Embeddings of distance and similarity structure

A: Finding good representations of data

Uncovering latent structure

We have seen various models of **latent structure**, e.g.

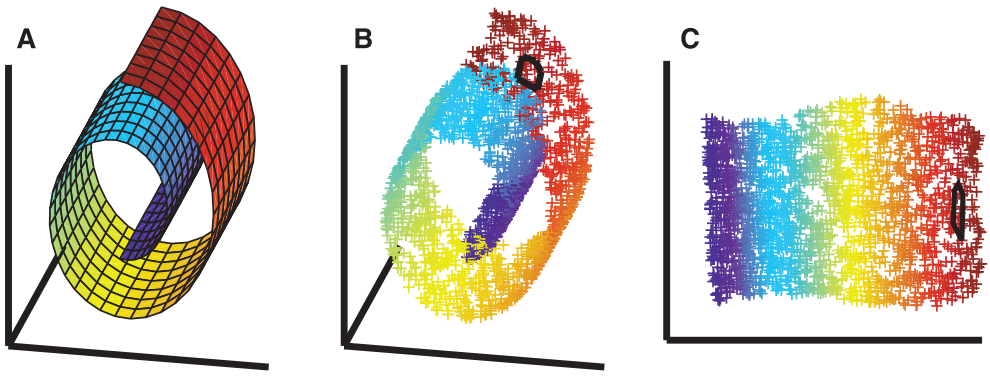
- clusters
- linear subspaces

and algorithms that map data to a representation in which this structure is exposed.

Uses of the latent representation:

- As a way to better understand the data
- As a representation for subsequent learning or processing

Embedding data into simpler spaces



- **Data space:** weird geometry, weird distance/similarity structure
- **Ideal embedding:** into a space where familiar geometry (e.g. Euclidean) and simple methods (e.g. linear separators) work well

Figure: S. Roweis and L. Saul (2000). Nonlinear dimensionality reduction by locally linear embedding.

B: Classical multidimensional scaling

Distance-preserving embeddings

- *Input:* An $n \times n$ matrix of pairwise distances

$$D_{ij} = \text{desired distance between points } i \text{ and } j,$$

  as well as an integer k.
- *Desired output:* an embedding $z^{(1)}, \ldots, z^{(n)} \in \mathbb{R}^k$ that realizes these distances as closely as possible.

Most widely-used algorithm: **classical multidimensional scaling.**

- $D \in \mathbb{R}^{n \times n}$: matrix of desired *squared* interpoint distances.
- Schoenberg (1938): D can be realized in Euclidean space if and only if $B = -\frac{1}{2}HDH$ is positive semidefinite, where $H = I_n - \frac{1}{n}11^T$.
- In fact, looking at this matrix B suggests an embedding even if it is not p.s.d.

The Gram matrix

In Euclidean space, write squared distances using dot products:

$$\|x - x'\|^2 \;=\; \|x\|^2 + \|x'\|^2 - 2x \cdot x' \;=\; x \cdot x + x' \cdot x' - 2x \cdot x'.$$

What about expressing dot products in terms of distances?

Let $z^{(1)}, \ldots, z^{(n)}$ be points in Euclidean space.

- Let $D_{ij} = \|z^{(i)} - z^{(j)}\|^2$ be the squared interpoint distances.
- Let $B_{ij} = z^{(i)} \cdot z^{(j)}$ be the dot products. **"Gram matrix"**

Moving between D and B:

- We've seen: $D_{ij} = B_{ii} + B_{jj} - B_{ij} - B_{ji}$. So D is linear in B.
- For $H = I_n - \frac{1}{n}11^T$, we have $B = -\frac{1}{2}HDH$.

And we can read off an embedding from the Gram matrix.

Question

Consider the two points $x_1 = (1, 0)$ and $x_2 = (-1, 0)$ in $\mathbb{R}^2$.

1. What is the matrix D of squared interpoint distances?
2. Write down $H = I_n - \frac{1}{n}11^T$.
3. Compute $B = -\frac{1}{2}HDH$. Is this the correct Gram matrix?

Recovering an embedding from interpoint distances

$D \in \mathbb{R}^{n \times n}$: matrix of desired *squared* interpoint distances.
Suppose these are realizable in Euclidean space: that is, there exist vectors $z^{(1)}, \ldots, z^{(n)}$ such that $D_{ij} = \|z^{(i)} - z^{(j)}\|^2$.

We have seen how to obtain the Gram matrix, $B_{ij} = z^{(i)} \cdot z^{(j)}$.

- B is p.s.d. (why?). Thus its eigenvalues are nonnegative.
- Compute spectral decomposition:

$$B = U \Lambda U^T = Y Y^T,$$

  where Λ is the diagonal matrix of eigenvalues and $Y = U\Lambda^{1/2}$.
- Denote the rows of Y by $y^{(1)}, \ldots, y^{(n)}$. Then

$$y^{(i)} \cdot y^{(j)} = (YY^T)_{ij} = B_{ij} = z^{(i)} \cdot z^{(j)}.$$

  If dot products are preserved, so are distances.

Result: embedding $y^{(1)}, \ldots, y^{(n)}$ exactly replicates D. **What is the dimensionality?**

Classical multidimensional scaling

A slight generalization works even when the distances cannot necessarily be realized in Euclidean space.

Given $n \times n$ matrix D of squared distances, target dimension k:

1. Compute $B = -\frac{1}{2} HDH$.
2. Compute the spectral decomposition $B = U\Lambda U^T$, where the eigenvalues in Λ are arranged in decreasing order.
3. Zero out any negative entries of Λ to get Λ_+.
4. Set $Y = U\Lambda_+^{1/2}$.
5. Set Y_k to the first k columns of Y.
6. Let the embedding of the n points be given by the rows of Y_k.

Example

CITIES	ATLA.	CHIC.	DENV.	HOUS.	L.A.	MIAMI	N.Y.	S.F.	SEAT.	WASH D.C.
ATLANTA		587	1212	701	1936	604	748	2139	2182	543
CHICAGO	587		920	940	1745	1188	713	1858	1737	597
DENVER	1212	920		879	831	1726	1631	949	1021	1494
HOUSTON	701	940	879		1374	968	1420	1645	1891	1220
LOS ANGELES	1936	1745	831	1374		2339	2451	347	959	2300
MIAMI	604	1188	1726	968	2339		1092	2594	2734	923
NEW YORK	748	713	1631	1420	2451	1092		2571	2408	205
SAN FRANCISCO	2139	1858	949	1645	347	2594	2571		678	2442
SEATTLE	2182	1737	1021	1891	959	2734	2408	678		2329
WASHINGTON D.C.	543	597	1494	1220	2300	923	205	2442	2329	

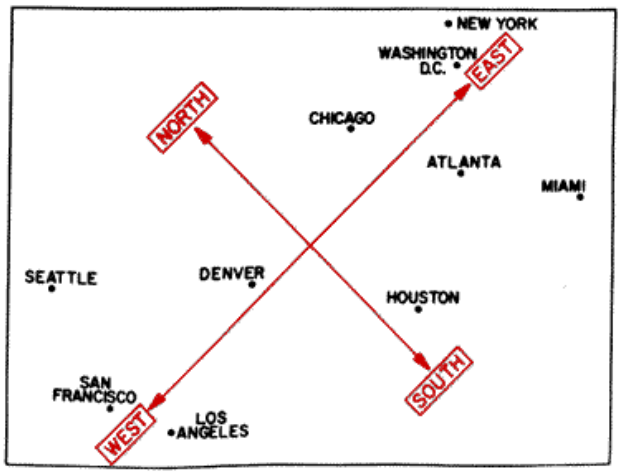


Figure: J.B. Kruskal and M. Wish (1978). Multidimensional scaling.

Alternative: embedding by gradient descent

Given target distances $\{D_{ij}\}$, write down a cost function for the desired embedding $y_1, \ldots, y_n \in \mathbb{R}^k$, e.g.

$$L(y_1, \ldots, y_n) = \sum_{i,j} \left| D_{ij}^2 - \|y_i - y_j\|^2 \right|$$

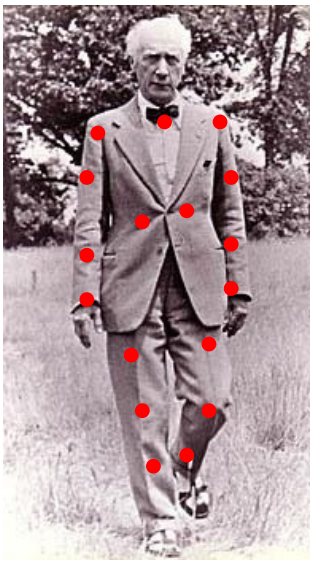
and use gradient descent to optimize the $\{y_i\}$ directly.

Problem: no guarantees available.

C: Manifold learning

Low dimensional manifolds

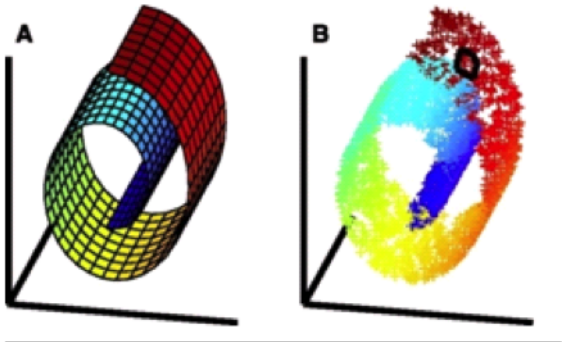
Sometimes data in a high-dimensional space $\mathbb{R}^d$ in fact lies close to a k-dimensional manifold, for $k \ll d$



1. Motion capture
   M markers on a human body yields data in $\mathbb{R}^{3M}$
2. Speech signals
   Representation can be made arbitrarily high dimensional by applying more filters to each window of the time series

This whole area: "Manifold learning"

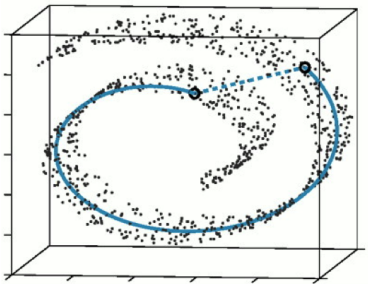
The ISOMAP algorithm



ISOMAP (Tenenbaum et al, 1999): given data $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$,

1. Estimate *geodesic distances* between the data points: that is, distances along the manifold.
2. Then embed these points into Euclidean space so as to (approximately) match these distances.

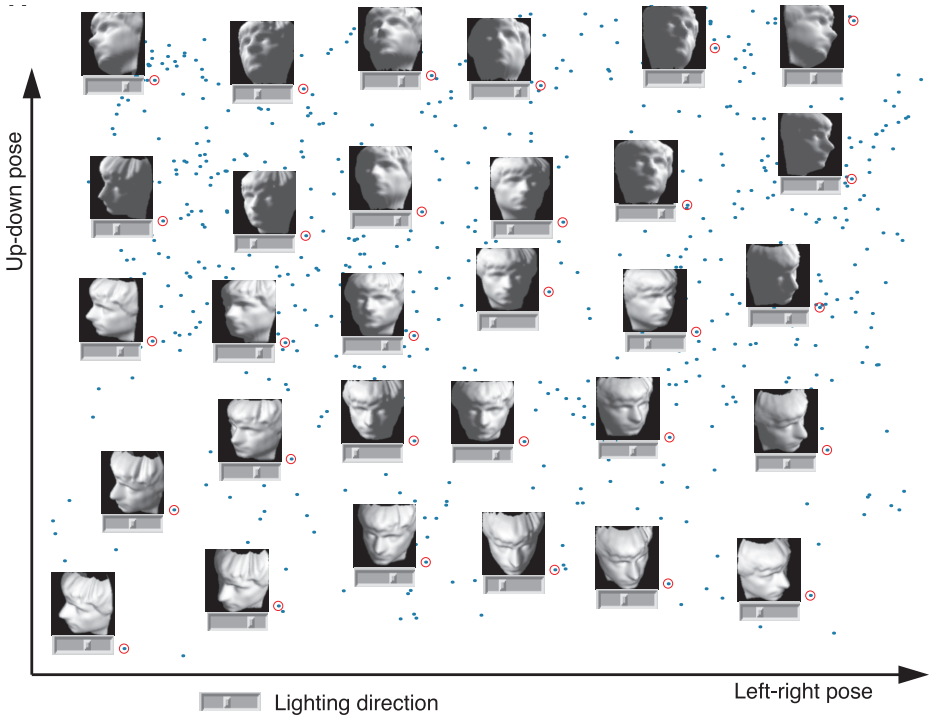
Estimating geodesic distances



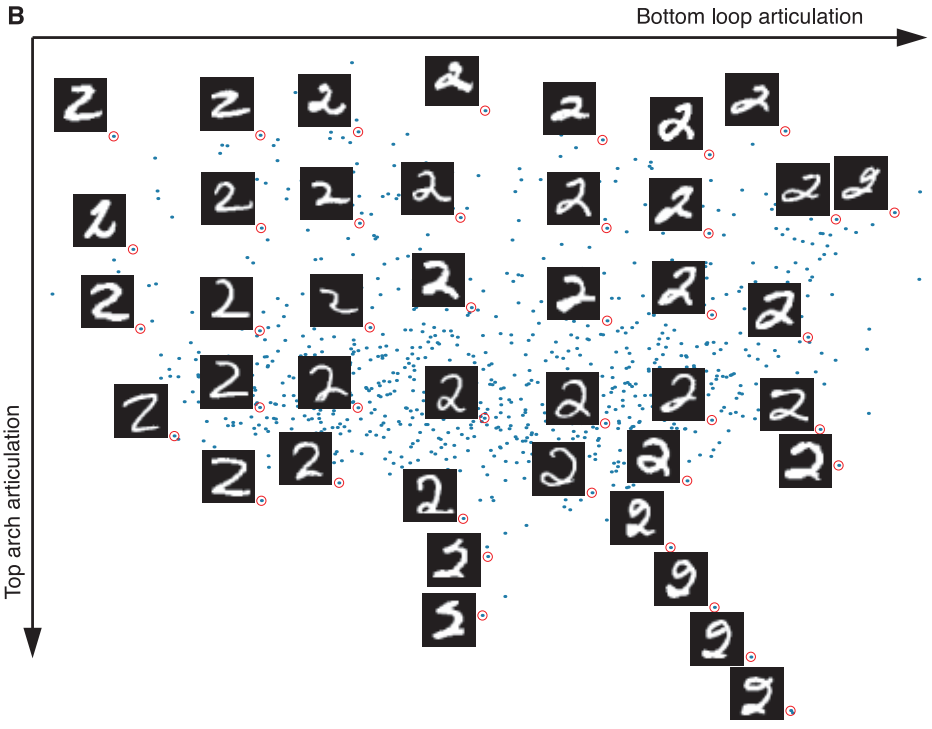
Key idea: for **nearby** points,
geodesic distance $\approx$ Euclidean distance

1. Construct the **neighborhood graph.**
   Given data $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$, construct a graph $G = (V, E)$ with
   - Nodes $V = \{1, 2, \ldots, n\}$ (one per data point)
   - Edges $(i, j) \in E$ whenever $x^{(i)}$ and $x^{(j)}$ are close together
2. Compute distances in this graph.
   Set length of $(i, j) \in E$ to $\|x^{(i)} - x^{(j)}\|$.
   Compute distances between all pairs of nodes.

ISOMAP: examples



ISOMAP: examples



More manifold learning

1. Other good algorithms, such as
   - Locally linear embedding
   - Laplacian eigenmaps
   - Maximum variance unfolding
2. Notions of intrinsic dimensionality
3. Statistical rates of convergence for data lying on manifolds
4. Capturing other kinds of topological structure

D: Out-of-sample extension

The problem

The methods we've seen so far:

- Input: some distance information about a set of points $x_1, \ldots, x_n \in \mathcal{X}$
- Output: embedding $y_1, \ldots, y_n \in \mathbb{R}^k$ that satisfies these constraints

Problem: How to embed new points $x \in \mathcal{X}$?

Two solution strategies

- Input: some distance information about a set of points $x_1, \ldots, x_n \in \mathcal{X}$
- Output: embedding $y_1, \ldots, y_n \in \mathbb{R}^k$ that satisfies these constraints

Goal: Want to embed a new point $x \in \mathcal{X}$

1. **Landmark points.**
   - Typically $k \ll n$. Can we embed x using its distances to just $O(k)$ of the x_i?
   - Choose $O(k)$ landmark points, spread out nicely, to use for this purpose.

2. **Learning an embedding function.**
   Learn a mapping $f : \mathcal{X} \to \mathbb{R}^k$ such that $f(x_i) = y_i$, and apply this to new points.

E: Embeddings of similarity assessments

Partial information about distance/similarity function

Want an embedding of $x_1, \ldots, x_n$ that respects some distance function $d(x, x')$ or similarity function $s(x, x')$. But these functions are unknown!

Partial information is available in the form of comparisons, e.g.:

1. A collection of "similar" pairs and "dissimilar" pairs
2. Triplet comparisons: x is more similar to x' than to x''
3. Quadruplet comparisons: x is more similar to x' than u is to u'

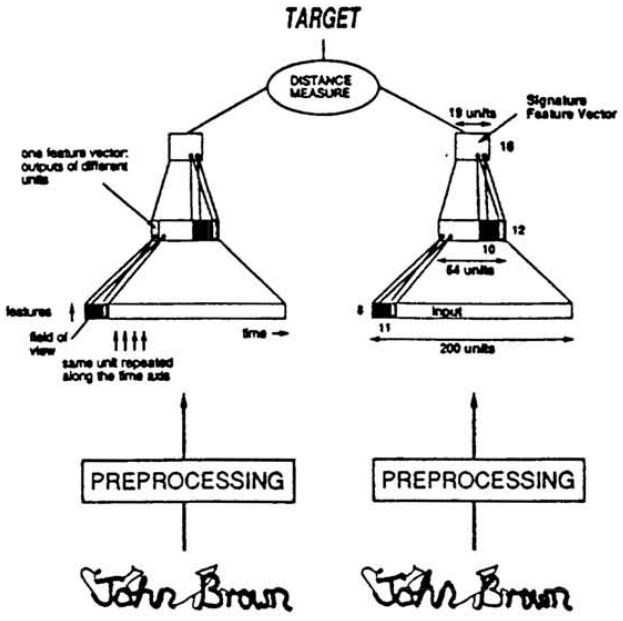
How to embed based on these?

Example: triplets

- Input: a collection T of triplets (i, j, k), meaning that x_i is more like x_j than x_k
- Desired output: Embedding $y_1, \ldots, y_n \in \mathbb{R}^k$

F: Siamese nets

Application: checking signatures for forgery



Similarity function $s(x, x') = f(\phi(x), \phi(x'))$:

- ϕ is produced by a neural network
- $f(\cdot)$ can be, e.g., cosine similarity
- Net is trained using a loss function that penalizes similar pairs that are far apart and dissimilar pairs that are close together

Bromley, Guyon, LeCun, Sackinger, Shah (1993). Signature verification using a "Siamese" time delay neural network.