# Topic 2: DIMENSIONALITY REDUCTION

CMSC 35400/STAT 37710 Machine Learning

Risi Kondor, The University of Chicago

# Dimensionality reduction

In ML data points are often represented as high dimensional real valued vectors

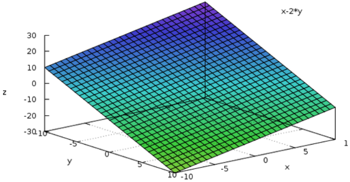
$$\mathbf{x} = (x_1, x_1, x_3, \ldots, x_d)^\top \in \mathbb{R}^d.$$

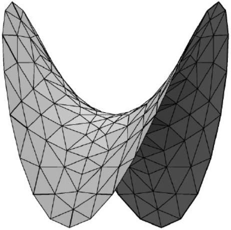
The individual dimensions are called **features** (**attributes)**.

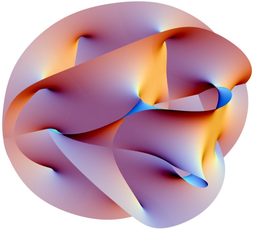
Example: Pixels of an image, a music file, etc.

But is the problem intrinsically high dimensional??? Often we can convert high dimensional problems to lower dimensional ones without losing too much information.

# Dimensionality reduction





- Real world data often lie on or near lower dimensional structures (manifolds). (Really?)
    - Variables (features) may be correlated or dependent.
    - Physical systems have a small number of degrees of freedom (e.g., pose and lighting in Vision).
- IDEA: find the manifold and restrict learning algorithm to it.

# Dimensionality reduction

**Advantages:**

- **Visualization:** humans can only imagine things in 2D or 3D.
- **Computational efficiency:** learning algorithms work faster in low dimensions.
- **Better performance:** the projection might eliminate noise.
- **Interpretability:** the vectors spanning the subspace might have interesting interpretations.

# Dimensionality reduction

Dimensionality reduction is a typical **unsupervised learning** task. Two types:

- Linear:
  - Principal Component Analysis (PCA)
- Nonlinear (“manifold learning”):
  - Multidimensional scaling
  - Locally linear embedding
  - Isomap
  - Laplacian Eigenmaps
  - Stochastic neighbor embedding
  - etc.

# Fact 1

If a matrix $A \in \mathbb{R}^{d \times d}$ is symmetric, then its (normalized) eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ form an orthonormal basis for $\mathbb{R}^d$.

Note: If the eigenvalues are not distinct, then the eigenvectors are not unique. However, there is always some choice of eigenvectors which forms an orthonormal basis.

# Fact 2 (Rayleigh quotient)

Let $\mathbf{v}_1, \ldots, \mathbf{v}_d$ be the normalized eigenvectors of a symmetric matrix $A \in \mathbb{R}^{d \times d}$ and let $\lambda_1 < \lambda_2 < \ldots < \lambda_d$ be the corresponding eigenvalues. Then

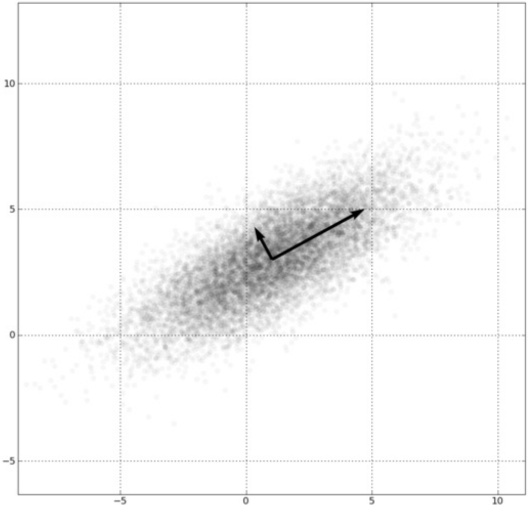
$$\underset{\mathbf{w} \in \mathbb{R}^d \setminus \{0\}}{\operatorname{argmin}} \frac{\mathbf{w}^\top A \, \mathbf{w}}{\|\mathbf{w}\|^2} = \mathbf{v}_1.$$

Similarly,

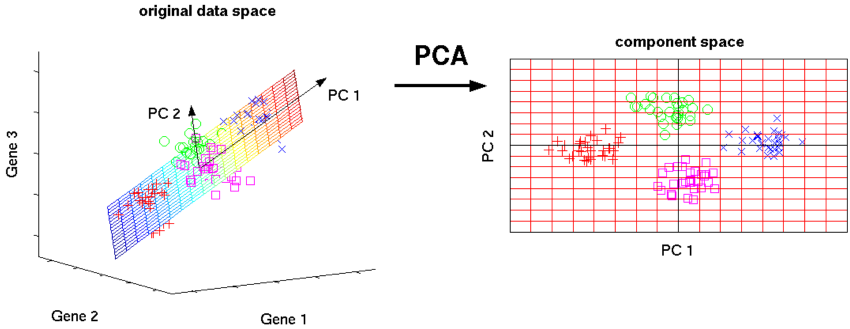
$$\underset{\mathbf{w} \in \mathbb{R}^d \setminus \{0\}}{\operatorname{argmax}} \frac{\mathbf{w}^\top A \, \mathbf{w}}{\|\mathbf{w}\|^2} = \mathbf{v}_d.$$

# Principal Component Analysis

# The principal directions in data



# Finding the principal subspace




How can we find the most relevant subspace for the data? By finding a basis for it. The individual basis vectors are called the **principal components**.

# The first principal component

Given a data set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of $n$ vectors in $\mathbb{R}^d$, what is the direction that is most informative for this data?

1. First center the data: $\mathbf{x}_i \leftarrow \mathbf{x}_i - \boldsymbol{\mu}$ where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ .
2. Find the unit vector $\boldsymbol{p}_1$ that is the solution to

$$\boldsymbol{p}_1 = \arg\max_{\|\mathbf{v}\|=1} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i \cdot \mathbf{v})^2. \tag{1}$$

This vector is called the first **principal component** of the data.

# Finding the first principal component

**Theorem.** The first principal component, $\boldsymbol{p}_1$, is the eigenvector $\mathbf{v}_d$ of the **sample covariance matrix**

$$\widehat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top$$

with largest eigenvalue.

**Proof.**

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i \cdot \mathbf{v})^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}^\top \mathbf{x}_i)(\mathbf{x}_i^\top \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}^\top(\mathbf{x}_i \mathbf{x}_i^\top)\mathbf{v} =$$

$$\frac{1}{n}\mathbf{v}^\top\left(\textstyle\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top\right)\mathbf{v} = \mathbf{v}^\top \widehat{\Sigma}\mathbf{v}.$$

Since $\|\mathbf{v}\| = 1$, (1) is equivalent to the Rayleigh quotient optimization problem

$$\boldsymbol{p}_1 = \underset{\mathbf{v}\in\mathbb{R}^d\setminus\{0\}}{\arg\max} \frac{\mathbf{v}^\top \widehat{\Sigma}\,\mathbf{v}}{\|\mathbf{v}\|},$$

so $\boldsymbol{p}_1$ is indeed the eigenvector $\mathbf{v}_d$ of $A$ with largest eigenvalue.

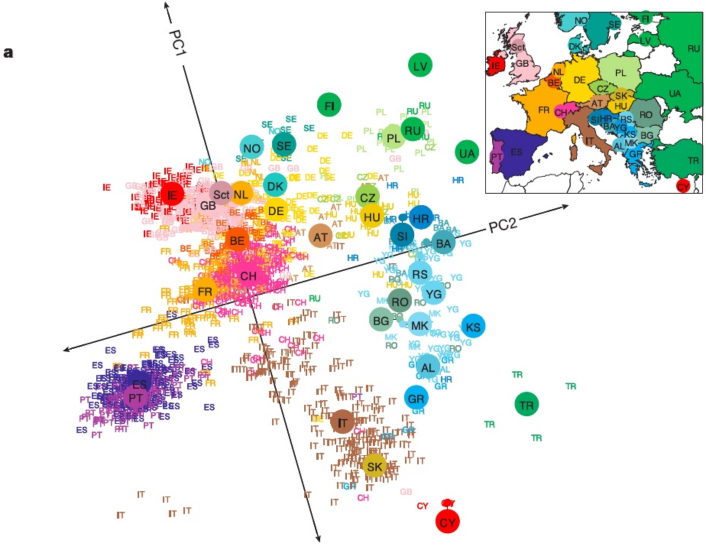
# Finding further principal components

Recall that $\widehat{\Sigma}$ can be written as

$$\widehat{\Sigma} = \sum_{i=1}^{d} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top.$$

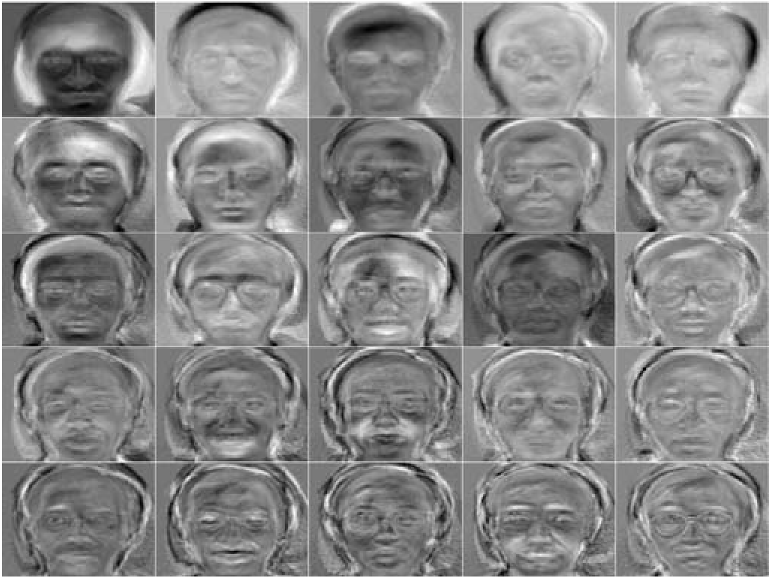
After we've found the first principal component $\boldsymbol{p}_1 = \mathbf{v}_d,$ project the data to $\operatorname{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_{d-1}\}$. This just removes $\lambda_d \mathbf{v}_d \mathbf{v}_d^\top$ from the sum. So the second principal component is $\boldsymbol{p}_2 = \mathbf{v}_{d-1}$, and so on.

# DNA data




[Matthew Stephens, John Novembre]

# Eigenfaces

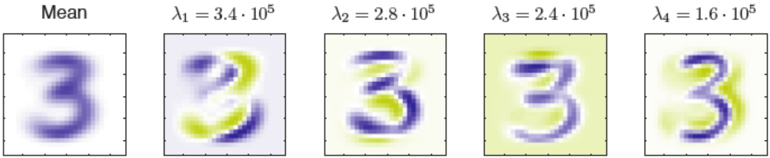


[Christopher de Coro]

# Reconstruction from eigenfaces

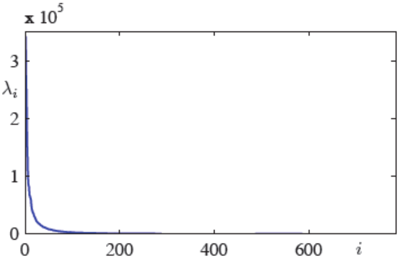
[Christopher de Coro]

# Example: digits




These are the EVectors for the four largest EValues.

- Often the eigenvalues drop off rapidly (e.g., exponentially)
- Sometimes there is a sharp drop somewhere, called the **spectral gap** $\rightarrow$ natural place to put cut-off




[Source: Peter Orbanz]

# Summary of PCA

Advantages:

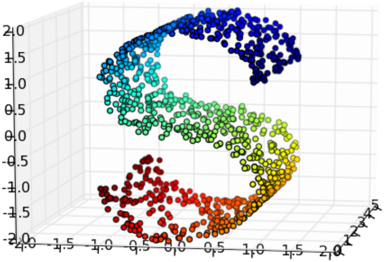
- Finds best projection
- Rotationally invariant

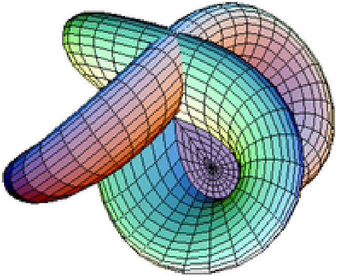
Disadvantages:

- Full PCA is expensive to compute
- Components not sparse
- Sensitive to outliers
- Linear

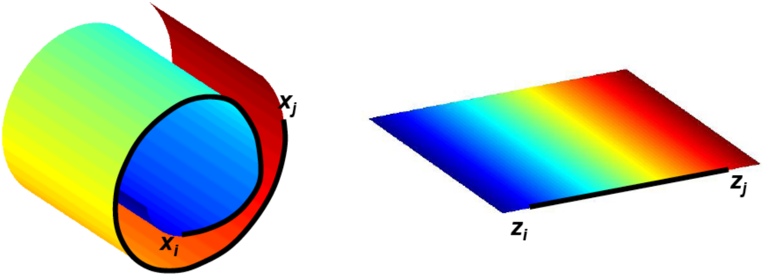
# NONLINEAR DIMENSIONALITY REDUCTION

- If the data lies close to a linear subspace of $\mathbb{R}^d$, PCA can find it.
- But what if the data lies on a nonlinear **manifold**? Data which at first looks very high dimensional often really has low dimensional structure.


2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0



# General principle




$$d_X(x_i, x_j) \approx |z_i - z_j|$$

Find a map $\phi \colon \mathbb{R}^d \to \mathbb{R}^p$ that maps the manifold to a lower dimensional Euclidean space in a way that preserves local distances as much as possible (some methods can only map individual data points not the whole of $\mathbb{R}^d$).

Question: Can this always be done? Depends on the topology.

# Methods

- Multidimensional Scaling
- Isomap
- Locally Linear Embedding
- Laplacian Eigenmaps
- SNE, etc..

# Multidimensional scaling (MDS)

# Classical MDS

- **Input:** $n$ data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$.
- **Output:** $n$ corresponding lower dimensional points $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^p$ (with $p \ll d$) that minimize the so-called *strain*

$$\mathcal{E}_{\mathrm{CMDS}} = \|D - D^*\|^2_{\text{Frob}} = \sum_{i,j} (D_{i,j} - D^*_{i,j})^2,$$

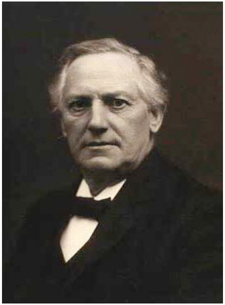
where $D_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ and $D^*_{i,j} = \|\mathbf{y}_i - \mathbf{y}_j\|^2$.

# The Gram matrix

The **Gram matrix** of $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is the $n \times n$ positive semidefinite matrix

$$G_{i,j} = \mathbf{x}_i \cdot \mathbf{x}_j .$$

(Again, we assume that the data has been centered, i.e., $\sum_i \mathbf{x}_i = 0$ .)



Jørgen Pedersen Gram
1850–1916

Exercise: Prove that if $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, then $\operatorname{rank}(G) \le d$.

# Classical MDS

**Proposition 1.** The CMDS problem can equivalently be written as minimizing

$$\mathcal{E} = \| G - G^* \|_{\mathrm{Frob}}^2,$$

where $G$ is the centered Gram matrix of $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $G^*$ is the Gram matrix of $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$.

**Approach:**

1. Compute the centered Gram matrix $G$.
2. Solve $G^* = \mathrm{argmin}_{\tilde{G} \succeq 0,\ \mathsf{rank}(\tilde{G}) \leq p} \| \tilde{G} - G \|_{\mathrm{Frob}}^2$.
3. Find $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n \in \mathbb{R}^p$ with Gram matrix $G^*$.

# Classical MDS

**Proposition 2.** Let $G = Q\Lambda Q^\top$ be the eigendecomposition of the Gram matrix with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ and $\lambda_1 \geq \ldots \geq \lambda_d$. Then

$$\underset{\tilde{G} \succeq 0,\ \text{rank}(\tilde{G}) \leq p}{\mathrm{argmin}} \| \tilde{G} - G \|_{\mathrm{Frob}}^2 = Q\Lambda^* Q^\top,$$

where $\Lambda^* = \mathrm{diag}(\lambda_1, \ldots, \lambda_p, 0, 0, \ldots)$.

Exercise: Prove this proposition.

# Gram $\rightarrow$ Data

**Proposition 3.** Let $G \in \mathbb{R}^{n \times n}$ be a p.s.d. matrix of rank $d$ with eigen-decomposition

$$G = Q \Lambda Q^\top .$$

Let $\mathbf{x}_i = [Q \Lambda^{1/2}]_{i,*}^\top$. Then the Gram matrix of $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is $G$.

**Notation:**

- $M_{i,*}$ denotes the $i$'th row of $M$.
- Given $D = \mathrm{diag}(d_1, \ldots, d_m)$, $D^p := \mathrm{diag}(d_1^p, \ldots, d_m^p)$.

Exercise: Prove this proposition.

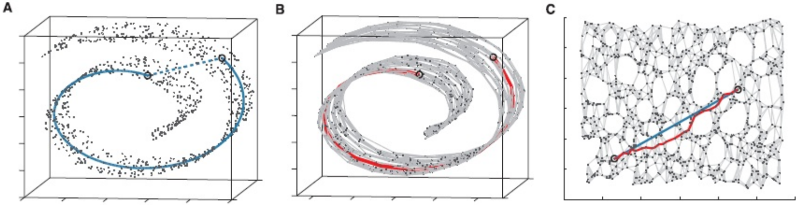
# Summary of Classical MDS

1. Compute the centered Gram matrix $G$ (see homework for how).
2. Compute the eigendecomposition $Q\Lambda Q^\top$ of $G$.
3. Assuming $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ and $\lambda_1 \geq \ldots \geq \lambda_d$, set $\Lambda^* = \mathrm{diag}(\lambda_1, \ldots, \lambda_p, 0, 0, \ldots)$ and $G^* = Q\Lambda^* Q^\top$.
4. Let $\mathbf{y}_i = [Q\Lambda^{1/2}]_{i,*}^\top$.

# Isomap

Tenenbaum, de Silva & Langford, 2000

# Isomap




1. Convert data into a graph (e.g., a symmetrized $k$-nn graph).
2. Compute all pairs shortest path distances.
3. Use MDS to compute $\phi\colon \mathbb{R}^d \to \mathbb{R}^p$ that tries to preserve these distances.

Underlying assumptions:

1. Data lies on a manifold.
2. Goedesic distance on manifold is approximated by distance in the graph.
3. The optimal embedding preserves these distances as much as possible.

# Shortest path distances

Let $\mathcal{G}$ be a weighted graph with vertex set $\{1, 2, \ldots, n\}$, and a distance $(\delta_{i,j})_{i,j=1}^{n}$ on each edge. If $i$ and $j$ are not neighbors, then set $\delta_{i,j} = \infty$. If $i = j$, then set $\delta_{i,j} = 0$.

The shortest path distance in $\mathcal{G}$ from $i$ to $j$ is

$$d(i,j) = \min_{(v_1, v_2, \ldots, v_\ell) \in \mathcal{P}(i,j)} \sum_{k=1}^{\ell-1} \delta_{v_k, v_{k+1}},$$

where $\mathcal{P}$ is the set of paths that start at $i$ and end at $j$ (i.e., $v_1 = i$ and $v_\ell = j$).

# Shortest path distances

**Proposition.** The matrix $D$ of all pairwise distances $(D_{i,j} = d(i, j))$ can be computed in $O(n^3)$ time.

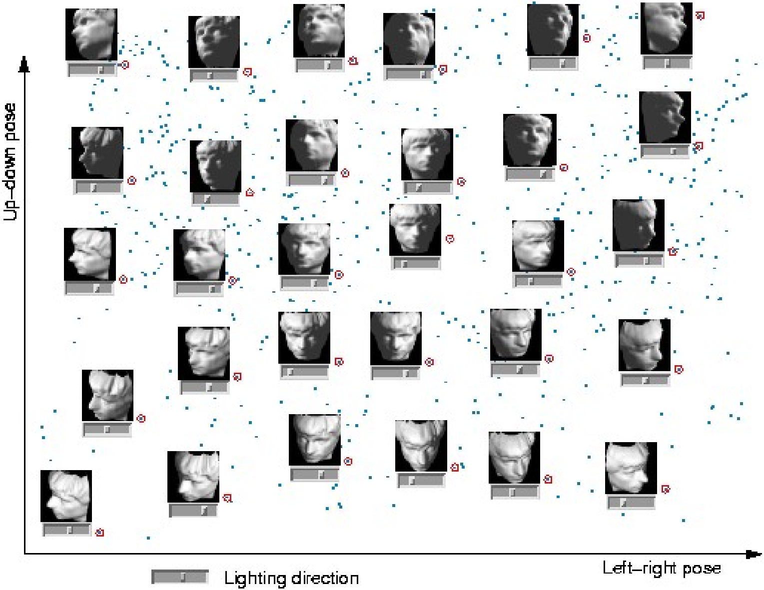
**Proposition.** Let $D^{(k)}$ be the matrix of shortest path distances along the restricted set of paths where each intermediate vertex comes from $\{1, 2, \ldots, k\}$. Then $D^{(k)}$ can be computed from $D^{(k-1)}$ in $O(n^2)$ time.

# Floyd–Warshall algorithm

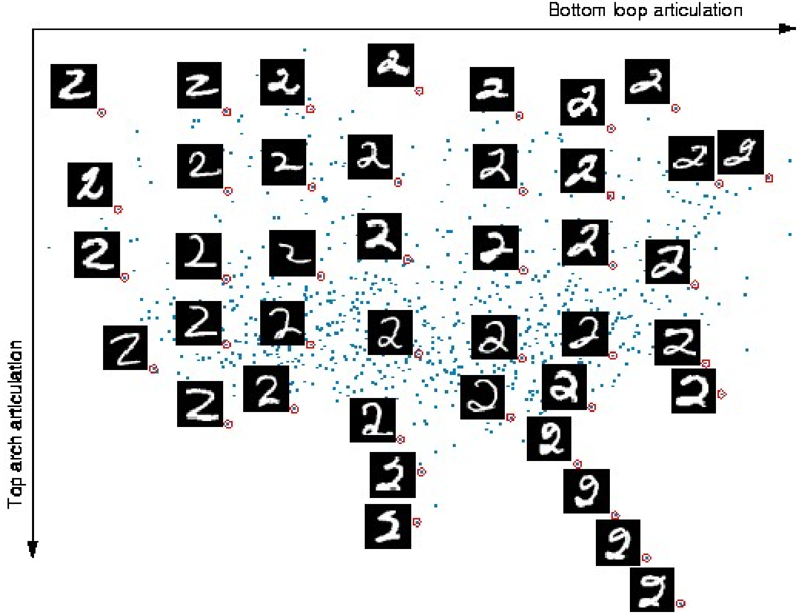
```
INPUT: matrix A with A_{i,j} = δ_{i,j} as on previous slide;
for k = 1 to n {
  for i = 1 to n {
    for j = 1 to n {
      if (A_{i,j} > A_{i,k} + A_{k,j}) then A_{i,j} ← A_{i,k} + A_{k,j} ;
    }
  }
}
OUTPUT: matrix A, in which A_{i,j} is shortest path distance
from vertex i to j
```

Overall complexity: $O(n^3)$.

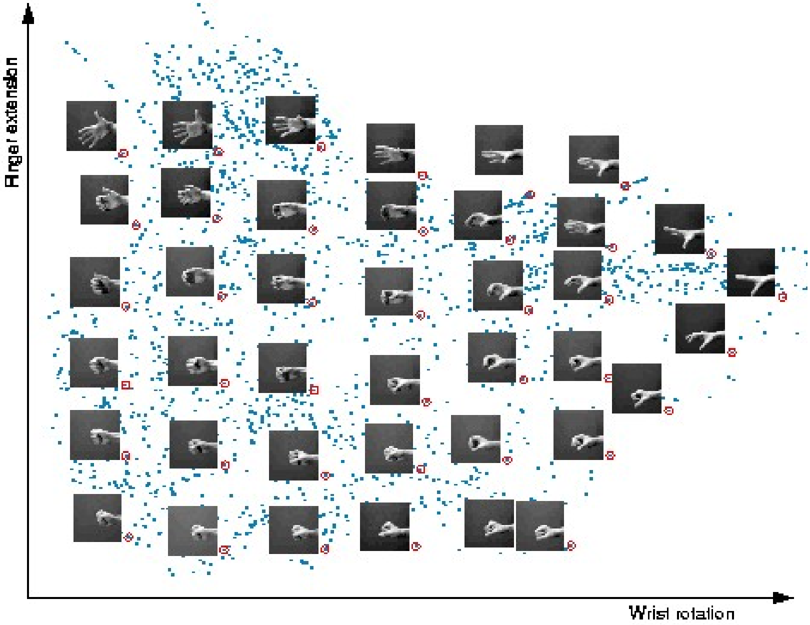
# Isomap example



# Isomap example



# Isomap example



# Properties of Isomap

- One of the first algorithms that can deal with manifolds.
- The topology must still be that of (a patch of) $\mathbb{R}^p$.
- Relatively efficient computation $O(n^3)$.
- Fragile: a single mistake in $k$–nn graph can mess up embedding.
- Not obvious how to set $k$.

# Locally Linear Embedding (LLE)

Roweis & Saul, 2000

# LLE

Again trying to find an embedding $\mathbb{R}^D \to \mathbb{R}^d$, mapping $\mathbf{x}_i \mapsto \mathbf{y}_i$. Again start with a $k$-nn graph based on distances in $\mathbb{R}^D$.

IDEA: Each point should be approximately reconstructable as a linear combination of its neighbors (locally linear property of manifolds):

$$\mathbf{x}_i \approx \sum_{j \in \mathrm{knn}(i)} w_{i,j} \mathbf{x}_j,$$

where $(w_{i,j})_{i,j}$ is a matrix of weights. Also have constraints $\sum_j w_{i,j} = 1$.

Now find an embedding that preserves these weights, i.e., $n$ vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^p$, such that

$$\mathbf{y}_i \approx \sum_j w_{i,j} \mathbf{y}_j$$

for the same matrix of weights.

# Phase 1: find the weights

Do this separately for each $i$. Formulate it as minimizing

$$\Phi = \left\| \mathbf{x}_i - \sum_{j \in \mathrm{knn}(i)} w_{i,j} \mathbf{x}_j \right\|^2 \qquad \text{s.t.} \qquad \sum_j w_{i,j} = 1.$$

**Solution.** Thanks to the constraint,

$$\Phi = \left\| \sum_{j \in \mathrm{knn}(i)} w_{i,j} (\mathbf{x}_i - \mathbf{x}_j) \right\|^2 = \mathbf{w}^\top K^{(i)} \mathbf{w},$$

where $K^{(i)}$ is the local Gram matrix, $K^{(i)}_{j,j'} = (\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_{j'})$, and $\mathbf{w} = (w_j)_{j \in \mathrm{knn}(i)}$.

# Phase 1: find the weights

The local optimization problem is

$$\underset{\mathbf{w}}{\text{minimize}}\ \mathbf{w}^\top K^{(i)}\mathbf{w} \qquad \text{s.t.} \qquad \mathbf{w}^\top \mathbf{1} = 1.$$

Introduce the Lagrangian:

$$\mathcal{L}(\lambda) = \mathbf{w}^\top K^{(i)}\mathbf{w} - \lambda(\mathbf{w}^\top \mathbf{1} - 1)$$

and solve

$$\frac{\partial}{\partial w_j}\mathcal{L}(\mathbf{w}) = \left[2K^{(i)}\mathbf{w} - \lambda\mathbf{1}\right]_j = 0 \qquad j \in \mathrm{knn}(i)$$

$$\mathbf{w} = \lambda(K^{(i)})^{-1}\mathbf{1} \qquad \text{enforcing constraints:} \quad \mathbf{w} = \frac{(K^{(i)})^{-1}\mathbf{1}}{\|\,(K^{(i)})^{-1}\mathbf{1}\,\|_1}.$$

# Phase 2: find the $\boldsymbol{y}_i$'s

Now minimize (w.r.t. $\mathbf{y}_1, \ldots, \mathbf{y}_n$)

$$\Psi = \sum_i \| \, \mathbf{y}_i - \sum_j w_{i,j} \mathbf{y}_j \, \|^2 \qquad s.t. \qquad \sum_i \mathbf{y}_i = 0 \qquad \frac{1}{n} \sum_i \mathbf{y}_i \mathbf{y}_i^\top = I.$$

**Solution.**

$$\Psi = \sum_{i,j} \mathbf{y}_i^\top M \mathbf{y}_j \ldots$$