CSE 291B 2/12

Metric Multidimensional scaling (MDS)

- Problem: from distances D_{ij}, how to compute $\vec{x}_i \in \mathbb{R}^d$ such that $\|\vec{x}_i - \vec{x}_j\| \approx D_{ij}$?
- Step #1: derive Gram matrix of inner products

$$G_{ij} = \frac{1}{2}\left[\frac{1}{N}\sum_k \left(D_{kj}^2 + D_{ik}^2\right) - \frac{1}{N^2}\sum_{kl} D_{kl}^2 - D_{ij}^2\right]$$

- Step #2: singular value decomposition (SVD)

$$G = V\Sigma V^T \quad \text{with} \quad \Sigma = \begin{pmatrix} \sigma_1 & & \emptyset \\ & \sigma_2 & \\ \emptyset & & \ddots \; \sigma_N \end{pmatrix} \quad \text{and} \quad VV^T = V^TV = I$$

$$X_{i\alpha} = \sqrt{\sigma_\alpha}\, V_{i\alpha} \quad \text{for} \quad i = 1..N, \quad \alpha = 1..d$$

- Estimating the dimensionality d

Singular values σ_α measure variance in αth dimension:

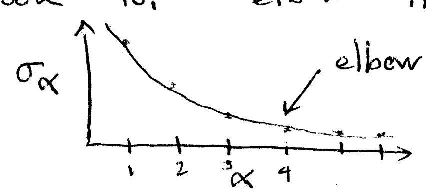
$$\mathrm{Var}[X_\alpha] = \frac{1}{N}\sum_{i=1}^N X_{i\alpha}^2 \quad \text{b/c } \{\vec{x}_i\} \text{ are centered}$$

$$= \frac{1}{N}\sum_{i=1}^N \sigma_\alpha V_{i\alpha}^2$$

$$= \frac{\sigma_\alpha}{N} \quad \text{b/c } VV^T = I$$

Rules of thumb:

1) look for "elbow" in eigenvalue spectrum



2) choose d to capture $(1-\delta)$ fraction of total variance:

$$\sum_{\alpha=1}^d \sigma_\alpha \geq \left(\frac{1-\delta}{\delta}\right)\sum_{\alpha=d+1}^N \sigma_\alpha$$

Duality of MDS and PCA

- Inputs $\vec{x}_i \in \mathbb{R}^D$; assume inputs are centered $\sum \vec{x}_i = \vec{0}$

$$X = \begin{array}{c} \xleftrightarrow{\quad N \quad} \\ \begin{bmatrix} \vec{x}_1 & \vec{x}_2 & \cdots & \vec{x}_N \end{bmatrix} \end{array} \updownarrow D$$

- Inner products $(N \times N)$

$$G_{ij} = \vec{x}_i \cdot \vec{x}_j \Longleftrightarrow G = X^T X$$

- Covariance matrix $(D \times D)$

$$C_{\alpha\beta} = \frac{1}{N} \sum_{i=1}^{N} X_{i\alpha} X_{i\beta} \Longleftrightarrow C = \frac{1}{N}(XX^T)$$

- Eigenvalues

G and C have same non-zero eigenvalues up to factor $(\frac{1}{N})$.

\# non-zero eigenvalues $\leq \min(D, N)$

- Equivalence

PCA on inputs $\{\vec{x}_i\}_{i=1}^{N}$ gives same results as MDS on distances $\|\vec{x}_i - \vec{x}_j\|$.

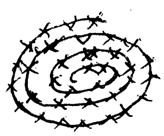
Maximum variance subspace = Maximally inner-product preserving subspace

Manifold learning

- Problem: given high dimensional data that lies on (or near) a low dimensional manifold, how to compute a "faithful" embedding?

- Examples of manifolds

  1) Spiral

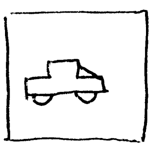
  

  extrinsic coordinates: (x,y) in 2D plane
  intrinsic coordinates: arc length along spiral

  no linear projection can "unfold" the spiral

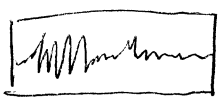
  $(x,y) \longrightarrow \theta$

  2) images of car from different viewpoints

  extrinsic coordinates: pixels
  intrinsic coordinates: camera location 3d

  3) vowel sounds

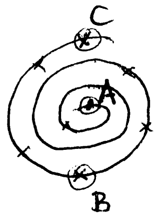
  extrinsic coordinates: waveform samples
  intrinsic coordinates: articulator positions (e.g. tongue, lips, etc.)

- Notation

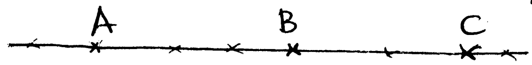
  inputs $\vec{x}_i \in \mathbb{R}^D$ $(i=1..N)$
  outputs $\vec{y}_i \in \mathbb{R}^d$ $(d \ll D)$

- Goals
  - nearby points remain nearby
  - distant points remain distant
  - estimate dimensionality d

- Non-monotonicity



$\boxed{\text{Rank ordering of Euclidean distances will not be preserved}}$



$\|A - C\| > \|A - B\|$

$\|A - C\| < \|A - B\|$

- Linear vs. nonlinear

  What computational price must we pay for nonlinear dimensionality reduction (NLDR)?

- Conventional wisdom (pre-2000)

  NLDR requres difficult nonlinear optimization:
  - latent variable models (mixture of factor analyzers)
  - autoencoder neural networks
  - self-organizing maps

  Pre-2000: many <u>local</u> minima!

- Since 2000

  spectral methods for NLDR:
  - based on tractable matrix computations
  - not much more complicated than MDS

  no <u>local</u> minima

Isomap algorithm

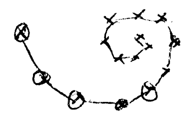
Isomap = "isometric mapping"
distance-preserving

Step #1: compute neighborhood graph $\sim O(DN^2)$

Nodes represent inputs $\{\vec{x}_i\}_{i=1}^{N}$.

Edges represent k-nearest neighbor (kNN) relations.

k is ONLY free parameter of algorithm.

Ex: spiral  graph is discretized skeleton of underlying manifold

(k=2)

Step #2: compute shortest paths $\sim O(N^3)$

Weight each edge by local distance between kNN.

Compute distance D_{ij} of shortest path from node i to node j (e.g. Dijkstra's algorithm).

Graph distances approximate "geodesic" distances along manifold.

(Not the same as Euclidean distances, except for kNN.)

Step #3: Run MDS on graph-based distances.

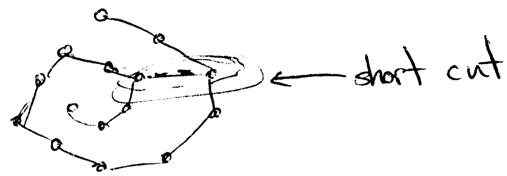
- Derive Gram matrix
- Perform SVD
- Estimate (manifold) dimensionality from eigenvalue spectrum.

- Strengths of Isomap
  - polynomial time optimizations
  - no local minima
  - non-iterative (just one pass through data)
  - non-parametric (no explicit assumptions about form of manifold)
  - only heuristic is neighborhood size

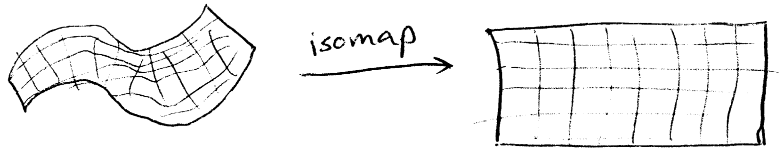
- Asymptotic convergence

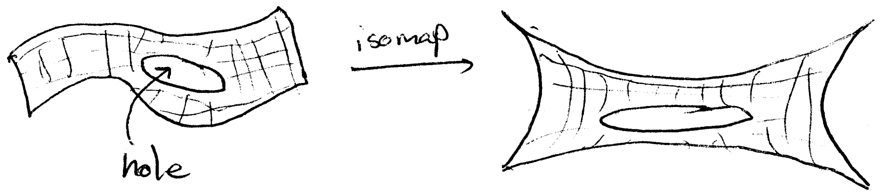
<u>Thm:</u> For data sampled from a manifold that is isometric to a <u>convex</u> subset of Euclidean space, Isomap will recover the subset up to rotation and translation.

- Issues
  - sensitivity to "short cuts" in graph when data is sparse or noisy



  - graph distances D_{ij} may not yield a positive semidefinite Gram matrix
  - convexity assumption: manifold has no "holes"





distortion!

- holes introduce distortions into the embedding
- spurious extra dimensions appear in embedding