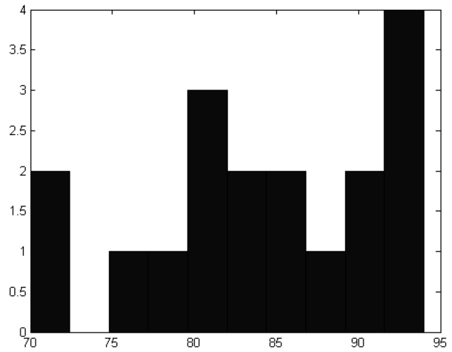
Announcements

- Presentation assignments – on class web page.
- Midterm

Histogram:



Manifold Learning

- Maybe a better term would be: distance preserving low-dimensional Euclidean representations that are suitable for some manifold data.
  - For Riemannian manifolds
    - Try to preserve local distances or geodesic distances.
    - With low-dimensional Euclidean embedding.

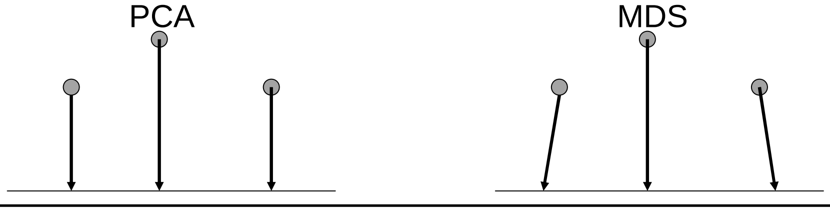
Multi-dimensional Scaling

- Given distances between points
  - In many applications (psychology) we have similarities, not points.
- Produce low-dimensional point set that preserves distances. If *x* are the initial points, and *y* are the low dimensional points, we want:

$$\min \sum_i \sum_j \left(\left\| x_i - x_j \right\| - \left\| y_i - y_j \right\| \right)^2$$

MDS vs. PCA

- PCA
  - linear projection of points,
  - can only decrease distances.
  - Tries to preserve points location.
- MDS can extend distances also.
- For low-dim points, they are equivalent
  - PCA preserves location and distances.



Distances and Inner products

Knowing pairwise distances between points is equivalent to knowing pairwise inner products.

Let $d_{ij} = \|x_i - x_j\|, \quad b_{ij} = x_i x_j^T, \quad B, D, X$ matrices.

$d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}$

Normalize, so that: $\sum_i x_i = \mathbf{0}$. So: $\sum_i b_{ij} = \sum_j b_{ij} = 0$

$\sum_i d_{ij}^2 = tr(B) + nb_{jj} \quad \sum_j d_{ij}^2 = tr(B) + nb_{ii} \quad \sum_{ij} d_{ij}^2 = 2ntr(B)$

So: $b_{ii} = \frac{\sum_j d_{ij}^2}{n} - \frac{\sum_{ij} d_{ij}^2}{2n}, \quad 2b_{ij} = \frac{\sum_j d_{ij}^2 + \sum_j d_{ij}^2}{n} - \frac{\sum_{ij} d_{ij}^2}{n} - d_{ij}^2$

Algorithm

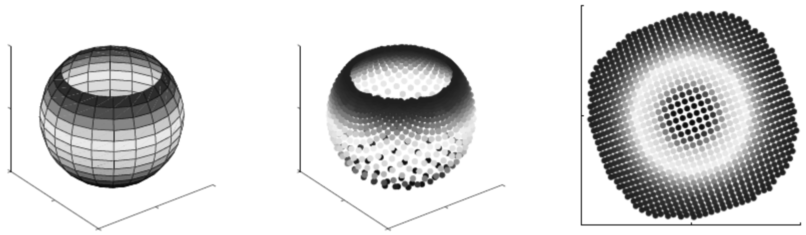
- Convert Distance matrix, D, to Inner Product matrix, B.
- Factor B = QAQ^T, where Q is an orthonormal (rotation) matrix, and A is diagonal.
  - Possible since B is symmetric.
  - Can do this with SVD.
- Use first d columns of QA$^{1/2}$ as d-dimensional points. These provide optimal approximation to inner products (and distances).

ISOMAP

- Like MDS, but tries to preserve geodesic distances on a manifold.
  - Compute near-neighbors
    - Assume Euclidean distances are appropriate for these.
  - Compute geodesic distances between all pairs of points
    - Geodesic distance taken as shortest path among set of local distances.
    - Can use shortest path algorithm.
  - Apply MDS to these distances.

LLE

- Embedding that only preserves local distances and angles.
- Inspired by manifold data, in which local distaces are ~ Euclidean.
- Also, local distances may be more meaningful/important.
- More modest goal than ISOMAP which tries to preserve all distances on manifold.



Local distances are well-preserved. Geodesic distances are not.

For example, the equator is ~ the yellow stripe.

LLE Algorithm

- For each point
  - Find nearest neighbors.
  - Rep. pt as weighted sum of neighbors.
- Approximate weights, points in low dimension.
- Error in reconstructing point using weights in low dimension indicates how much distances have changed.

Local Weights

$$\varepsilon = \left| x - \sum_j w_j \eta_j \right|^2 = \left| \sum_j w_j (x - \eta_j) \right|^2 = \sum_{jk} w_j w_k G_{jk} = \mathbf{w} G \mathbf{w}^T$$

with constraint $\sum_j w_j = 1$ and $G_{ij} = (x - \eta_j) \bullet (x - \eta_k)$

Writing this with Lagrange multipliers, we get :

$$\varepsilon = \mathbf{w} G \mathbf{w}^T + \lambda \left(1 - \sum_j w_j \right)$$

$$\frac{\partial e}{\partial \mathbf{w}} = 2\mathbf{w}G - \vec{\lambda} = 0, \quad 2\mathbf{w}G = \vec{\lambda}$$

Here, $\vec{\lambda}$ is a vector of all λ. Note that $2\mathbf{w}G$ scales with the magnitude of $\mathbf{w}$, so we can solve for $\mathbf{w}\mathrm{G} = 1$ and then scale $\mathbf{w}$ to sum to 1.

Low-dimensional approximation

Using all weights, reconstruction error is minimized (usually 0)

$$E(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2 = X^T (I - W)^T (I - W) X \equiv X^T M X$$

Note that M is a square matrix, but X is rectangular.
We want to find the low rank version of this to minimize the error.
This is done by choosing Y to correspond to the eigenvectors of M with smallest eigenvalues. (We ignore smallest; assuming Y's sum to 0 removes translation and produces eigenvector of all 1s, with eigenvalue of 0.