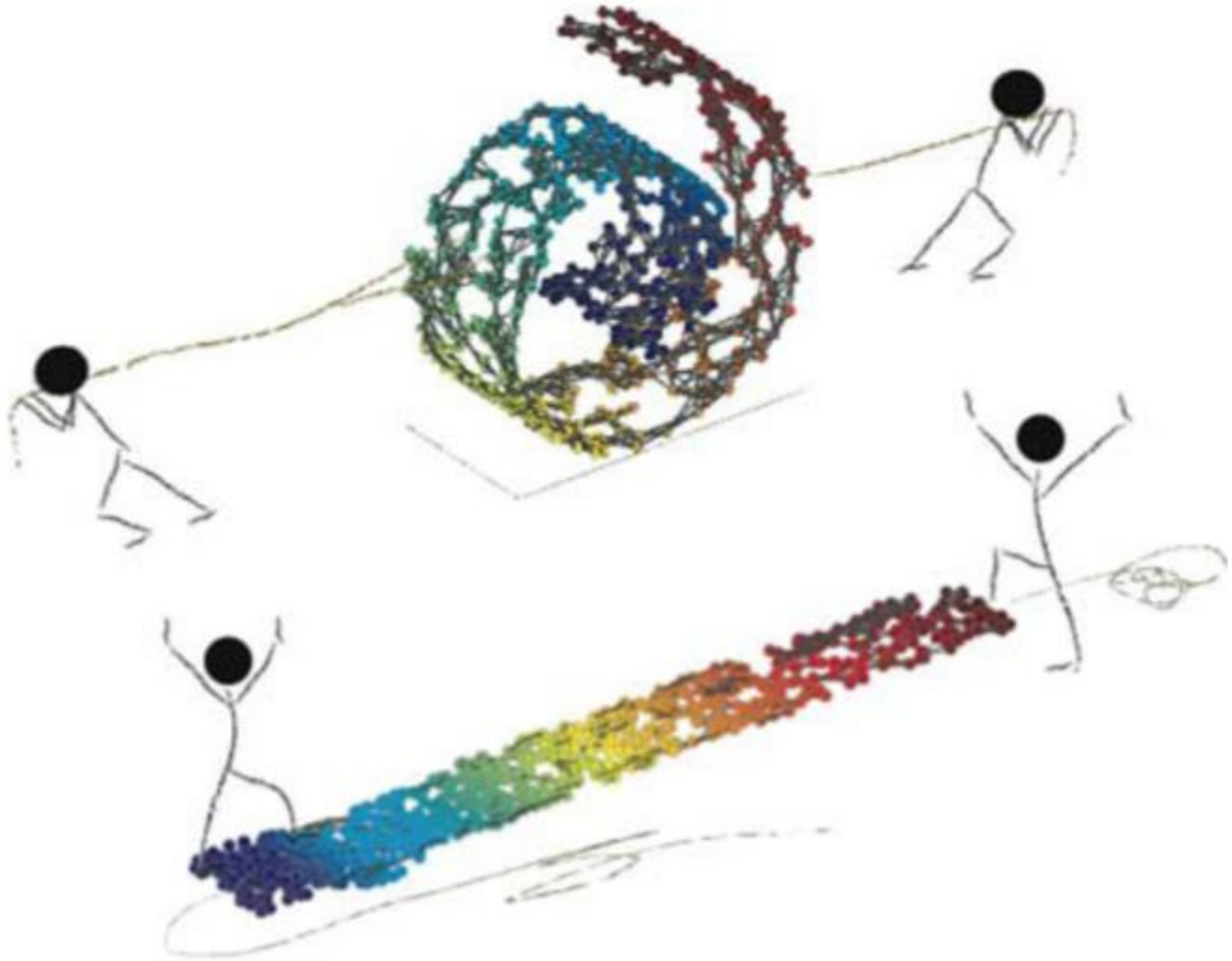
$$K_{ii} + K_{jj} - 2K_{ij} - ||x_i - x_j||^2 \text{ if } i \text{ and } j \text{ are neighbors.}$$

$$\sum_{ij} K_{ij} = 0$$

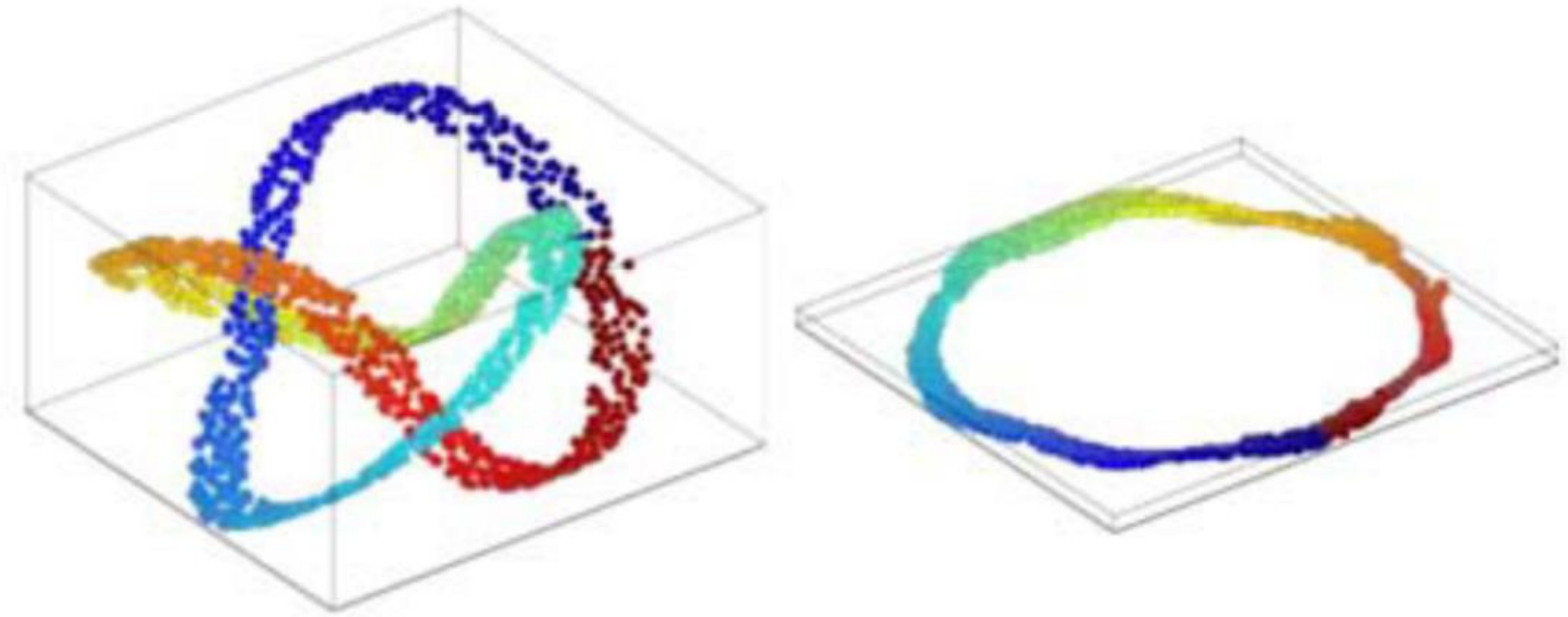
$$K \succeq 0$$

Can be solved by any SDP solver

Maximum Variance Unfolding (MVU)



Maximum Variance Unfolding (MVU)



Stochastic Neighbor Embedding (SNE)

Goal: Find a low-dim. map that preserves the "local geometry"

local geometry = similarity between points in local neighborhoods

Similar goal to LE, BUT a drastically different solution!

Idea:

Model the neighborhood structure/information as a probability distribution, then find a low-dimensional mapping that matches the same distribution!

Notation:

- $x_1,\ldots,x_n$ given high dim. data (given)
- $y_1,\ldots,y_n$ mapped low dim. Representation (to be learned)
- $p_{j|i}$ = probability of x_j being the neighbor of x_i (computed from data)
- $q_{j|i}$ = probability of y_j being the neighbor of y_i (to be matched to $p_{j|i}$)

Stochastic Neighbor Embedding

[Hinton and Roweis ’03]

Stochastic Neighbor Embedding approach:

Probability model for high-dim input data

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\tau_i^2)}{\sum_{k\neq i} exp(-||x_i - x_k||^2/2\tau_i^2)}$$

Meta parameter controlling the neighborhood size

Probability model for low-dim mapped data

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k\neq i} exp(-||y_i - y_k||^2)}$$

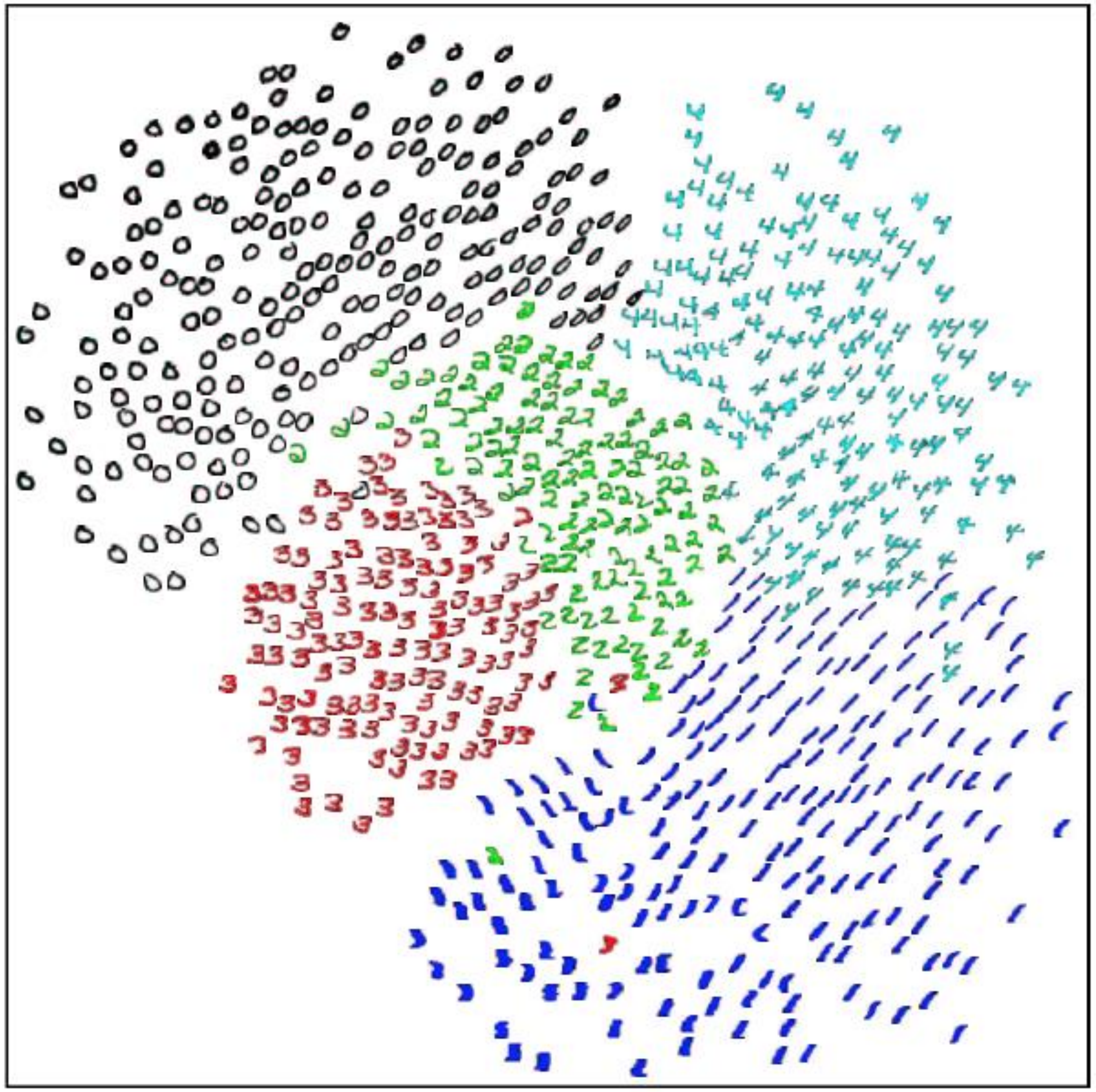
y’s are the variables that need to be learned

Key optimization: Maximize the similarity between the distributions

minimize$_y$ $\sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$

Highly non-convex, just do gradient descent and settle with the local optimal solution

Stochastic Neighbor Embedding



The individual class clusters are well all together producing an effective visualization

But the clusters are NOT well separated

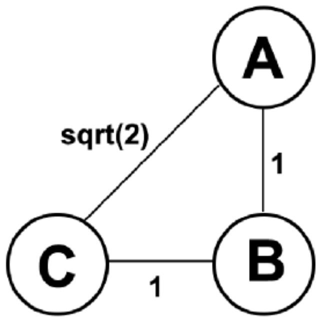
The issue: “crowding problem”

t-distributed Stochastic Neighbor Embedding

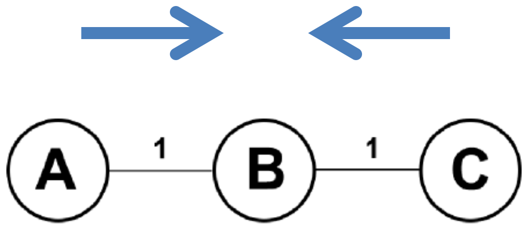
The crowding problem:

High dimensional data is being cramped into a low dimensional space, to match the probabilities, the clusters can “crowd” together

Consider three clusters/points A, B, C



Organization in high dimensions



Organization in low dimensions

Because of the gaussian-type neighborhood structure in low dimensions, large distance between A and C will be **penalized** a lot causing them to be mapped close (ie crowd) to each other

t-distributed Stochastic Neighbor Embedding

[Van der Maaten and Hinton ’08]

Solution to the crowding problem

Idea: instead of using a thin-tailed Gaussian in the lower dimensions, we can use a heavier-tailed distribution, e.g. student’s t-distribution!

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Symmetrize the high dimensional neighborhood distribution

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

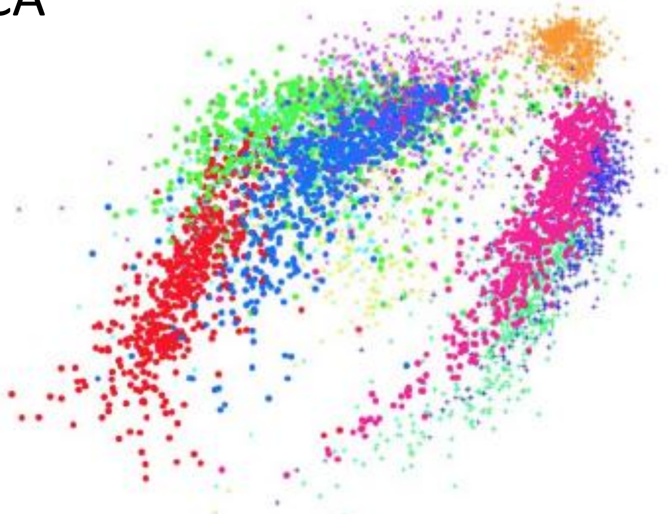
Use the heavier tailed student’s t-distribution

Final optimization:

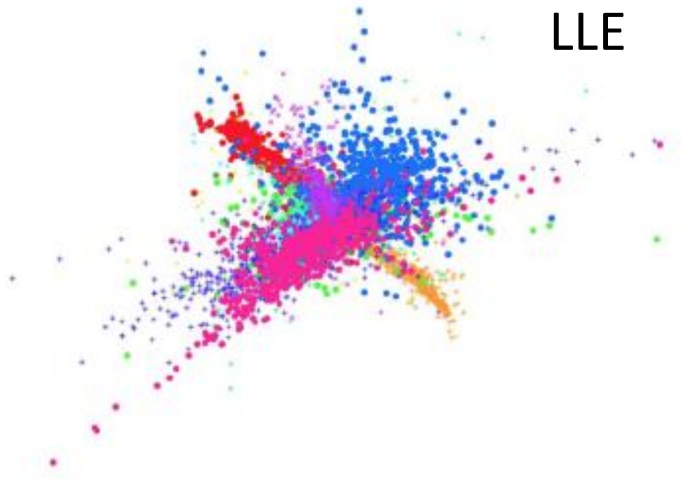
minimize$_y$ $$\sum_i KL(P_i||Q_i) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE

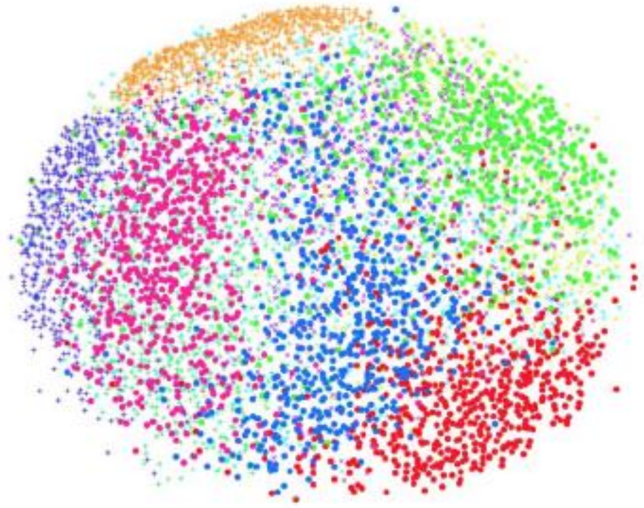
PCA



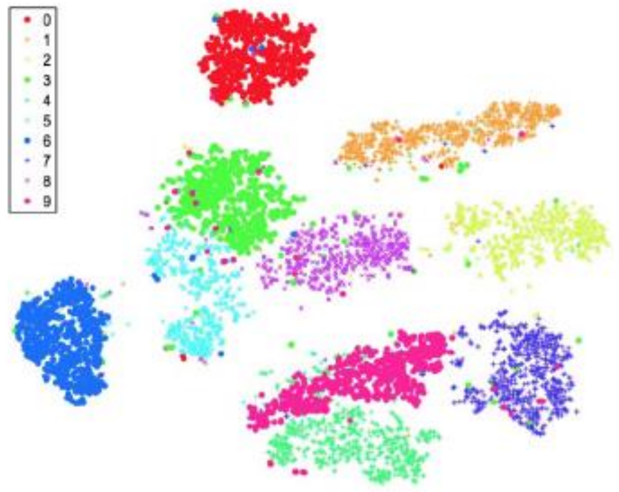
LLE



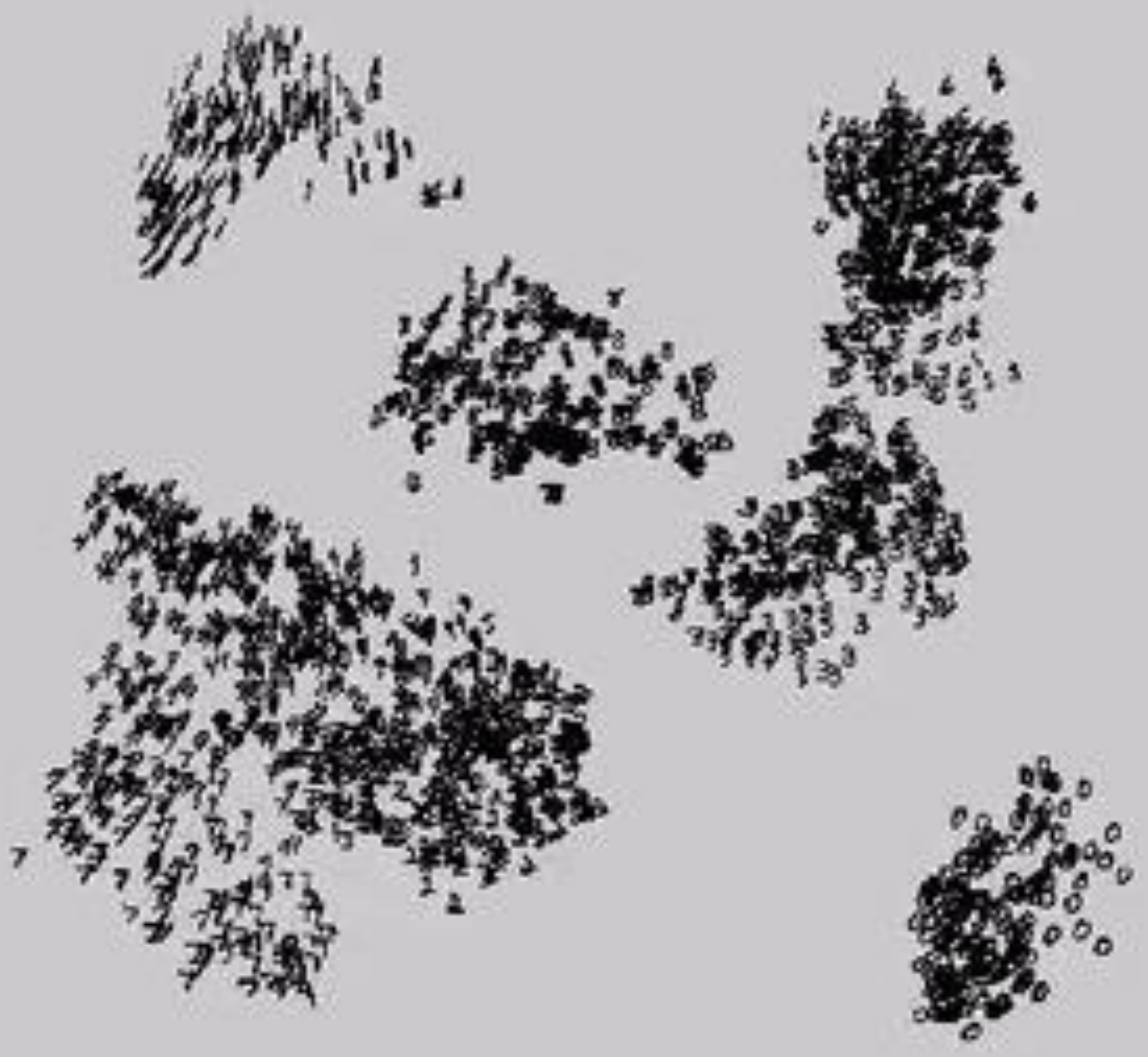
Sammon mapping



t-SNE

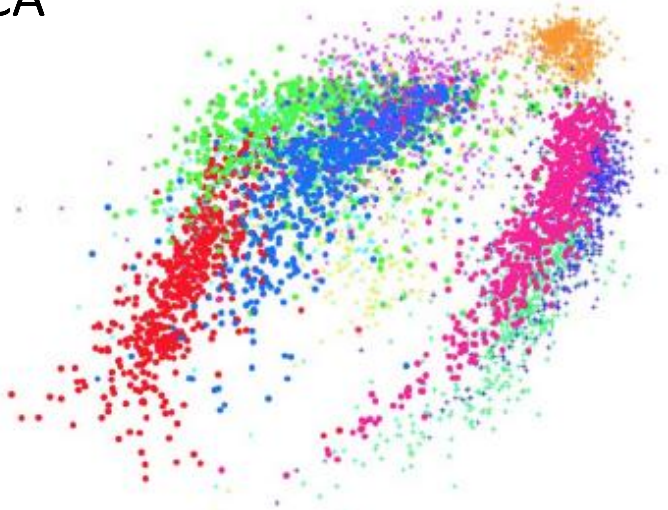


t-SNE

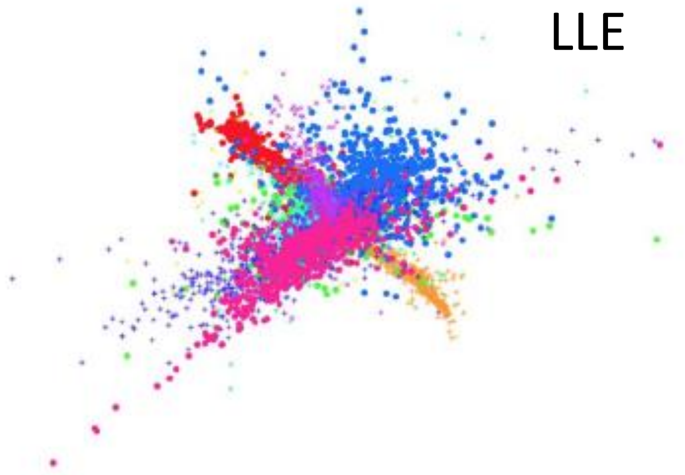


t-SNE

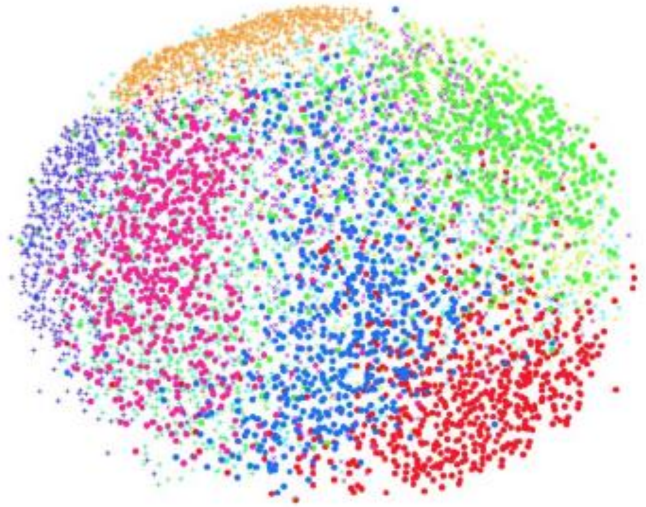
PCA



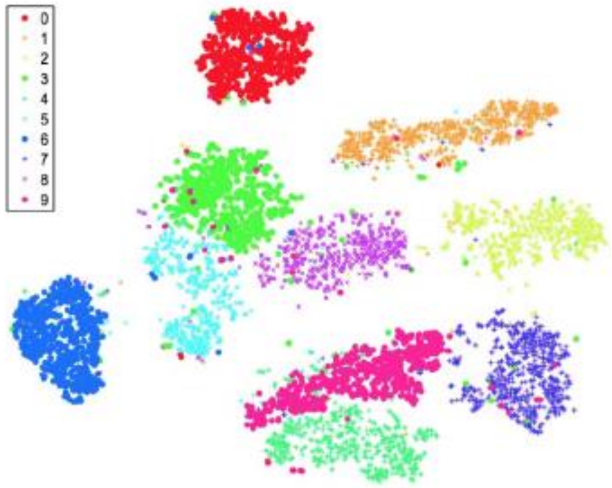
LLE



Sammon mapping



t-SNE



t-SNE thoughts

- Remarkably effective in visualizing cluster structure in data

- Arguably the best ultra low-dimensional technique that "just works"!

  A few contenders are popping up (eg. UMAP)

  [McInnes, Healy and Melville '18]

- Tends to cluster even when there may not be any clusters!

  Can result in false cluster discovery

  [Im, Verma and Branson '18]

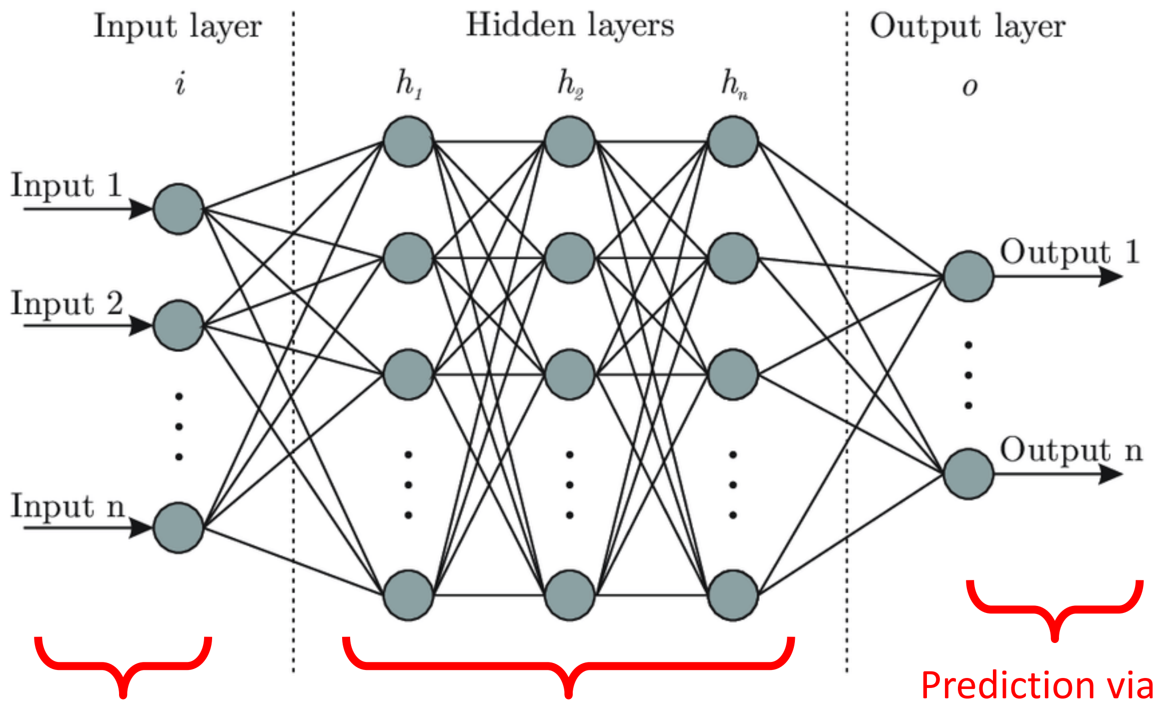
- Recent results have quantified the sufficient conditions needed for t-SNE to be provably successful in revealing the cluster structure in data.

  [Lindermann and Steinerberger'17, Arora et al. '18]

Auto-Encoders

Can we use Neural Networks to do (non-linear) dimension reduction?

Neural Networks



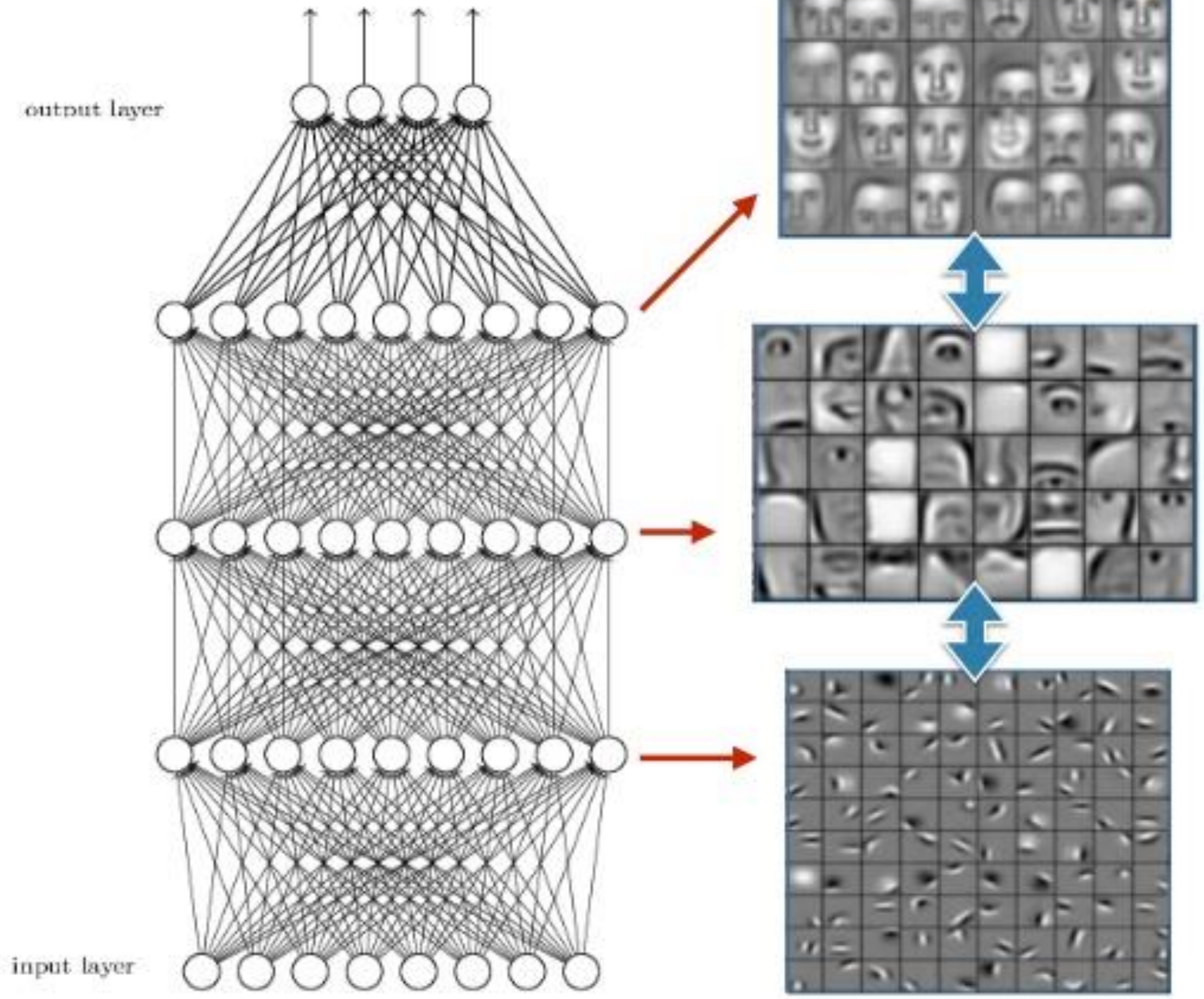
Weight learning is usually done by back-propagating the error signal (generated by the output label)

Original representation

Each hidden layer changes the data representation

Prediction via a simple softmax on the data representation according to the last hidden layer

Auto-Encoders



Hidden Layers

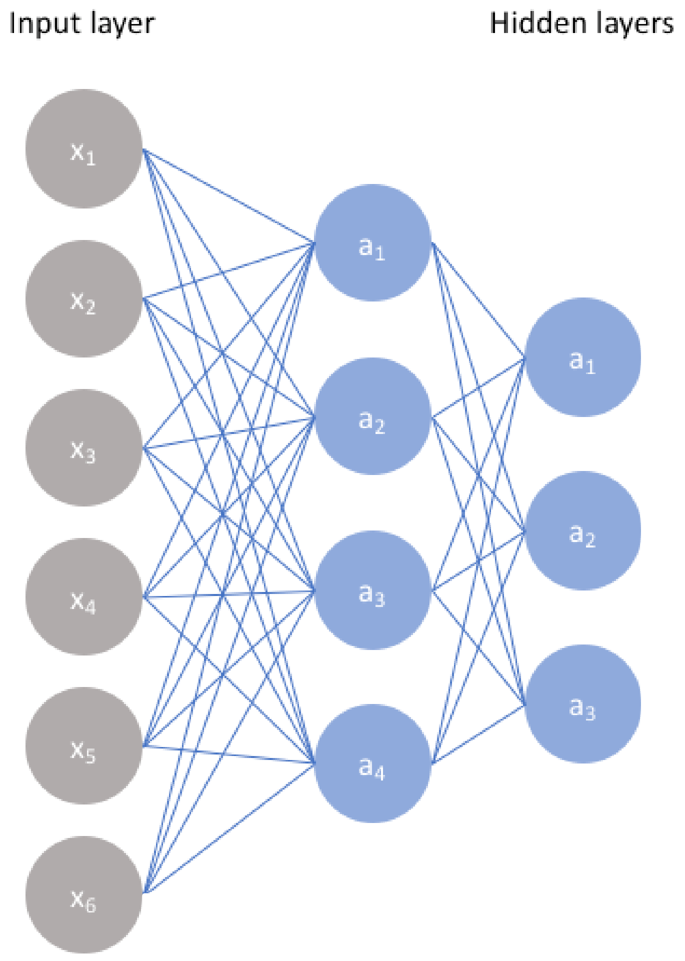
Final/deep hidden layers (the representation used for prediction)

Intermediate hidden layers

Initial hidden layers

Auto-Encoders

Idea: To re-represent input data in lower dimensions, we can have the hidden layers of a NN with a narrow width



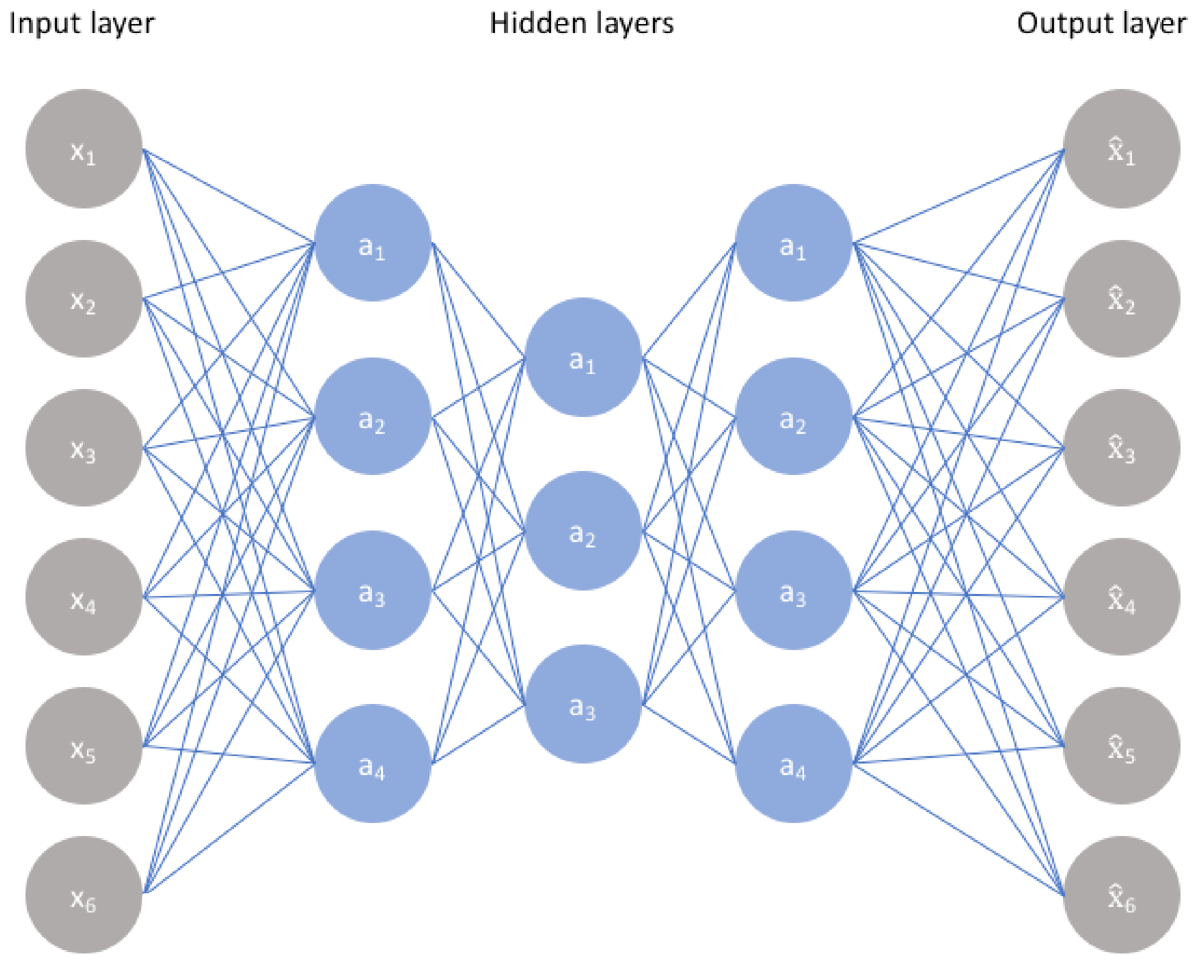
There are no labels, so…

- what should be the output?
- how can we learn the weights?

Idea:
Since the internal representation is supposed to represent the input, we should be able to retrieve/produce the input from the internal representation!

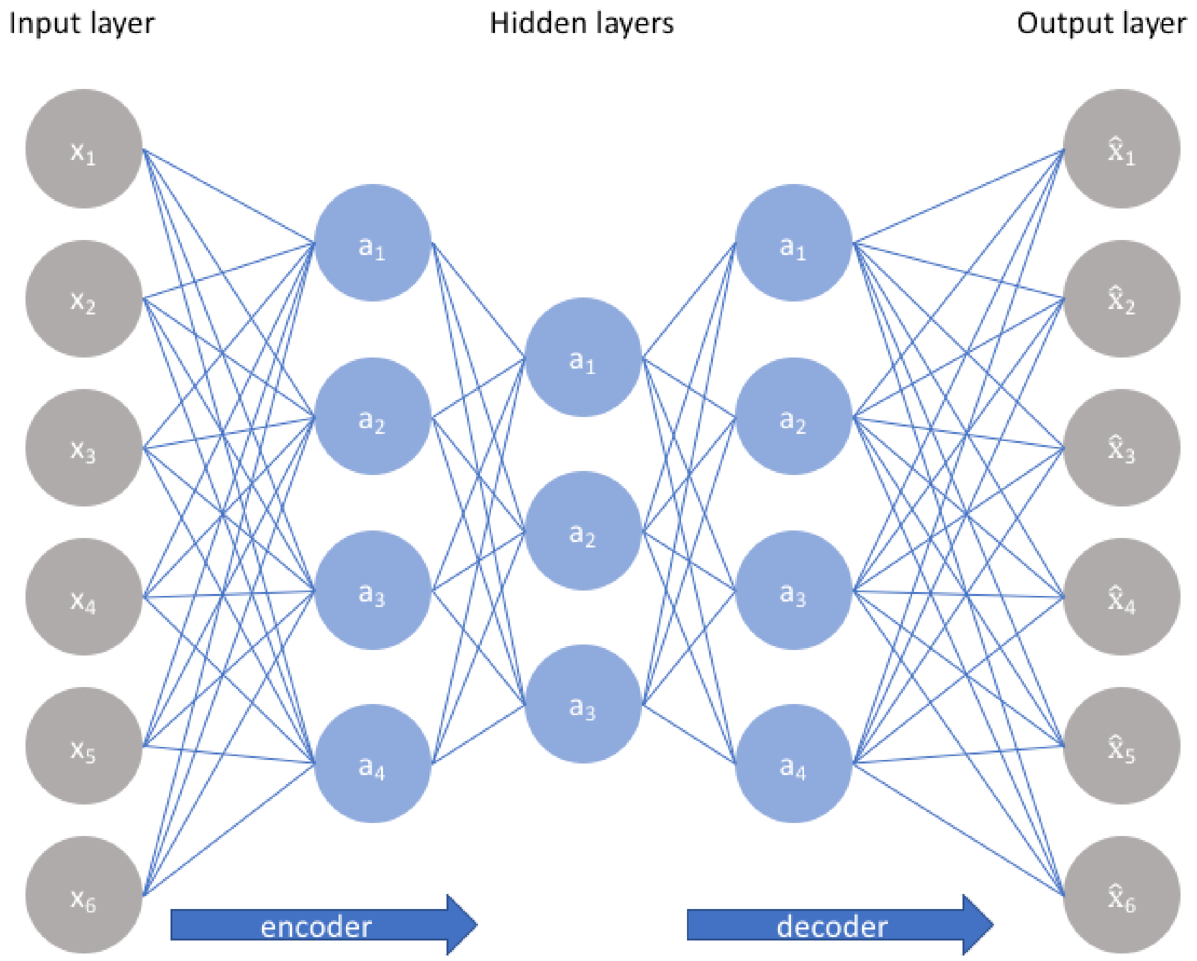
Auto-Encoders

Idea: To re-represent input data in lower dimensions, we can have the hidden layers of a NN with a narrow width



Auto-Encoders

The parameters are learned by backpropagating the error in input reconstruction

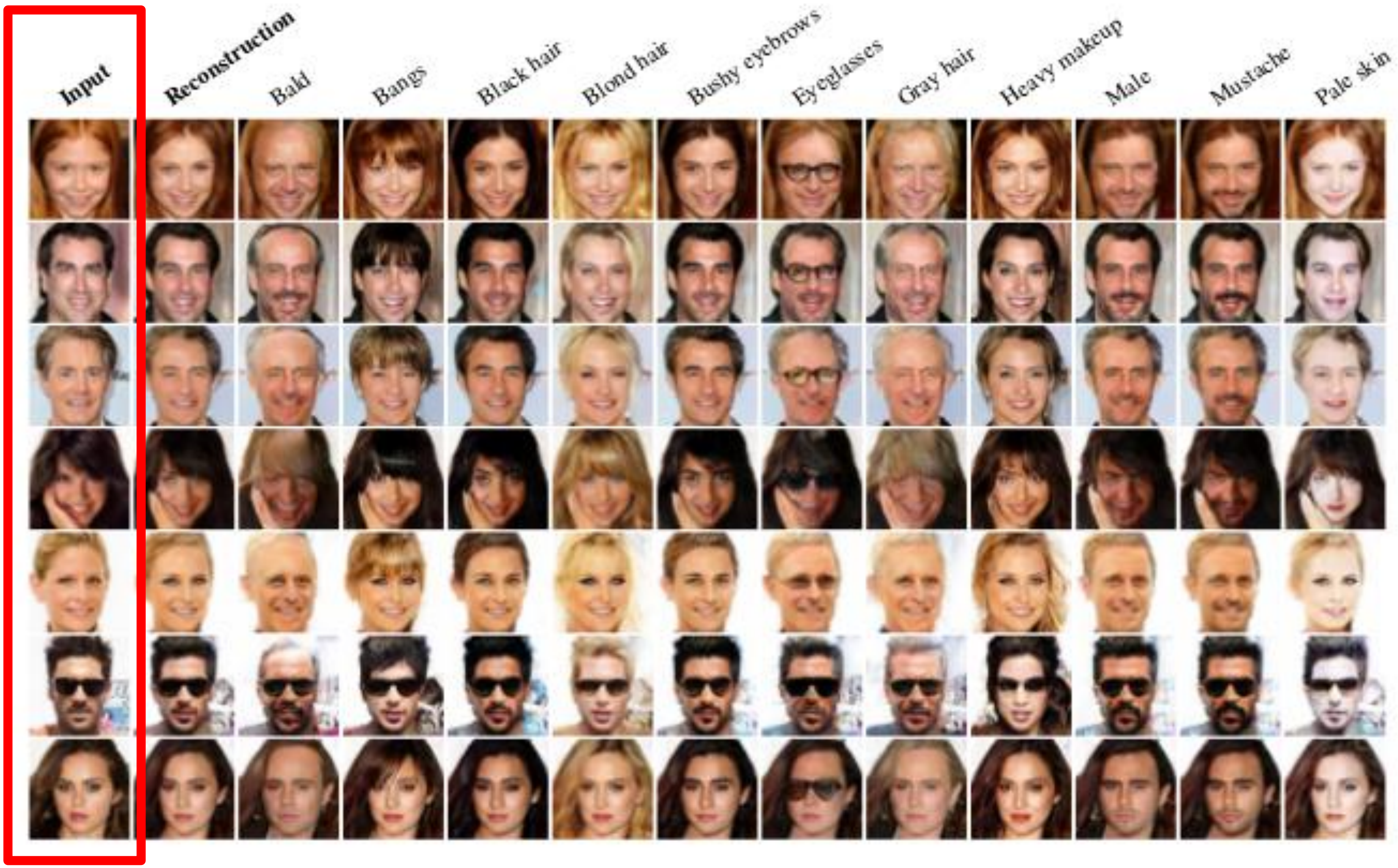


Auto-Encoders

Types of Auto-Encoders

- The specific type we saw is called undercomplete auto-encoder
  (hidden layer is of lower dimension than the input layer)
- Overcomplete auto-encoder
  (hidden layer is of higher dimension; it must be regularized)
  can be used for sparse representations
- Denoising auto-encoders (DAE)
  input is deliberately made noisy and the output is required to be a clean representation
- Variational Auto-Encoders (VAE)
  A generative model which can help generate new example datapoints

VAE Potential



Provable results in Manifold Learning

Let's fix a desirable property: **preserving geodesic distances**.

We are interested in the following question:

Given: a sample *X* from *n*-dimensional manifold $M \subset \mathbf{R}^D$, and

an embedding procedure $\mathcal{A} : M \to \mathbf{R}^d$

Define: the **quality** of embedding as $(1 \pm \epsilon)$-isometric, if for all p, q

$$(1-\epsilon) \leq \frac{D(\mathcal{A}(p),\mathcal{A}(q))}{D(p,q)} \leq (1+\epsilon)$$

Questions:

I. Can one come up an $\mathcal{A}$ that achieves $(1 \pm \epsilon)$-isometry?

II. How much can one reduce *d* and still have $(1 \pm \epsilon)$-isometry?

III. Do we need any restriction on *X* or *M*?

RP on Manifolds

An interesting result:

Let $M \subset \mathbf{R}^D$ be a *n*-dimensional manifold with volume *V* and curvature τ.

Then projecting it to a random **linear** subspace of dimension

$$O\left(\frac{n}{\epsilon^2} \log \frac{V}{\tau}\right)$$

Highly undesirable!
To have all distances within factor of 99% requires projection dimension > 10,000!

achieves $(1 \pm \epsilon)$-isometry with high probability.

Does not need any samples from the underlying manifold!

A Result via Nash Embeddings

For **any** compact *n*-dimensional manifold $M \subset \mathbf{R}^D$,
we present an algorithm that can embed *M* in

$$O(n + \ln(V/\tau^n))$$

dimensions that achieves $(1 \pm \epsilon)$-isometry (using only samples from *M*).

Embedding dimension is independent of ϵ !

Sample size is a function of ϵ

Nash Embedding: The Algorithm

Embedding Stage: Find a representation of *M* in lower dimensional space without worrying about maintaining any distances.

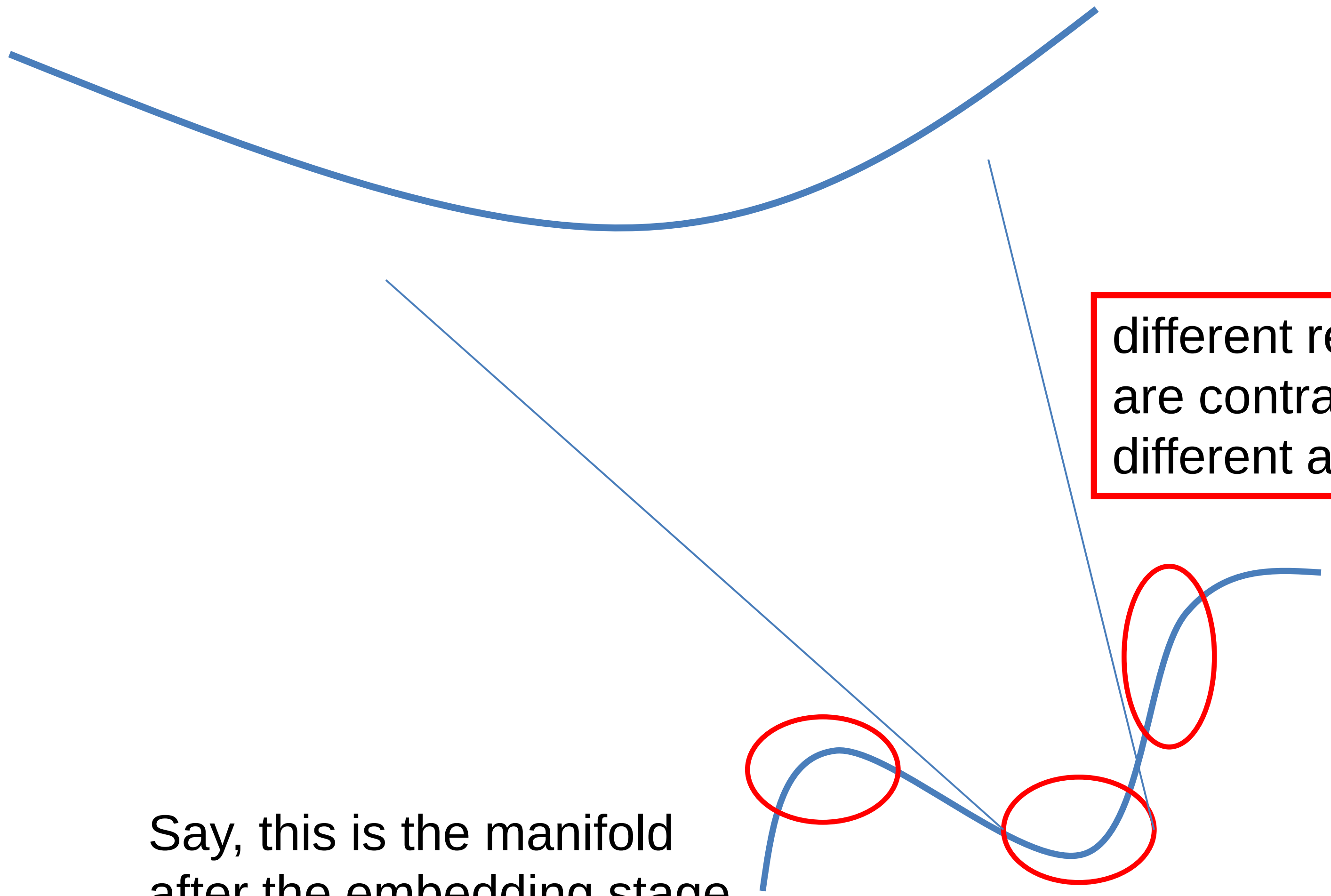
We can use a random linear projection without $1/\epsilon^2$ penalty

Correction Stage: Apply a series of corrections, each corresponding to a different region of the manifold, to restore back the distances.

This requires a bit of thinking…

Corrections

Zoomed in a local region

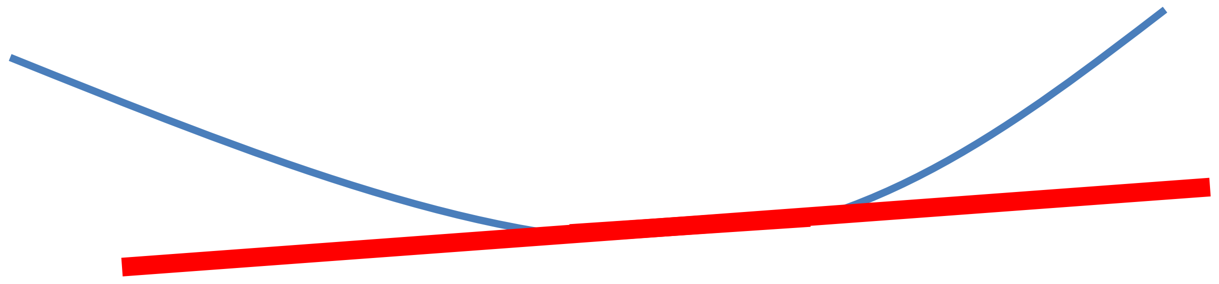


different regions are contracted by different amounts

Say, this is the manifold after the embedding stage

Corrections

Zoomed in a local region

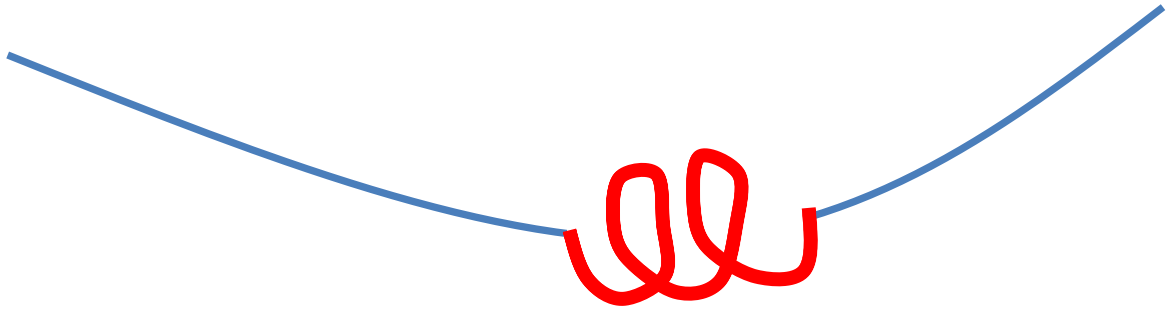


Suppose we linearly stretch this local region

cannot systematically attach the boundary of the stretched region back to the manifold…

Corrections

Zoomed in a local region



Instead we locally twist the space!

This creates the necessary stretch to restore back the local distances!

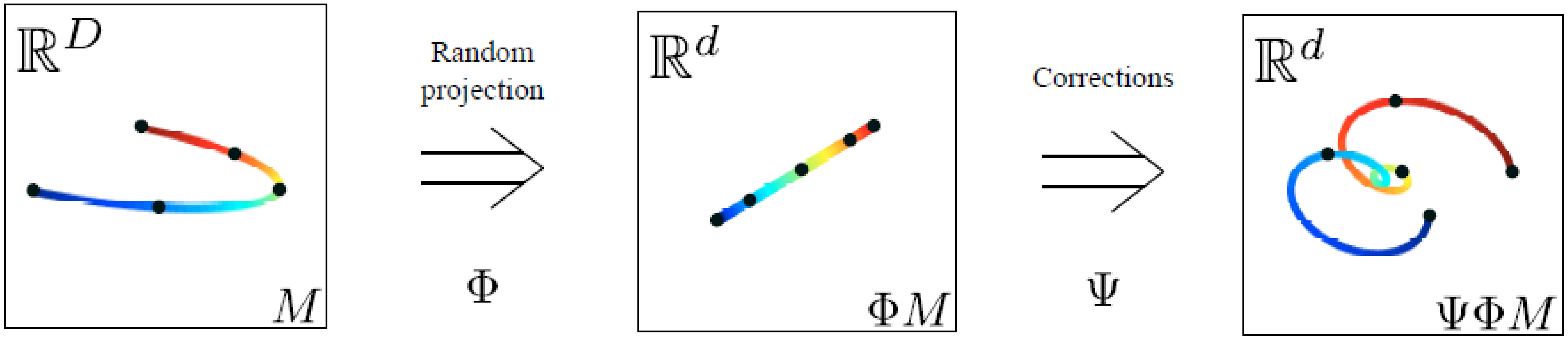
Technical hurdles to look-out for

- Care needs to be taken so as not to have **sharp** (non-differentiable) **edges on the boundary** while locally twisting the space.

- Sufficient ambient space needs to available to **create the local twist**.

- **Interference between different corrections** at overlapping localities need to be reconciled.

The algorithm at work



Theoretical guarantee

Theorem:

Let $M \subset \mathbf{R}^D$ be compact *n*-dimensional manifold with volume *V* and curvature τ. For any ϵ > 0, let *X* be $(D/\epsilon\tau)^n$-dense sample from *M*.

Then with high probability, our algorithm (given access to *X*) embeds any point from *M* in dimension

$$O(n + \ln(V/\tau^n))$$

with $(1 \pm \epsilon)$-isometry.

A quick proof overview

The goal is to **prove that the geodesic distances** between all pairs of points *p* and *q* in *M* are approximately preserved.

Recall that **length of any curve** γ is given by the expression:

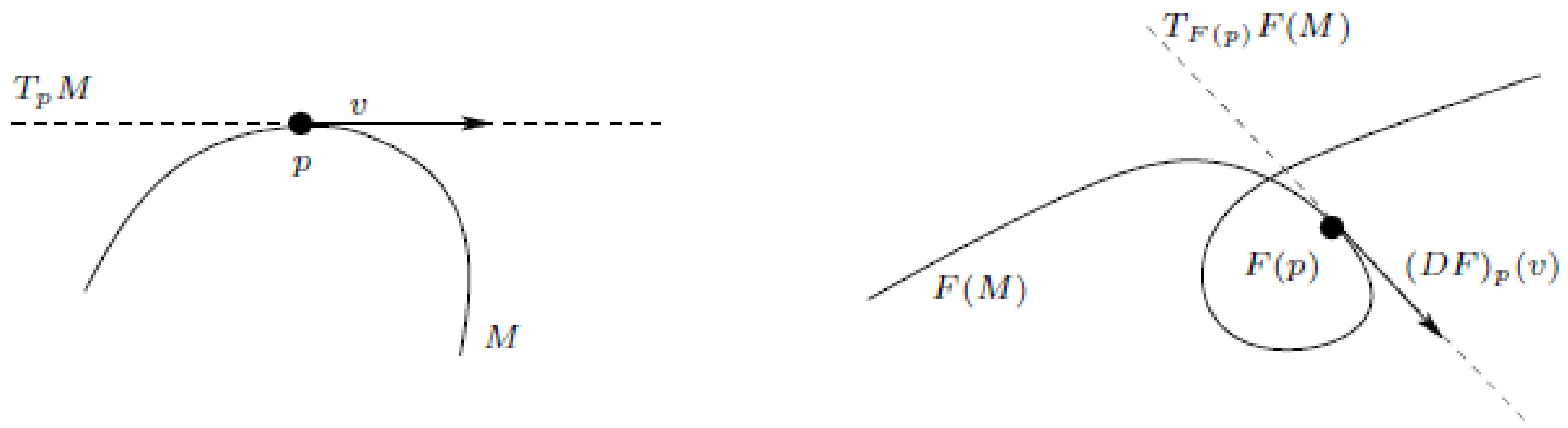
$$\int_a^b \|\gamma'(s)\| \, ds$$

length of a curve is the infinitesimal sum of the length of the tangent vectors along its path

Therefore, suffices to show that our algorithm preserves lengths of all vectors tangent to at all points in *M*.

A quick proof overview

From differential geometry, we know that for any smooth map *F*



the **exterior derivative** or the **pushforward** map *DF* acts on the tangent vectors.

We carefully analyze how each correction step of the algorithm changes the corresponding pushforward map.

Implications

- Gave the first **sample complexity result** for approximately isometric embedding for a manifold learning algorithms.

- **Novel algorithmic and analysis techniques** are of independent interest.

- One can use an existing manifold learning algorithm as the ‘embedding’ step. The corrections in second step enhance the embedding to make it isometric, making this as a **universal procedure**.