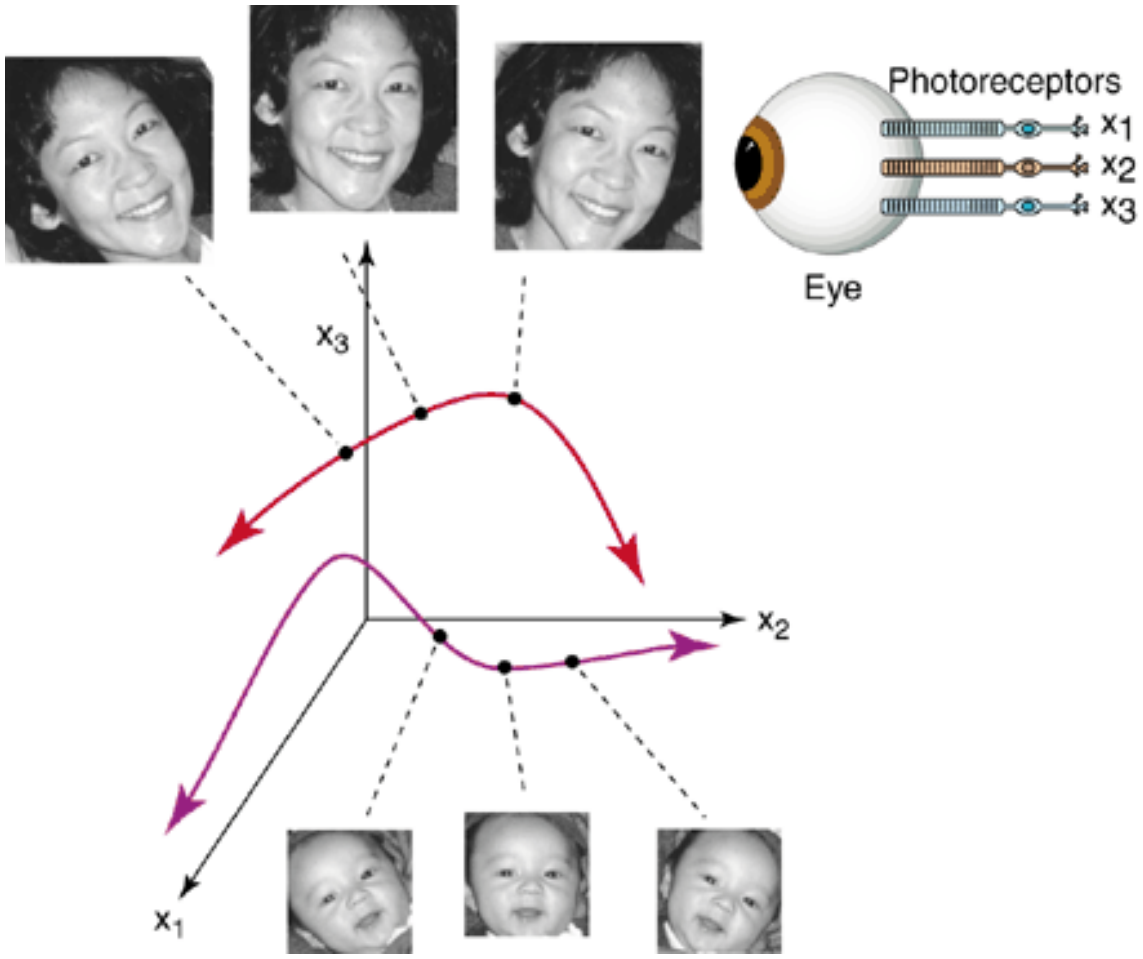
Lecture 16
Multidimensional Scaling & Isomap

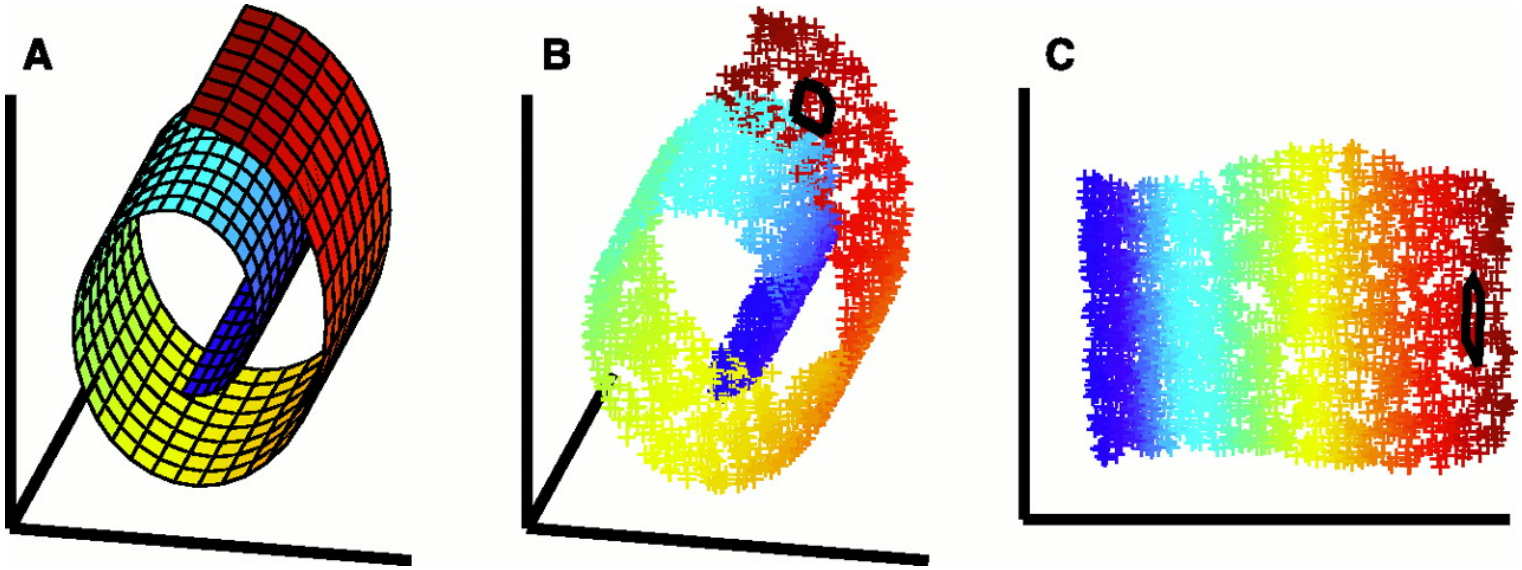
Motivation

- high-dimensional data often lies on an intrinsically low-dimensional manifold
- e.g. consider set of all rotations of a 20×20 pixel image, this might trace out a curve (or loop) in $\mathbb{R}^{400}$



- Goal: given high-dimensional points $\mathbf{x}_i$, recover the low-dimensional coordinates of the data, $\mathbf{y}_i$, that describe where the points lie on the mainfold
- (in other words) find an embedding of the data in a low-dimensional space, that preserves its essential regularities
- Useful for: data visualization (it's hard to see more than 3D), discovery of interesting structure in data, smooth interpolation

Another example



- data comes from a 2D plane, embedded in 3D space
- ideally we would like to find a mapping such that nearby points on the roll (B) are also adjacent in 2D (C)

Multidimensional Scaling

- images ($\mathbf{y}_i$) of original points ($\mathbf{x}_i$) should have approximately the same interpoint distances as the originals
- let δ_{ij} = distance between $\mathbf{x}_i$ and $\mathbf{x}_j$,
  and d_{ij} = distance between $\mathbf{y}_i$ and $\mathbf{y}_j$
  e.g. Euclidean distance $d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|_2$
- we want $\forall i, j\ (\delta_{ij} \approx d_{ij})$
- exact equality generally not possible, so minize an error function J of the $\mathbf{y}$'s

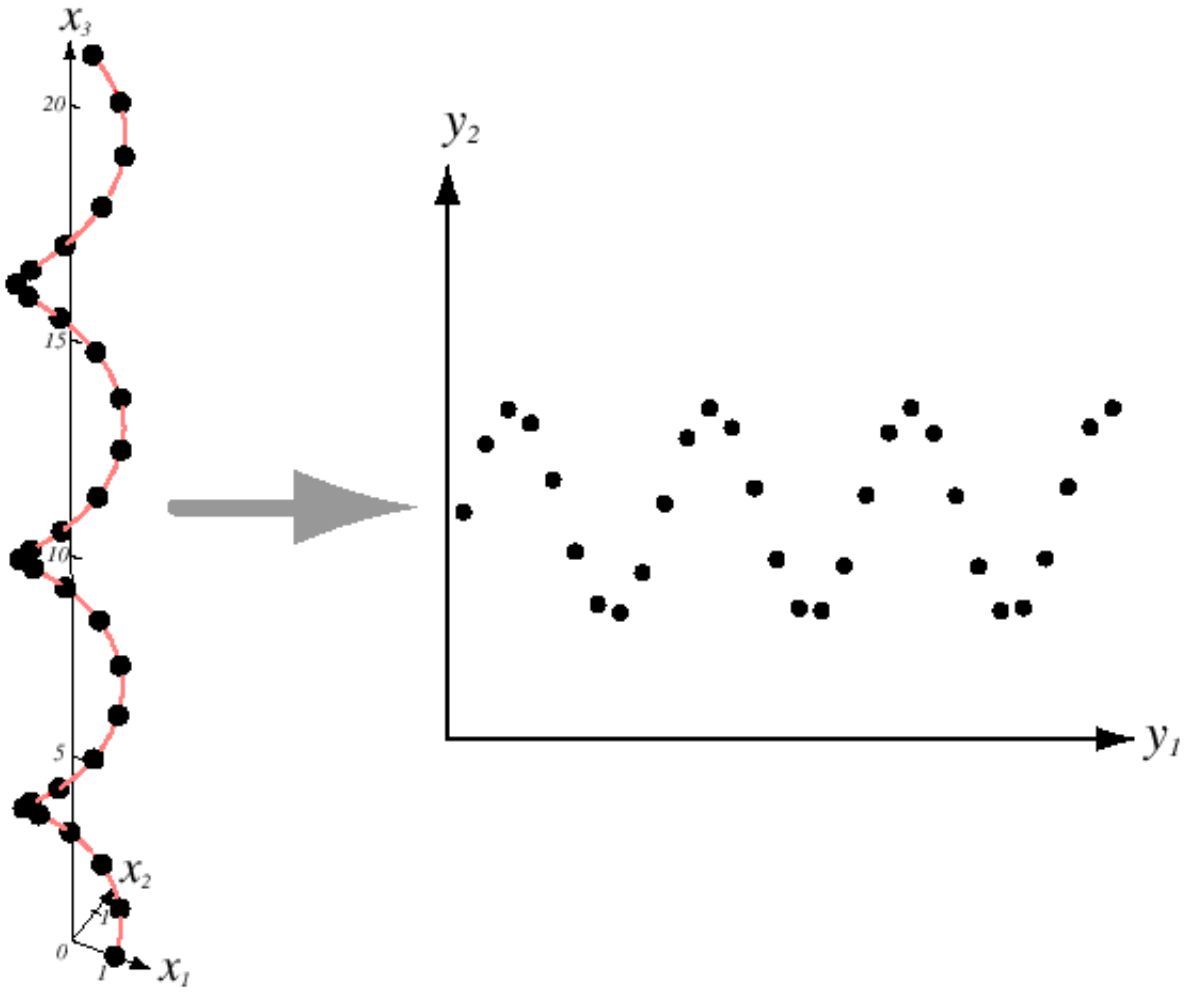
Error Functions

- remember: d_{ij} is a function of $\mathbf{y}_i$ and $\mathbf{y}_j$, and given the data the δ_{ij}'s are constant
- $J_{ee} = \dfrac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$ penalizes large absolute errors
- $J_{ff} = \displaystyle\sum_{i<j}\left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$ penalizes large relative errors
- $J_{ef} = \dfrac{1}{\sum_{i<j}\delta_{ij}} \displaystyle\sum_{i<j}\frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}$ a compromise between the two

Gradient Updates

- $\nabla J_{ee}(\mathbf{y}_k) = \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{kj}) \frac{\mathbf{y}_k - \mathbf{y}_j}{d_{kj}}$
- $\nabla J_{ff}(\mathbf{y}_k) = 2 \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}^2} \frac{\mathbf{y}_k - \mathbf{y}_j}{d_{kj}}$
- $\nabla J_{ef}(\mathbf{y}_k) = \frac{2}{\sum_{i<j} \delta_{ij}} \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}} \frac{\mathbf{y}_k - \mathbf{y}_j}{d_{kj}}$
- Algortithm:
  - compute or obtain distances δ_{ij}
  - initialize the points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ (e.g. randomly)
  - until convergence, $\forall_i \quad \mathbf{y}_i \leftarrow \mathbf{y}_i - \eta \nabla J(\mathbf{y}_i)$

MDS in action

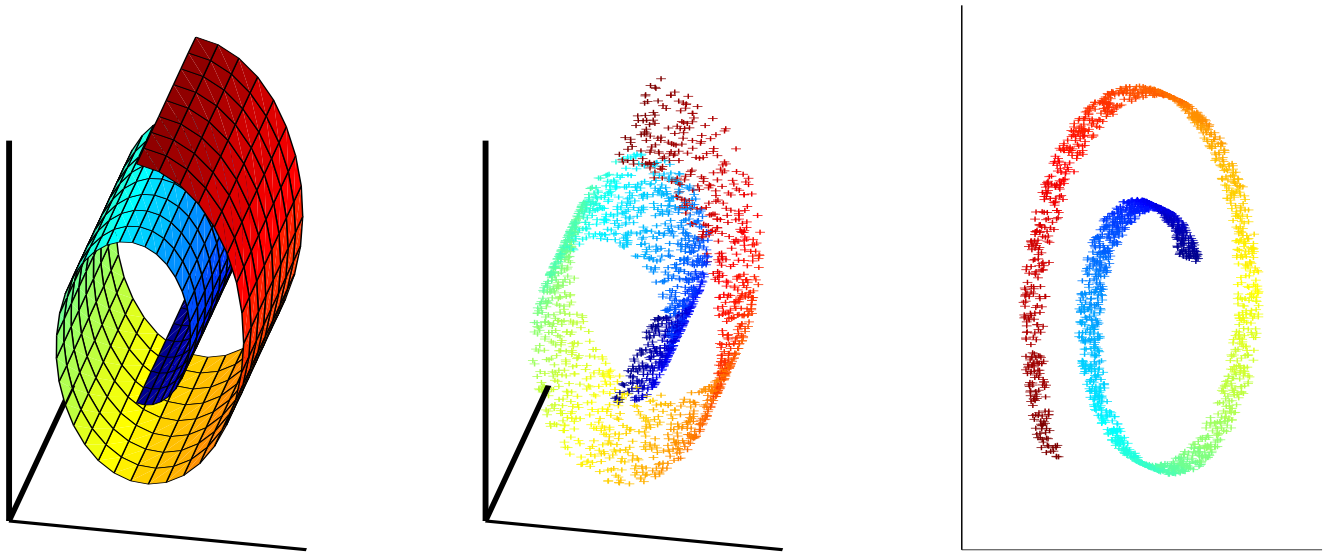


Other Notes

- δ_{ij}'s need not be actual distances between points, then can be arbitrary 'disparities' between items
  e.g. the dissimilarity between two faces, as judged by a human observer

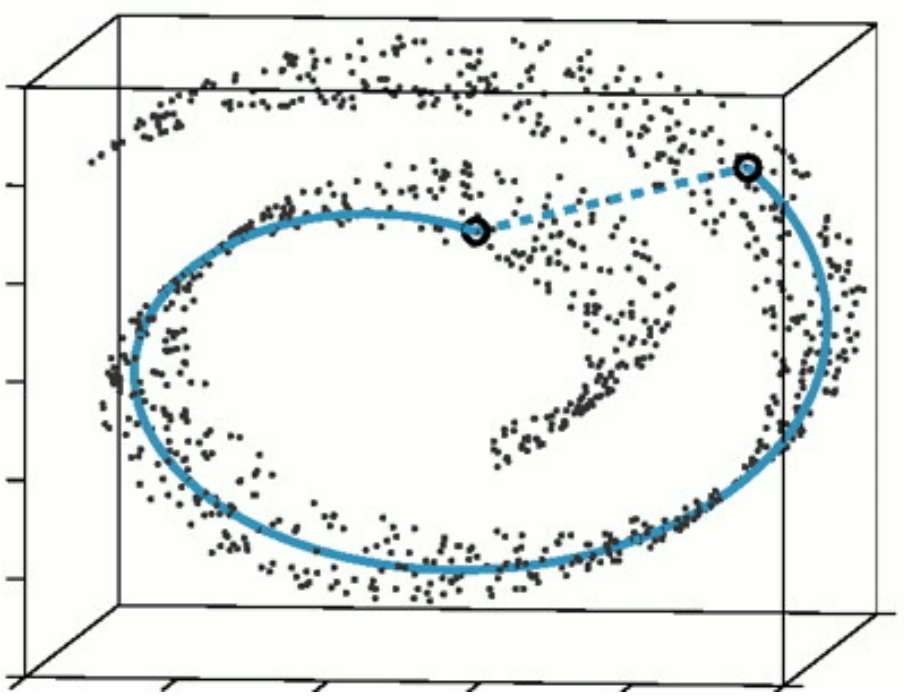
- MDS + Euclidean distance + Squared Error = PCA

- MDS can flatten the 'swissroll' but cannot unroll it – why?



Isomap

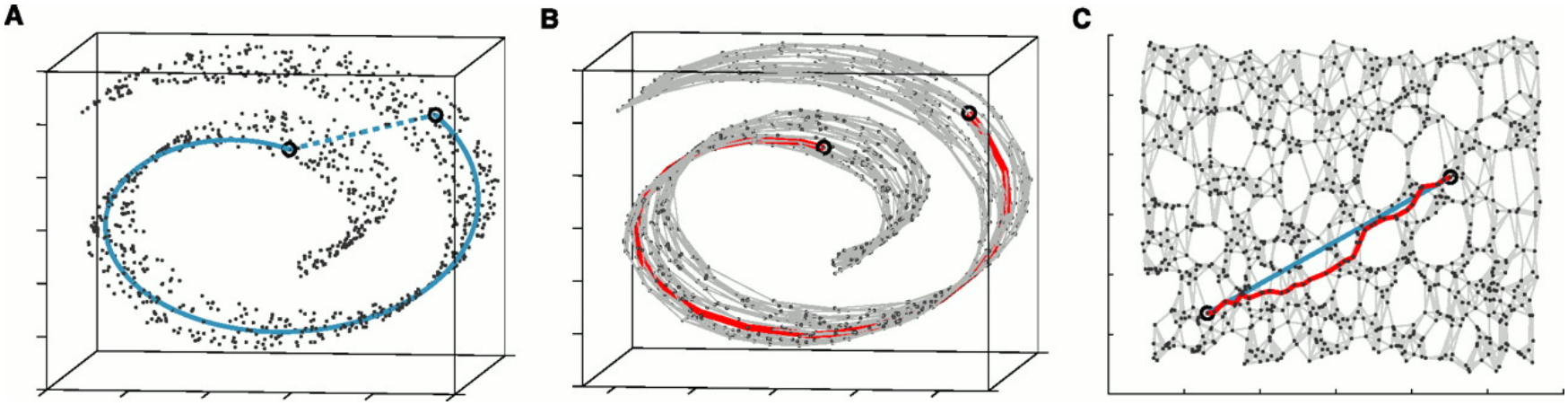
- Euclidean distance may be a poor measure of dissimilarity between points



- instead use geodesic distance, distance between points *along the manifold*

- this more accurately captures the neighbourhood relationships that should be preserved

Calculating Geodesic Distance



- without knowing the manifold, calculating geodesic distance is impossible
- for nearby points, geodesic distance $\approx$ Euclidean distance
- for faraway points, approximate geodesic distance by a sequence 'short hops' between neighbouring points

Isomap Algorithm

- Given input points $\{\mathbf{x}_i\}_{i=1}^n$, compute interpoint distances $\delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$
- Construct neighborhood graph, G, two possible ways:
  - edge $i, j \in G$ if i is is a K-nearest-neighbor of j
  - edge $i, j \in G$ if $\delta_{ij} < \epsilon$, ϵ some neighborhood size

  edge $i \leftrightarrow j$ gets weight δ_{ij}
- Compute shortest path distances d_{ij} between all pairs of nodes in G, using your favourite algorithm (Dijkstra, Floyd-Warshall)
- Use classical MDS on the shortest-path distances $\{d_{ij}\}$ to compute the images $\mathbf{y}_i$

Notes on Isomap

- if neighborhood graph has disconnected components, algorithm will fail
- is not very robust to noise