# L26: Advanced dimensionality reduction

**The "snapshot" PCA approach**

**Oriented Principal Components Analysis**

**Non-linear dimensionality reduction (manifold learning)**

- ISOMAP
- Locally Linear Embedding

# The "snapshot" PCA approach

## Problem definition

- Imagine that we have collected a small number of samples $x^{(p}$ $(p = 1 \dots P)$ each of which has a very high number of features $D$ (e.g., high-resolution images or 3D scans)
  - In the conventional PCA approach, we would compute the sample covariance matrix as follows

$$\hat{C} = \frac{1}{P-1}\sum_{p=1}^{P}\left(x^{(p} - \mu\right)\left(x^{(p} - \mu\right)^{T}$$

  - And then try to diagonalize it!

## This approach has two problems

- First, the sample covariance matrix will not be full-rank, in which case direct inversion is not possible (we'd need to use SVD)
- Second, $\hat{C}$ will be very large (e.g., 400MB for 100x100 images with double precision)
- However, we know that at most P eigenvectors will be non-zero…. Is there a better way to find these eigenvectors?

*[This material is based upon an unpublished manuscript by Sam Roweis, entitled "Finding the first few eigenvectors in a large space"]*

## The "snapshot" trick is based on the fact that the eigenvectors are linear combinations of the data samples

- Note that the eigenvectors capture the directions of variance, and there is no variance in directions normal to the subspace spanned by the data
- Thus, we will seek to express the PCA decomposition in a manner that depends only on the number of samples P

## Derivation

- Assume that the data has been centered by subtraction of its mean
- Then, the covariance matrix can be expressed as

$$\hat{C} = \frac{1}{P-1}\sum_{p=1}^{P}\left(x^{(p)}\right)\left(x^{(p)}\right)^{T}$$

- Since the eigenvectors are linear combinations of the data, we can then express them as

$$e^{j} = \sum_{p=1}^{P} \alpha_p^j x^{(p}$$

  - where $e^j$ denotes the $j^{th}$ eigenvector

- With this formulation, our goal becomes finding constants $\alpha_p^j$

– Since $e^j$ are the eigenvectors, they satisfy the condition

$$\hat{C}e^j = \lambda^j e^j$$

– Which after derivation becomes

$$\hat{C}\sum_{p=1}^{P}\alpha_p^j x^{(p} = \lambda^j \sum_{p=1}^{P}\alpha_p^j x^{(p} \quad \forall j$$

$$\left(\frac{1}{P-1}\sum_{p1=1}^{P}\left(x^{(p1)}\right)\left(x^{(p1)}\right)^T\right)\left(\sum_{p2=1}^{P}\alpha_{p2}^j x^{(p2}\right) = \lambda^j \sum_{p3=1}^{P}\alpha_{p3}^j x^{(p3} \quad \forall j$$

$$\frac{1}{P-1}\sum_{p1=1}^{P}\sum_{p2=1}^{P}\alpha_{p2}^j\left(x^{(p1)}\right)\left(x^{(p1)}\right)^T x^{(p2} = \lambda^j \sum_{p3=1}^{P}\alpha_{p3}^j x^{(p3} \quad \forall j$$

– We now define matrix $R$, which is the sample inner product of pairs of samples (i.e., the covariance matrix is the sample outer product)

$$R_{p1p2} = \frac{1}{P-1}\left(x^{(p1)}\right)^T\left(x^{(p2)}\right)$$

- Substituting this matrix into the previous expression yields

$$\sum_{p1=1}^{P}\sum_{p2=1}^{P}\alpha_{p2}^{j}\left(x^{(p1}\right)R_{p1p2}=\lambda^{j}\sum_{p3=1}^{P}\alpha_{p3}^{j}x^{(p3}$$

- which, merging subindices $p1$ and $p3$, and moving terms to the LHS, yields

$$\sum_{p1=1}^{P}\sum_{p2=1}^{P}x^{(p1}\left(\alpha_{p2}^{j}R_{p1p2}-\lambda^{j}\alpha_{p1}^{j}\right)=0$$

- This condition can be met by finding $\alpha_{p1}^{j}$ such that*

$$\left(\alpha_{p2}^{j}R_{p1p2}-\lambda^{j}\alpha_{p1}^{j}\right)=0\quad\forall j,p1,p2$$

- which can be written as

$$R\alpha^{j}=\lambda^{j}\alpha^{j}\quad\forall j$$

- Therefore, the $P$-dim vectors $\alpha_{p}^{j}$ are the eigenvectors of matrix $R$, which can be found in a conventional matter since $R$ has size $P\times P$
  - Once $\alpha_{p}^{j}$ have been found, the actual eigenvectors of the data $e^{j}$ are obtained by a weighted sum of the training samples

$$e^{j}=\sum_{p=1}^{P}\alpha_{p}^{j}x^{(p}$$

*This is one out of possibly many solutions (i.e., ways of expressing eigenvectors as linear combinations of the samples), but one that makes the problem very easy to solve, as we see next

# Oriented principal components analysis

## OPCA is a generalization of PCA that uses a generalized eigenvalue problem with two covariance matrices

- The cost function maximized by OPCA is the signal-to-noise ratio between a pair of high-dimensional signals $u$ and $v$

$$J_{OPCA}(w) = \frac{E[w^T u]^2}{E[w^T v]^2} = \frac{w^T S_u w}{w^T S_v w} \quad where \quad S_u = E[uu^T]; S_v = E[vv^T]$$

- Since $S_u$ and $S_v$ are symmetric, all the generalized eigenvectors are real, and can be sorted by decreasing generalized eigenvalues
- Note that the generalized eigenvectors will NOT be orthogonal but instead will meet the constraint

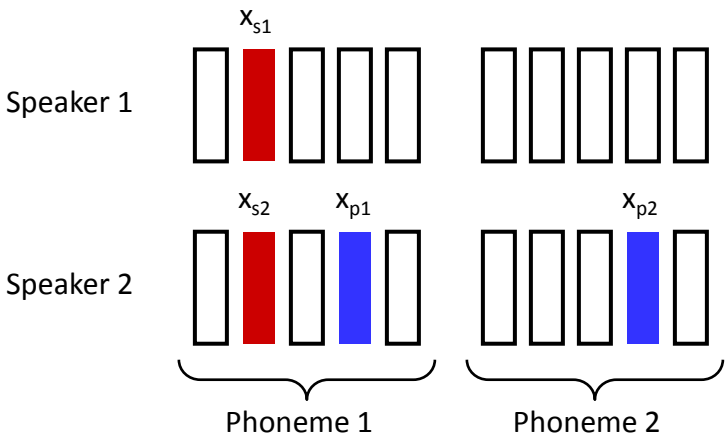
$$e^T S_u e = e^T S_v e = 0 \quad \forall i \neq j$$

- The term "oriented" is due to the fact that $e$ is similar to the ordinary principal direction of $u$, except that it is oriented towards the least principal direction of $v$
  - In other words, $S_v$ "steers" $e$ away from the directions of high energy in $v$
  - If $v$ is white noise, then there is no steering, and OPCA is identical to the PCA solution

# OPCA: an example

## Consider the problem of speech/speaker recognition

- Short-time speech signals are typically represented by a feature vector $x$, which capture some of its stationary spectral properties (e.g, LPC, cepstrum)
- Assume that we have data from two speakers (s1, s2) uttering two different phonemes (p1, p2), as illustrated below




- We are interested in finding projection vectors $w$ of the feature vector that maximizes the signal to noise ratio, signal being linguistic information (LI) and noise being speaker information (SI)

$$SNR = \frac{LI}{SI}$$

[Malayath et al., 1997]

- We then extract feature vectors $x_{p1}$ and $x_{p2}$ from the same speaker but different phonemes; the difference between these two vectors will only contain linguistic information

$$u = x_{p1} - x_{p2}$$

- We also extract feature vectors $x_{s1}$ and $x_{s2}$ from the same phoneme but different speakers; the difference between these two vectors will only contain speaker information

$$v = x_{s1} - x_{s2}$$

- We define the signal and noise covariance matrices as

$$S_u = E[(u - \bar{u})(u - \bar{u})^T]$$
$$S_v = E[(v - \bar{v})(v - \bar{v})^T]$$

- And the SNR becomes

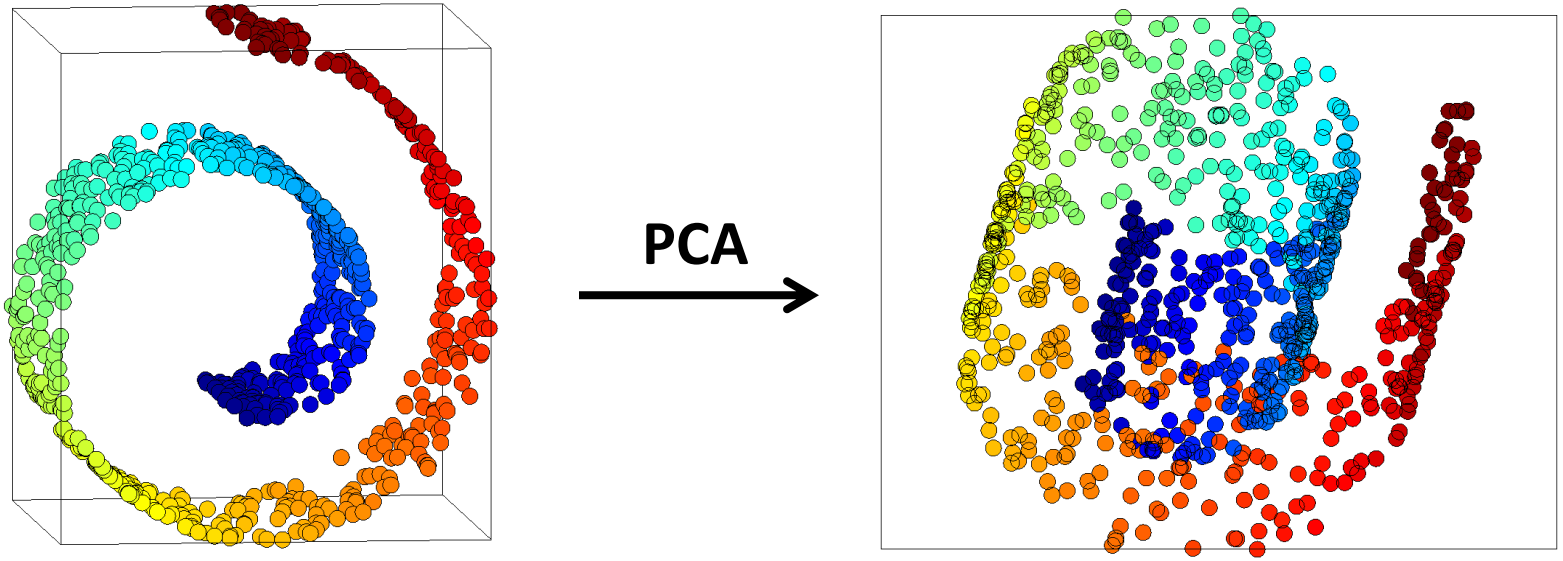
$$SNR = \frac{LI}{SI} = \frac{E[u^T w]^2}{E[v^T w]^2} = \frac{w^T S_u w}{w^T S_v w}$$

- Thus, the projection vectors $w$ that maximize the SNR are derived using an OPCA formulation
  - These vectors will span a speaker-independent subspace

# Non-linear dimensionality reduction

**PCA, LDA and their variants perform a global transformation of the data (rotation/translation/rescaling)**

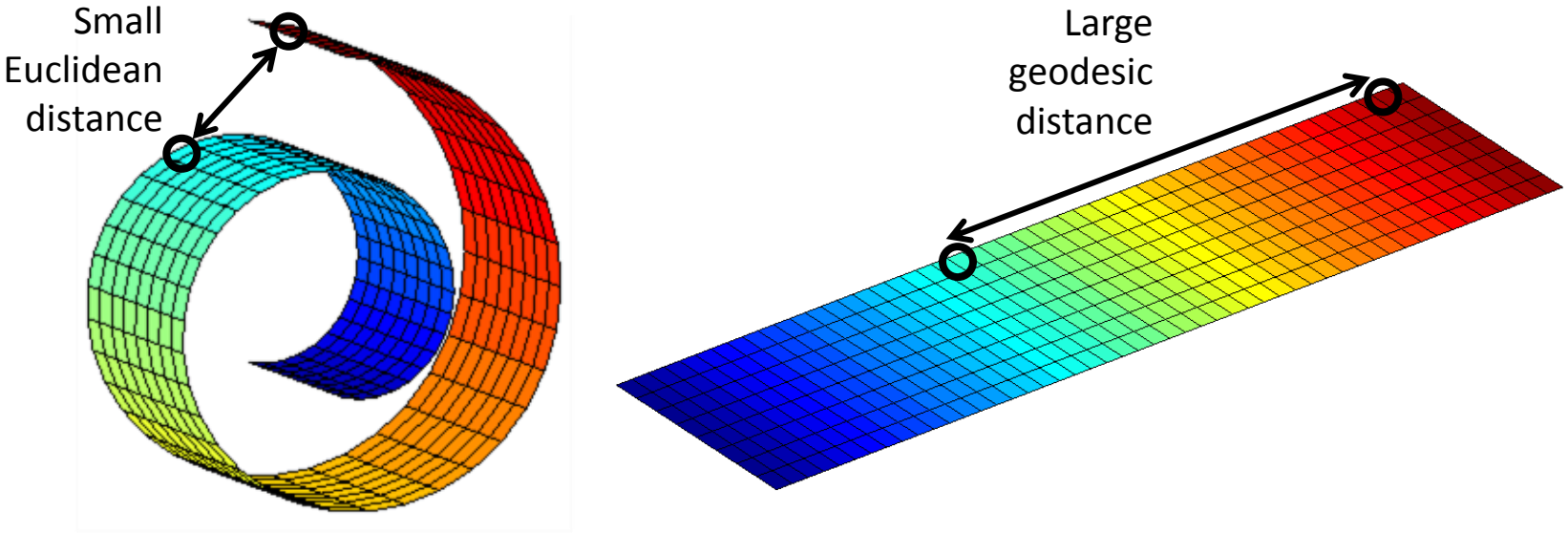
- These techniques assume that most of the information in the data is contained in a linear subspace
- What do we do when the data is actually embedded in a non-linear subspace (i.e., a low-dimensional manifold)?




From http://www.cs.unc.edu/Courses/comp290-090-s06

## One possible solution (ISOMAP)

- To find the embedding manifold, find a transformation that preserves the geodesic distances between points in the high-dimensional space
  - This approach is related to multidimensional scaling (e.g., Sammon's mapping; L10), except for MDS seeks to preserve the Euclidean distance in the high-dimensional space
- The issue is how to compute geodesic distances from sample data



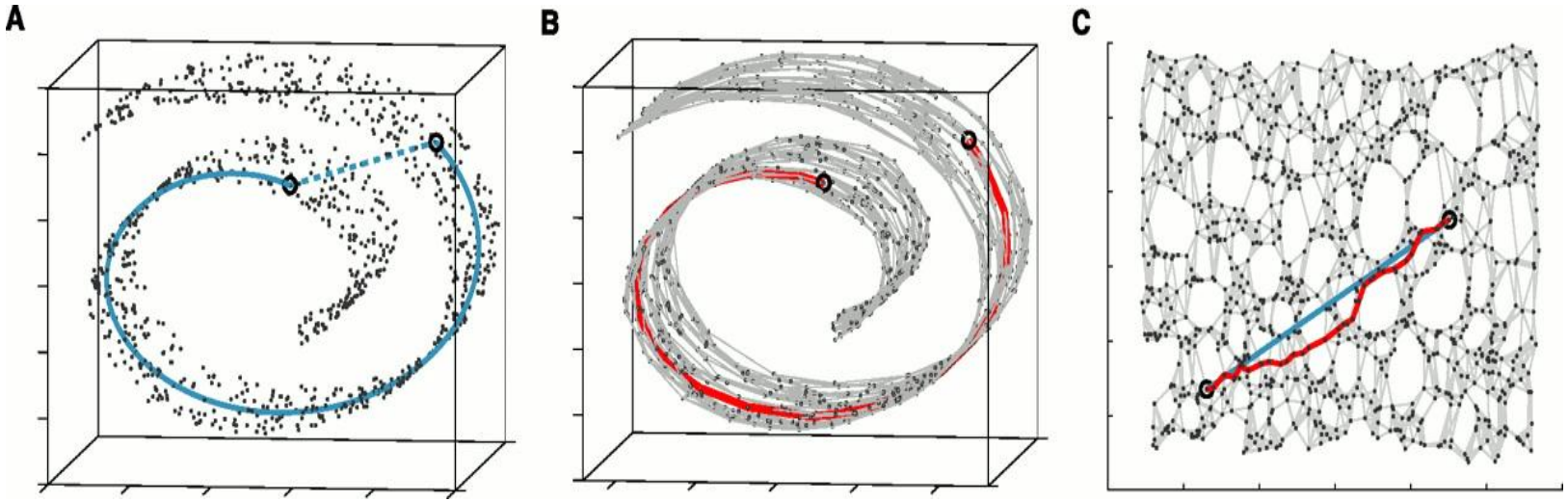
# ISOMAP

## ISOMAP [Tenenbaum et al., 2000] is based on two simple ideas

- For neighboring samples, Euclidean distance provides a good approximation to geodesic distance
- For distant points, geodesic distance can be approximated with a sequence of steps between clusters of neighboring points

## ISOMAP operates in three steps

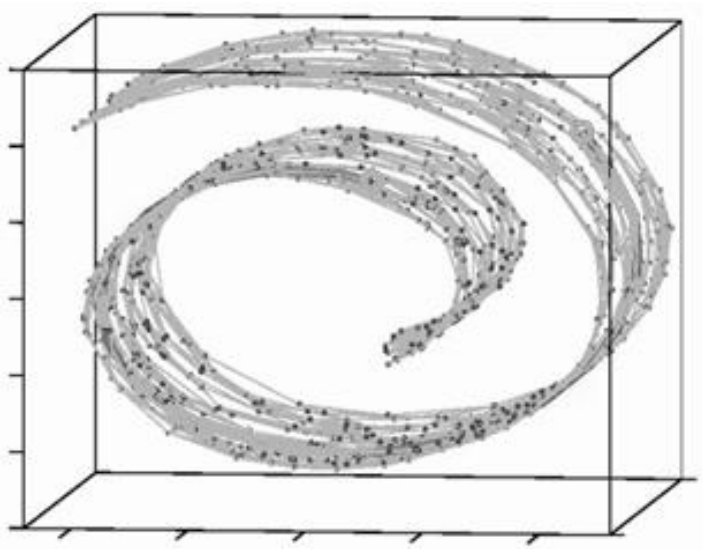
- Find nearest neighbors to each sample
- Find shortest paths (e.g., Dijkstra)
- Apply MDS




*[Tenenbaum et al., 1997]*

## STEP 1

- Determine which points are neighbors in the manifold, based on the distances $d_X(i,j)$ in the input space $X$
- This can be performed in two different ways
  - Connect each point to all points within a fixed radius $\epsilon$
  - Connect each point to all of its K nearest neighbors
- These neighborhood relations are represented as a weighted graph G, each edge of weight $d_X(i,j)$ between neighboring points
- Result:





*This discussion is borrowed from [Tenenbaum et al., 1997]*

## STEP 2

- Estimate the geodesic distances $d_M(i,j)$ between all pair of points on the manifold $M$ by computing their shortest path distances $d_G(i,j)$ in the graph G
- This can be performed, e.g., using Dijkstra's algorithm

## STEP 3

- Find $d$-dim embedding $Y$ that best preserves the manifold's estimated distances
  - In other words, apply classical MDS to the matrix of graph distances $D_G = \{d_G(i,j)\}$
- The coordinate vectors $y_i$ are chosen to minimize the following cost function

$$E = \|\tau(D_G) - \tau(D_Y)\|_{L^2}$$

- where $D_Y$ denotes the matrix of Euclidean distances $\{d_Y(i,j) = \|y_i - y_j\|\}$, and operator $\tau$ converts distances to inner products

$$\tau = -HSH/2$$

- where $S$ is the matrix of squared distances $\{S_{ij} = D_{ij}^2\}$, and $H$ is a centering matrix, defined as

$$H = I - \frac{1}{N} ee^T; \quad e = [111 \dots 1]^T$$

- It can be shown that the global minimum of $E$ is obtained by setting the coordinates $y_i$ to the top $d$ eigenvectors of the matrix $\tau(D_G)$

**(1) Construct neighborhood graph**

(a) Define graph $G$ by connecting points $i$ and $j$ if they are [as measured by $d_X(i,j)$]

closer than epsilon (*epsilon -Isomap*), or
if *i* is one of the K nearest neighbors of *j* (*K-Isomap*).

(b) Set edge lengths equal to $d_X(i,j)$

**(2) Compute shortest paths**

(a) Initialize

$d_G(i,j) = d_X(i,j)$ if $i,j$ are linked by an edge;
$d_G(i,j) = \infty$ otherwise.

(b) For $k = 1, 2 \ldots N$, replace all entries $d_G(i,j)$ by $\min\{d_G(i,j), d_G(i,k) + d_G(k,j)\}$

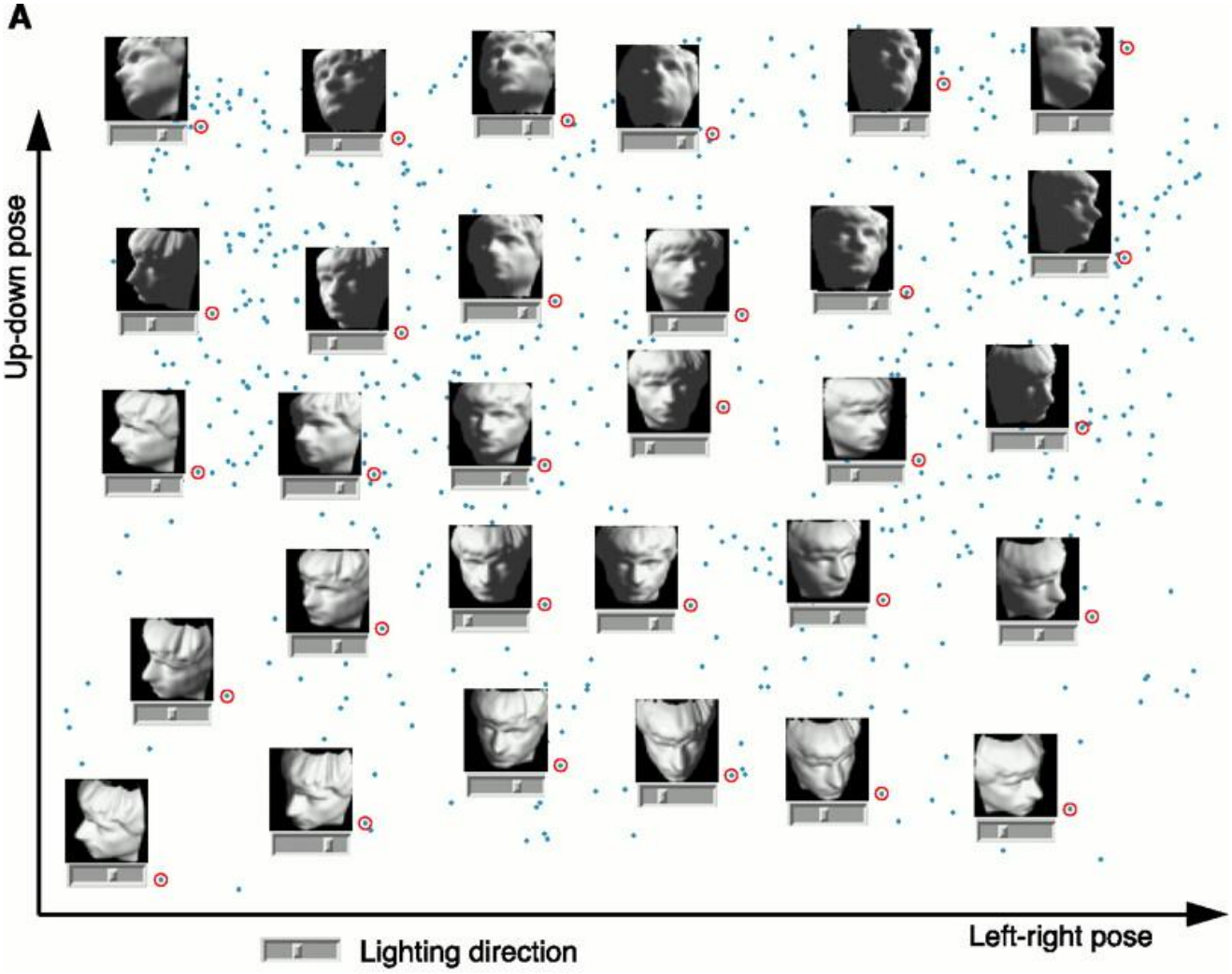
(c) Matrix $D_G = \{d_G(i,j)\}$ will contain the shortest path distances between all pairs of points in $G$

**(3) Construct d-dimensional embedding**

(a) Let $\lambda_p$ be the $p^{th}$ eigenvalue (in decreasing order) of matrix $\tau(D_G)$, and $v_p^i$ be the $i^{th}$ component of the $p^{th}$ eigenvector

(b) Set the $p^{th}$ component of the *d*-dimensional coordinate vector $y_i$ equal to $\sqrt{\lambda_p} v_p^i$

# ISOMAP results




*[Tenenbaum et al., 1997]*

# ISOMAP discussion

**ISOMAP is guaranteed asymptotically to recover the true dimensionality and geometric structure of a certain class of Euclidean manifolds**

- These are manifolds whose intrinsic geometry is that of a convex region of Euclidean space, but whose geometry in the high-dimensional space may be highly folded, twisted or curved
  - Intuitively, in 2D, these include any physical transformations one can perform on a sheet of paper without introducing tears, holes, or self-intersections
- For non-Euclidean manifolds (hemispheres or tori), ISOMAP will still provide a globally-optimal low-dimensional representation

**These guarantees are based on the fact that, as the number of samples increases, the graph distances $d_G(i,j)$ provide increasingly better approximations of the intrinsic geodesic distances $d_M(i,j)$**

- However, this proof has limited application [Carreira-Perpiñán, in press] because
  - Data in high-dimensions is scarce (!) and
  - Computational complexity would preclude the use of large datasets anyway

**Mapping**

- Note that ISOMAP does not provide a mapping function $Y = f(X)$
- One could however be learned from the pairs $\{X_i, Y_i\}$ in a supervised fashion

# Locally Linear Embedding

## LLE uses a different strategy than ISOMAP to recover the global non-linear structure

- ISOMAP estimates pairwise geodesic distances for all points in the manifold
- Instead, LLE uses only distances within locally linear neighborhoods

## Intuition

- Assume that the dataset consists of $N$ vectors $x_i$, each having $D$ dimensions, sampled from an underlying manifold
- Provided that there is sufficient data (i.e., the manifold is well sampled) one can expect that each point and its neighbors will lie on a linear patch of the manifold

## Approach

- LLE solves the problem in two stages
  - First, compute a linear approximation of each sample in the original space
  - Second, find the coordinates in the manifold that are consistent with the previous linear approximation

This discussion is borrowed from [Roweis and Saul, 2000]

## Algorithm (part 1)

- The local geometry of these patches is modeled by linear weights that reconstruct each data point as a linear combination of its neighbors
- Reconstruction errors are measured with the L2 cost function

$$\epsilon(W) = \sum_{i=1}^{N} \left| X_i - \sum_j W_{ij} X_j \right|^2$$

  - where weights $W_{ij}$ measure the contribution of the $j^{th}$ example to the reconstruction of the $i^{th}$ example
- Weights $W_{ij}$ are minimized subject to two constraints
  - Each data point is reconstructed only from its neighbors
  - Rows of the weight matrix sum up to one: $\sum_j W_{ij} = 1$
- These constraints ensure that, for any particular sample, the weights are invariant to translation, rotation or scaling
- This problem can be solved using least-squares (details next)

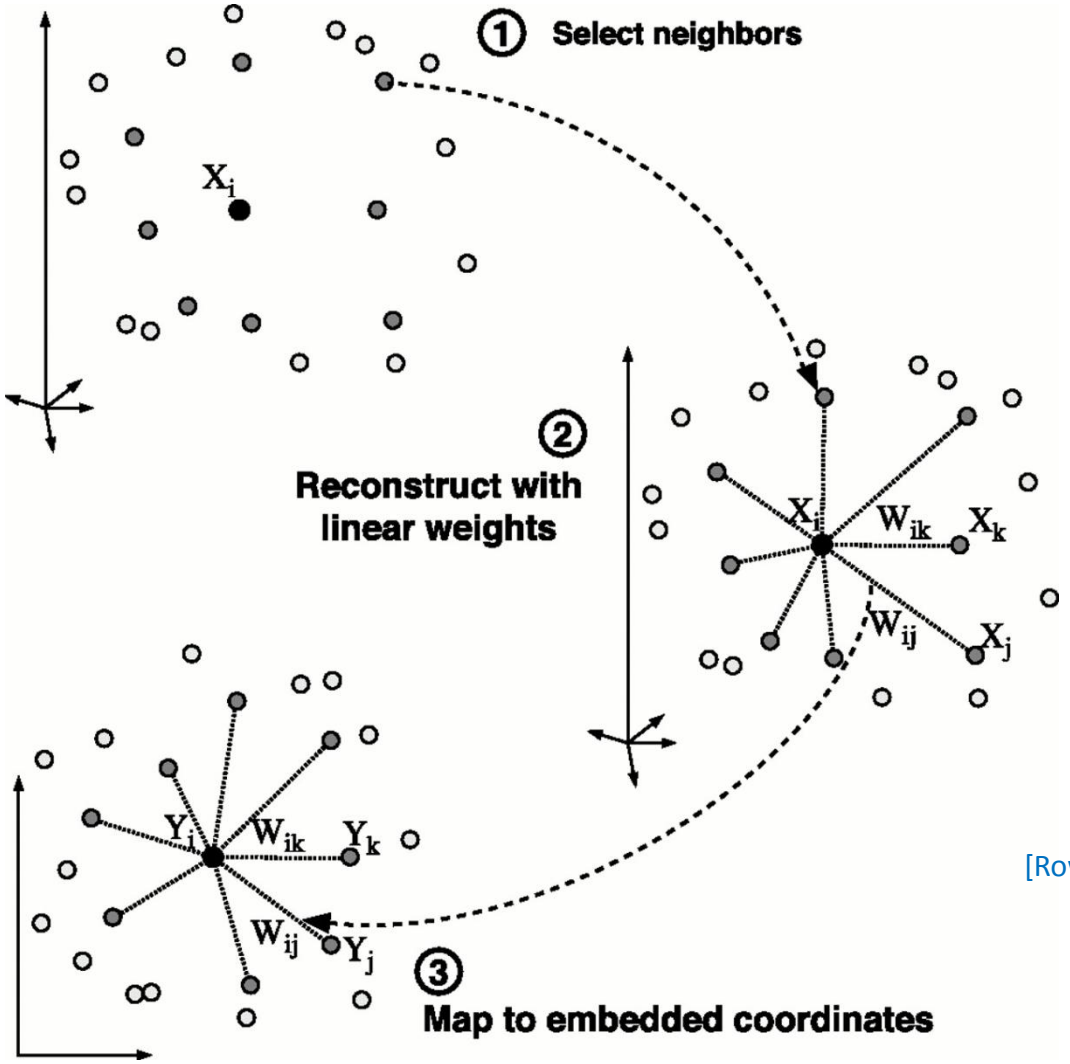
## Algorithm (part 2)

- Given that these weights reflect intrinsic properties of the local geometry, we expect that they will also be valid for local patches in the manifold
- Therefore, we seek to find $d$-dimensional coordinates $Y_i$ that minimize the following cost function

$$\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2$$

- Note that this function is similar to the previous one, except that here $W_{ij}$ are fixed and we solve for $Y_i$
- This minimization problem can be solved as a sparse $N \times N$ eigenvalue problem, whose bottom $d$ non-zero eigenvectors are the orthogonal coordinates of the data in the manifold (details follow)

# LLE summary




[Roweis and Saul, 2000]

# LLE: solving for $W_{ij}$

**The linear weights $W_{ij}$ can be solved as a constrained LS problem**

- Consider a particular sample $x$ with K nearest neighbors $\eta_j$ and reconstruction weights $w_j$ (that sum up to one)
- These weights can be found in three steps:
- **STEP 1**: Evaluate inner product between neighbors to compute the neighborhood correlation matrix $C_{jk}$ and its inverse $C^{-1}$

$$C_{jk} = \eta_j^T \eta_k$$

- **STEP 2**: Compute Lagrange multiplier $\lambda$ that enforces the constraint $\sum_j w_j = 1$

$$\lambda = \frac{1 - \sum_{jk} C_{jk}^{-1}(x^T \eta_k)}{\sum_{jk} C_{jk}^{-1}}$$

- **STEP 3**: Compute the reconstruction weights as

$$w_j = \sum_k C_{jk}^{-1}(x^T \eta_k + \lambda)$$

  - NOTE: If $C$ is singular, regularize by adding a small multiple of the identity matrix
- Since this 3-step process requires matrix inversion, a more efficient solution is to
  - First, solve the linear system of equations $\sum_j C_{jk} w_k = 1$, and
  - Then, rescale the weights so they sum up to one (which yields the same results)

*[Roweis and Saul, 2000; Saul and Roweis, 2001]*

# LLE: solving for $Y_i$

**The embedding vectors $Y_i$ are found by minimizing the cost function**

$$\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2$$

- To make the optimization problem well-posed we introduce two constraints
  - Since coordinates $Y_i$ can be translated without affecting the cost function, we remove this degree of freedom by imposing that they are centered

$$\sum_j Y_j = 0$$

- To avoid degenerate solutions, we constraint the embedding vectors to have unit covariance matrix

$$\frac{1}{N} \sum_i Y_i Y_i^T = I$$

- This allows the cost function to be expressed in a quadratic form involving inner products of the embedding vectors

$$\Phi(Y) = \sum_{ij} M_{ij} \left( Y_i^T Y_j \right)$$

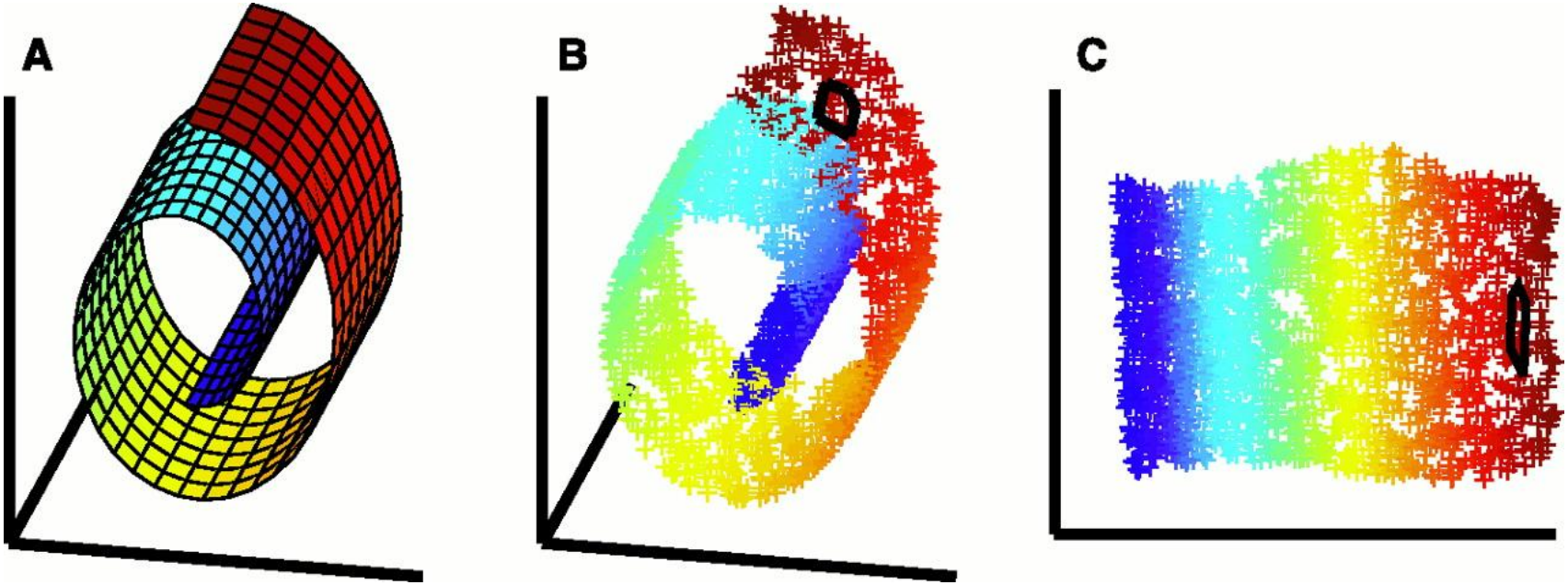
- where

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}$$

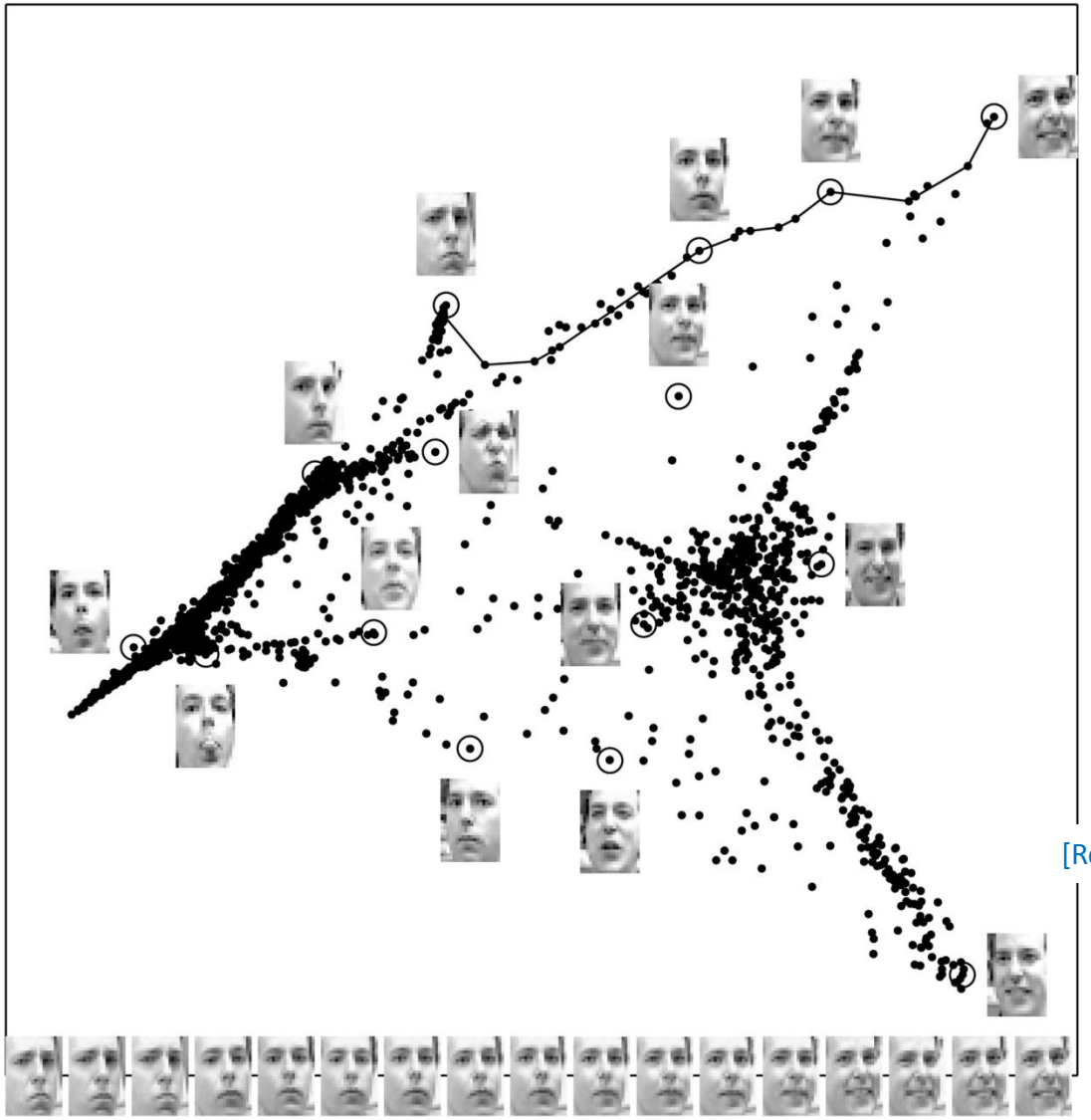
**It can be shown that the optimal embedding is found by computing the bottom $d+1$ eigenvectors of matrix $M$**

- We discard the bottom eigenvector, which is the unit vector
  - This eigenvector represents a free translation mode with eigenvalue zero
  - Discarding this eigenvector enforces the constraint that the embedding coordinates have zero mean
- The remaining $d$ eigenvectors form the $d$ embedding coordinates found by LLE

# LLE examples




*[Roweis and Saul, 2000]*



[Roweis and Saul, 2000]

# LLE discussion

**LLE is simple and attractive, but shares some of the limitations of MDS methods**

- Sensitivity to noise
- Sensitivity to non-uniform sampling of the manifold
- Does not provide a mapping (though one can be learned in a supervised fashion from the pairs $\{X_i, Y_i\}$
- Quadratic complexity on the training set size
- Unlike ISOMAP, no robust method to compute the intrinsic dimensionality, and
- No robust method to define the neighborhood size K