Topic 2: DIMENSIONALITY REDUCTION

STAT 37710/CAAM 37710/CMSC 35400 Machine Learning
Risi Kondor, The University of Chicago

Dimensionality reduction

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

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

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)**.

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.

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???

Dimensionality reduction

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

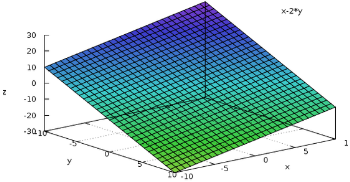
$$\mathbf{x} = (x_1, x_1, x_3, \ldots, x_d)^\top \in \mathbb{R}^d.$$

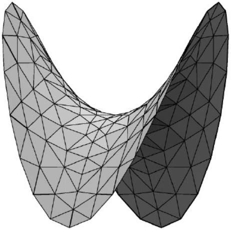
The individual dimensions are called **features** (**attributes)**.

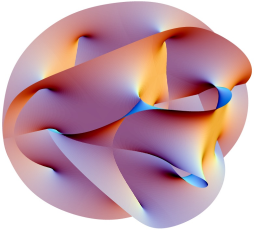
Example: Pixels of an image, a music file, etc.

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

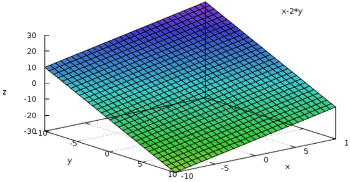


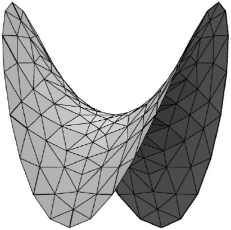


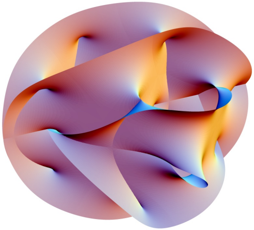


- Real world data often lie on or near lower dimensional structures (manifolds).

Dimensionality reduction

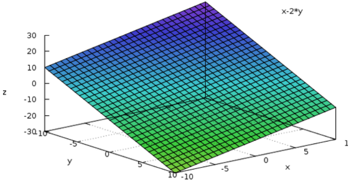


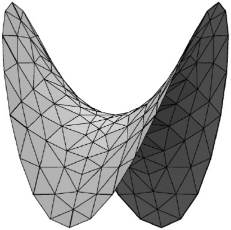


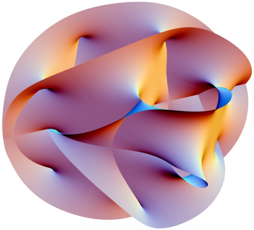


- Real world data often lie on or near lower dimensional structures (manifolds). (Really?)

Dimensionality reduction

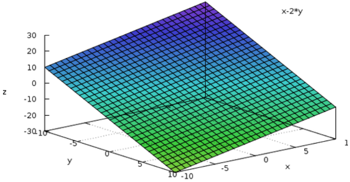


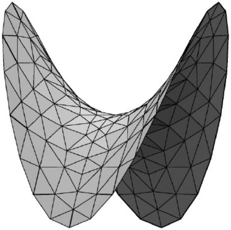


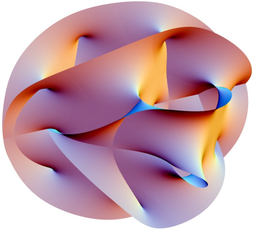


- Real world data often lie on or near lower dimensional structures (manifolds). (Really?)
  - Variables (features) may be correlated or dependent.

Dimensionality reduction

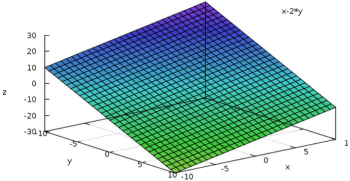


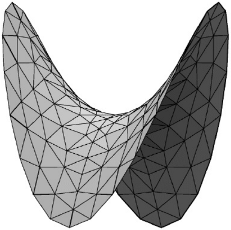


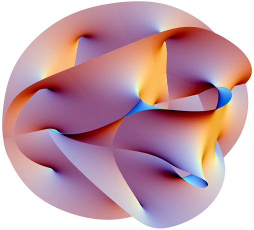


- Real world data often lie on or near lower dimensional structures (manifolds). (Really?)
    - Variables (features) may be correlated or dependent.
    - Physical systems have a small number of degrees of freedom (e.g., pose and lighting in Vision).

Dimensionality reduction

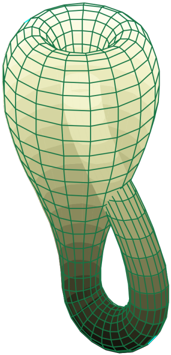


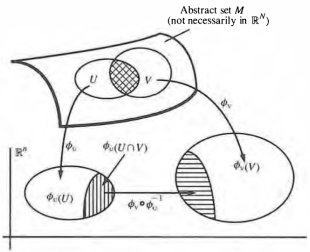




- Real world data often lie on or near lower dimensional structures (manifolds). (Really?)
  - Variables (features) may be correlated or dependent.
  - Physical systems have a small number of degrees of freedom (e.g., pose and lighting in Vision).
- IDEA: find the manifold and restrict learning algorithm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

Differentiable manifolds





In mathematics, a d dimensional **manifold** is a topological space such that each point has a neighborhood that is homeomorphic to $\mathbb{R}^d$. A differentiable manifold has additional structure, and a Riemannian manifold has a metric too $\rightarrow$ geodesics.

Dimensionality reduction

Advantages:

- **Visualization:** humans can only imagine things in 2D or 3D.

Dimensionality reduction

Advantages:

- **Visualization:** humans can only imagine things in 2D or 3D.
- **Computational efficiency:** learning algorithms work faster in low dimensions.

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.

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.

Dimensionality reduction

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

- Linear:

Dimensionality reduction

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

- Linear:
  - Principal Component Analysis (PCA)

Dimensionality reduction

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

- Linear:
  - Principal Component Analysis (PCA)
- Nonlinear ("manifold learning"):

Dimensionality reduction

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

- Linear:
    - Principal Component Analysis (PCA)
- Nonlinear ("manifold learning"):
    - Multidimensional scaling

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

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

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

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

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 $\mathbb{R}^d$.

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

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

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

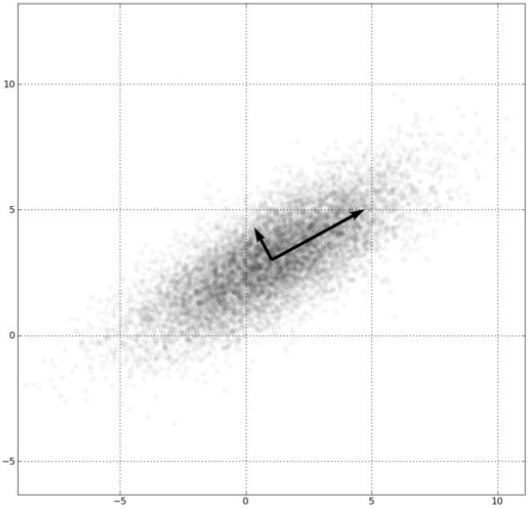
$$\underset{\mathbf{w} \in \mathbb{R}^d \setminus \{0\}}{\operatorname{argmin}} \frac{\mathbf{w}^\top A\, \mathbf{w}}{\|\mathbf{w}\|^2} = \mathbf{v}_1.$$

Similarly,

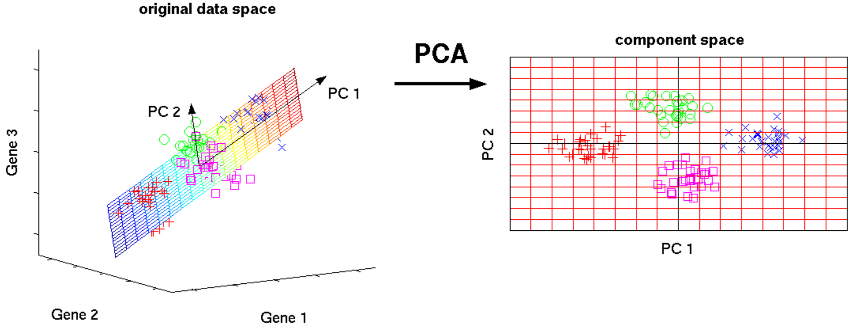
$$\underset{\mathbf{w} \in \mathbb{R}^d \setminus \{0\}}{\operatorname{argmax}} \frac{\mathbf{w}^\top A\, \mathbf{w}}{\|\mathbf{w}\|^2} = \mathbf{v}_d.$$

Principal Component Analysis

The principal directions in data

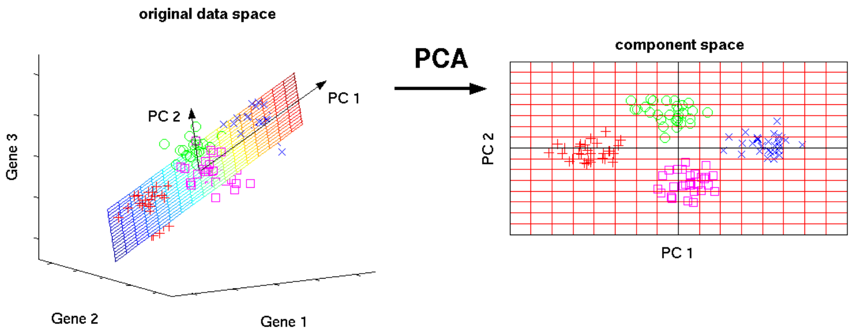


Finding the principal subspace



How can we find the most relevant subspace for the data?

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?

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 $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$.

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

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

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.

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.

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.

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

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.

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

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.

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

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.

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

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

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.

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

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

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

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

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

Further principal components

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

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

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 $\operatorname{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_{d-1}\}$.

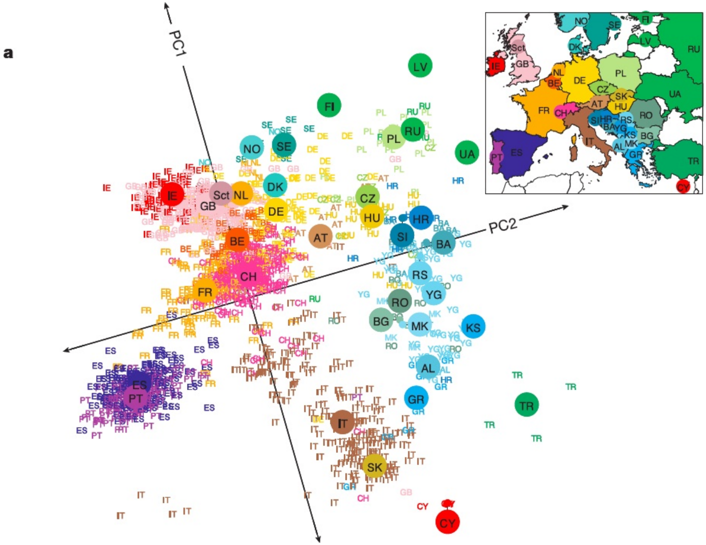
Further principal components

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

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

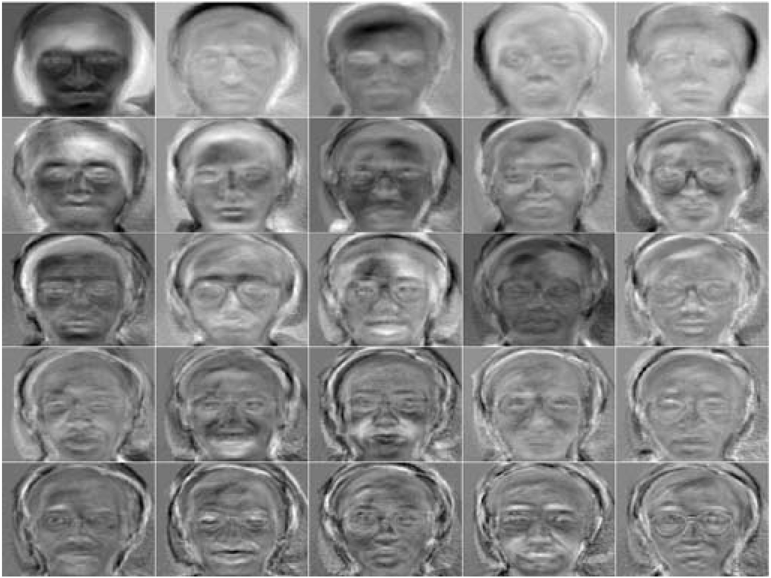
After we've found the first principal component $\boldsymbol{p}_1 = \mathbf{v}_d,$ project the data to $\mathrm{span}\,\{\mathbf{v}_1, \ldots, \mathbf{v}_{d-1}\}$. This just removes $\lambda_d \mathbf{v}_d \mathbf{v}_d^\top$ from the sum. So the second principal component is $\boldsymbol{p}_2 = \mathbf{v}_{d-1}$,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

DNA data



[Matthew Stephens, John Novembre]

Eigenfaces

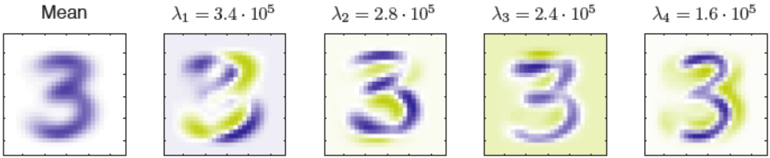


[Christopher de Coro]

Reconstruction from eigenfaces

[Christopher de Coro]

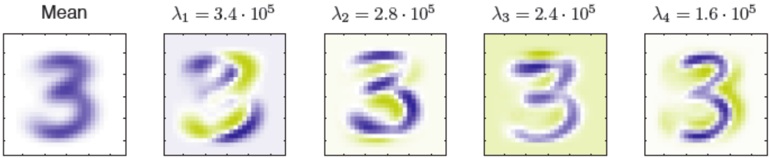
Example: digits



These are the EVectors for the four largest EValues.

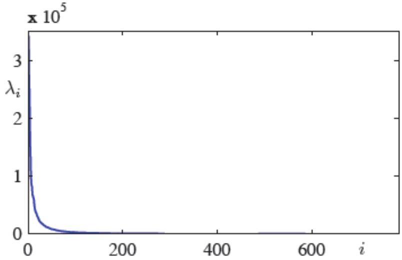
- Often the eigenvalues drop off rapidly (e.g., exponentially)

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

Summary of PCA

Advantages:

- Finds best projection
- Rotationally invariant

Summary of PCA

Advantages:

- Finds best projection
- Rotationally invariant

Disadvantages:

- Full PCA is expensive to compute

Summary of PCA

Advantages:

- Finds best projection
- Rotationally invariant

Disadvantages:

- Full PCA is expensive to compute
- Components not sparse

Summary of PCA

Advantages:

- Finds best projection
- Rotationally invariant

Disadvantages:

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

Summary of PCA

Advantages:

- Finds best projection
- Rotationally invariant

Disadvantages:

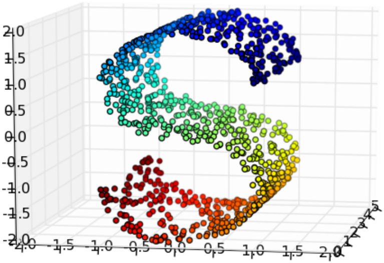
- Full PCA is expensive to compute
- Components not sparse
- Sensitive to outliers
- Linear

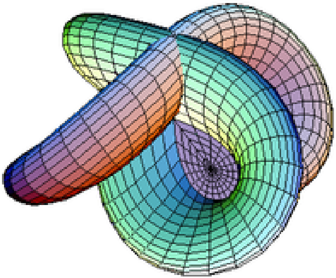
NONLINEAR DIMENSIONALITY REDUCTION

- If the data lies close to a linear subspace of $\mathbb{R}^d,$ PCA can find it.

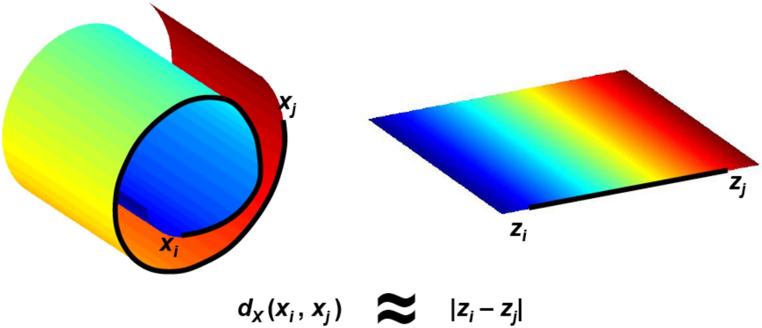
- If the data lies close to a linear subspace of $\mathbb{R}^d,$ PCA can find it.
- But what if the data lies on a nonlinear **manifold**?

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

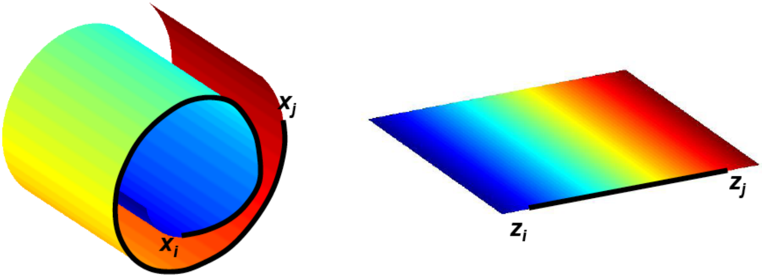


General principle



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 $\mathbb{R}^d$).

General principle

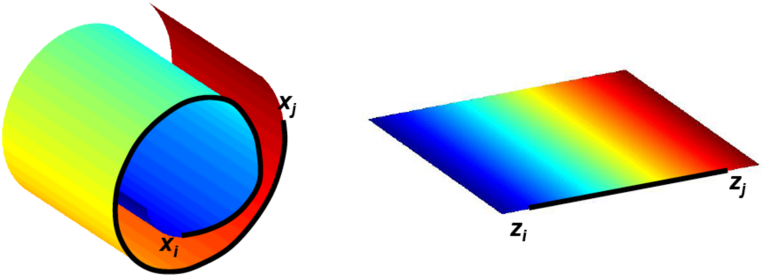


$$d_X(x_i, x_j) \approx |z_i - z_j|$$

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?

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

Methods

- Multidimensional Scaling
- Isomap

Methods

- Multidimensional Scaling
- Isomap
- Locally Linear Embedding

Methods

- Multidimensional Scaling
- Isomap
- Locally Linear Embedding
- Laplacian Eigenmaps

Methods

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

Multidimensional scaling (MDS)

Classical MDS

- **Input:** n data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$.

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*

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

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

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

The Gram matrix

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

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

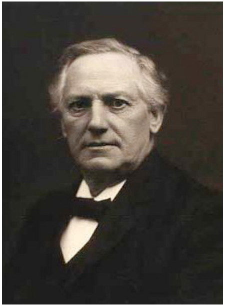
(Again, we assume that the data has been centered, i.e., $\sum_i \mathbf{x}_i = 0$.)

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 $\mathrm{rank}(G) \leq d$.

Classical MDS

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

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

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

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.

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

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 $G^* = \mathrm{argmin}_{\tilde{G} \succeq 0,\ \text{rank}(\tilde{G}) \leq p} \| \, \tilde{G} - G \, \|_{\mathrm{Frob}}^2$.
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 $\lambda_1 \geq \ldots \geq \lambda_d$.

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

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

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

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 $\to$ Data

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

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

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 $\mathbf{x}_i = [Q \Lambda^{1/2}]_{i,*}^\top$.

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.

Gram $\longrightarrow$ 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 $D = \mathrm{diag}(d_1, \ldots, d_m)$, $D^p := \mathrm{diag}(d_1^p, \ldots, d_m^p)$.

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).

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.

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

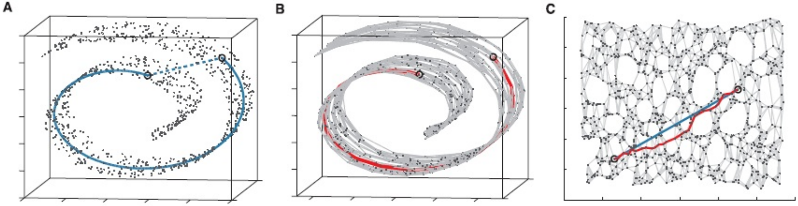
Summary of Classical MDS

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

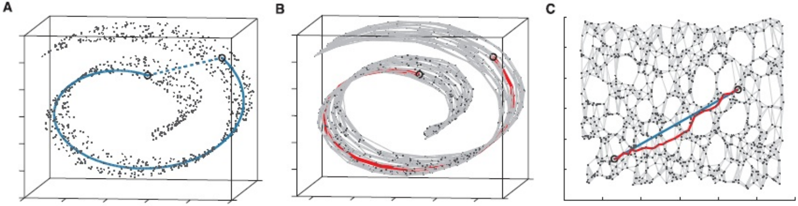
Tenenbaum, de Silva & Langford, 2000

Isomap



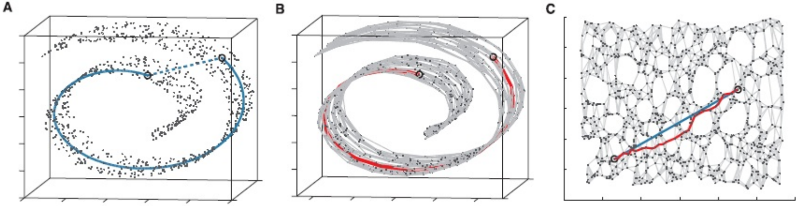
1. Convert data into a graph (e.g., a symmetrized k-nn graph).

Isomap



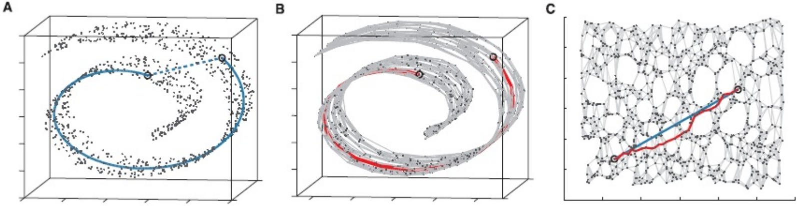
1. Convert data into a graph (e.g., a symmetrized k-nn graph).
2. Compute all pairs shortest path distances.

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 $\phi: \mathbb{R}^d \to \mathbb{R}^p$ that tries to preserve these distances.

Isomap

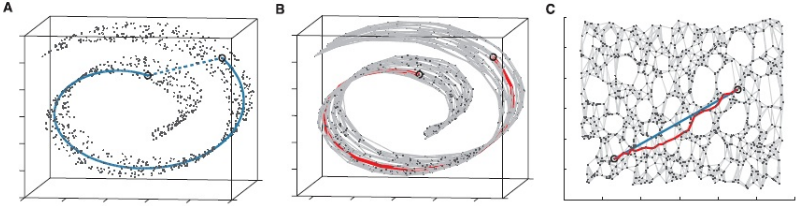


1. Convert data into a graph (e.g., a symmetrized k-nn graph).
2. Compute all pairs shortest path distances.
3. Use MDS to compute $\phi: \mathbb{R}^d \to \mathbb{R}^p$ that tries to preserve these distances.

Underlying assumptions:

1. Data lies on a manifold.

Isomap

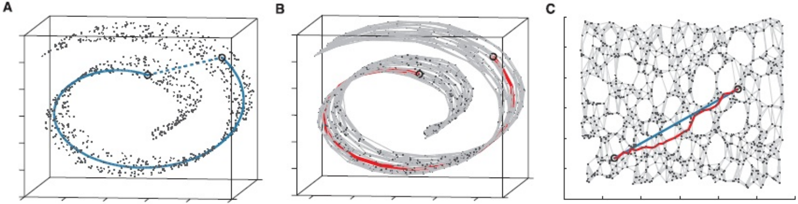


1. Convert data into a graph (e.g., a symmetrized k-nn graph).
2. Compute all pairs shortest path distances.
3. Use MDS to compute $\phi\colon \mathbb{R}^d \to \mathbb{R}^p$ that tries to preserve these distances.

Underlying assumptions:

1. Data lies on a manifold.
2. Goedesic distance on manifold is approximated by distance in the graph.

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.

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 $\delta_{i,j} = 0$.

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

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.

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

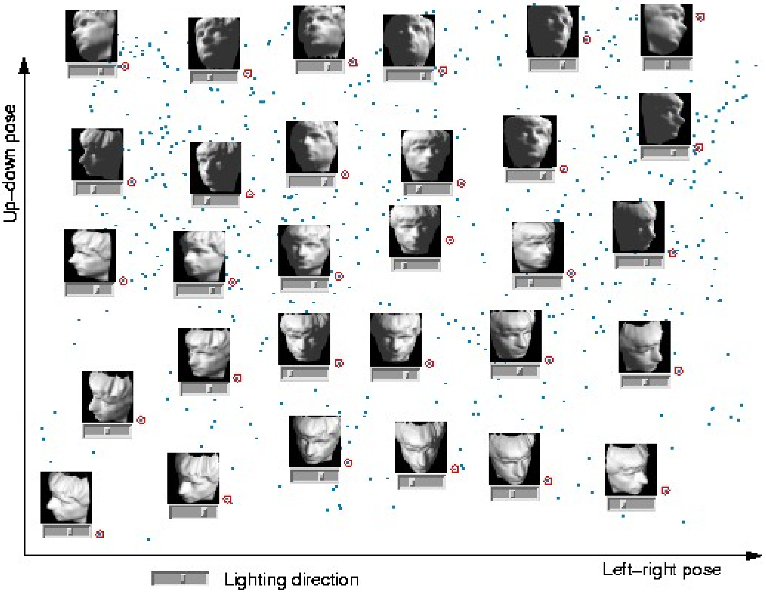
```
INPUT: matrix A with A_{i,j} = δ_{i,j} as on previous slide;
for k = 1 to n {
  for i = 1 to n {
    for j = 1 to n {
      if (A_{i,j} > A_{i,k} + A_{k,j}) then A_{i,j} ← A_{i,k} + A_{k,j} ;
    }
  }
}
OUTPUT: matrix A, in which A_{i,j} is shortest path
distance from vertex i to j
```

Floyd–Warshall algorithm

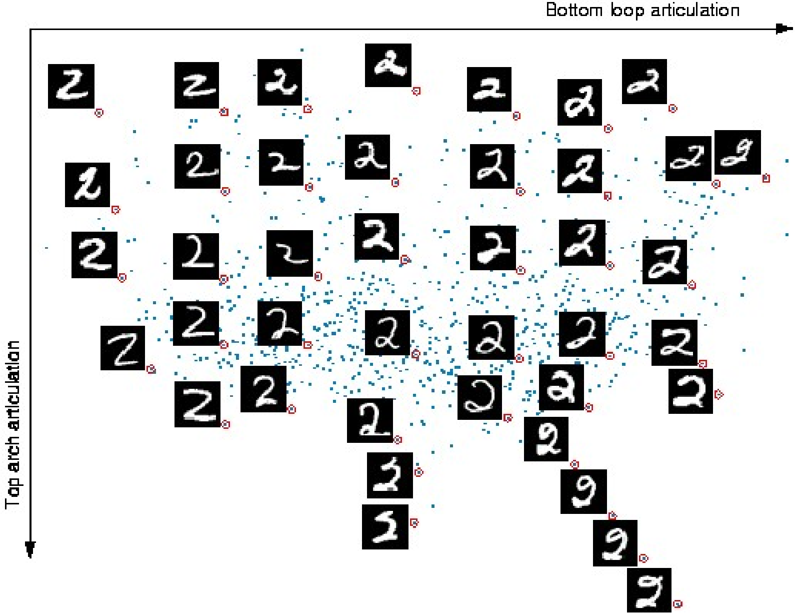
```
INPUT: matrix A with A_{i,j} = δ_{i,j} as on previous slide;
for k = 1 to n {
  for i = 1 to n {
    for j = 1 to n {
      if (A_{i,j} > A_{i,k} + A_{k,j}) then A_{i,j} ← A_{i,k} + A_{k,j} ;
    }
  }
}
OUTPUT: matrix A, in which A_{i,j} is shortest path
distance from vertex i to j
```

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

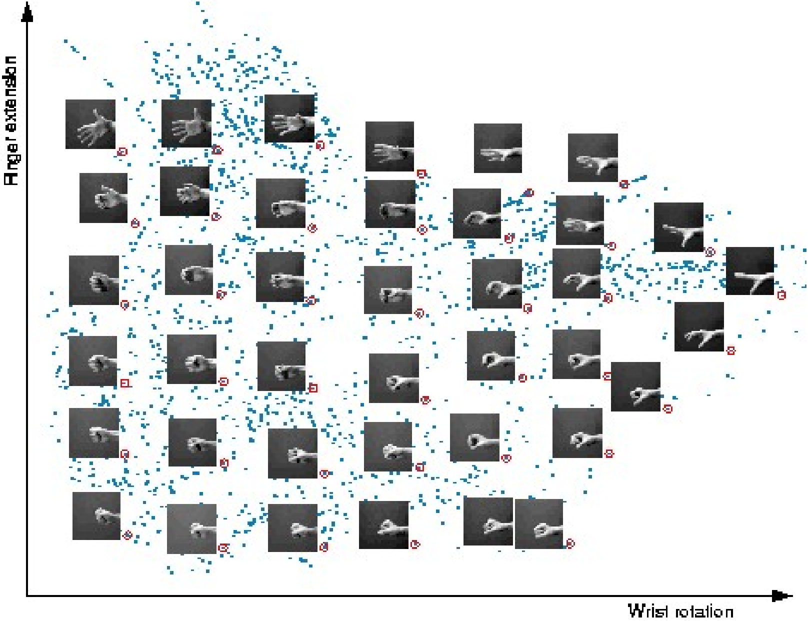
Isomap example



Isomap example



Isomap example



Properties of Isomap

- One of the first algorithms that can deal with manifolds.

Properties of Isomap

- One of the first algorithms that can deal with manifolds.
- The topology must still be that of (a patch of) $\mathbb{R}^p$.

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 $O(n^3)$.

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.

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 $\mathbf{x}_i \mapsto \mathbf{y}_i$.

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 $\mathbb{R}^D$.

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 $\sum_j w_{i,j} = 1$.

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

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

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,

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

Phase 1: find the weights

Do this separately for each i. Formulate it as minimizing

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

Solution. Thanks to the constraint,

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

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

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

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:

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

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

$$\frac{\partial}{\partial w_j} \mathcal{L}(\mathbf{w}) =$$

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

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

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

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

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

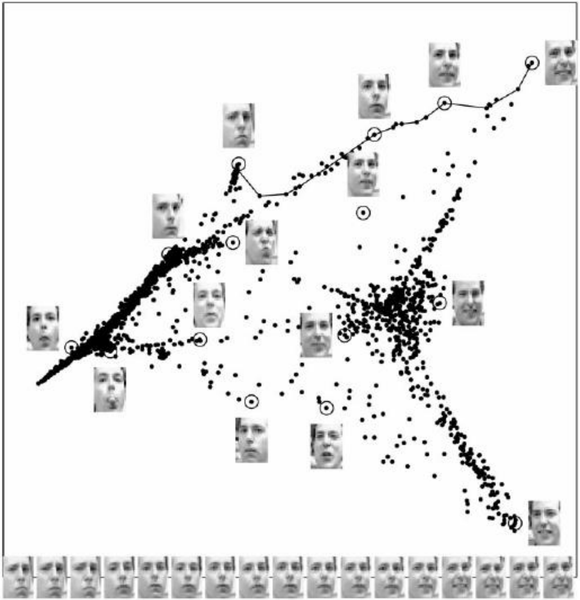
Phase 2: find the $\boldsymbol{y}_i$'s

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.

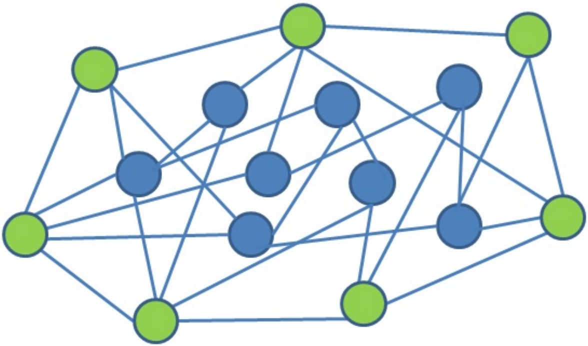
$$\Psi = \sum_{i,j} \mathbf{y}_i^\top M \mathbf{y}_j \ldots$$



Laplacian Eigenmaps

Belkin and Niyogi, 2002

Spectral Graph Theory



Spectral graph theory is about relating functions on graphs (i.e., $f : V \to \mathbb{R}$ where V is the vertex set of the graph) to the structure of the graph.

Unweighted graphs

Let $\mathcal{G}$ be an unweighted, undirected graph with vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E \subseteq V \times V$.

- The **adjacency matrix** of $\mathcal{G}$ is the matrix $A \in \{0,1\}^{n \times n}$ with

$$A = \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{otherwise,} \end{cases}$$

  where $i \sim j$ means that vertices i and j are adjacent.

Unweighted graphs

Let $\mathcal{G}$ be an unweighted, undirected graph with vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E \subseteq V \times V$.

- The **adjacency matrix** of $\mathcal{G}$ is the matrix $A \in \{0,1\}^{n \times n}$ with

$$A = \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{otherwise,} \end{cases}$$

  where $i \sim j$ means that vertices i and j are adjacent.
- The **degree matrix** of $\mathcal{G}$ is $D = \mathrm{diag}(d(1), d(2), \ldots, d(n))$, where $d(i)$ is the degree (number of neighbors) of vertex i.

Unweighted graphs

Let $\mathcal{G}$ be an unweighted, undirected graph with vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E \subseteq V \times V$.

- The **adjacency matrix** of $\mathcal{G}$ is the matrix $A \in \{0, 1\}^{n \times n}$ with

$$A = \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{otherwise,} \end{cases}$$

  where $i \sim j$ means that vertices i and j are adjacent.
- The **degree matrix** of $\mathcal{G}$ is $D = \mathrm{diag}(d(1), d(2), \ldots, d(n))$, where $d(i)$ is the degree (number of neighbors) of vertex i.
- The **Laplacian matrix** of $\mathcal{G}$ is

$$L = D - A.$$

Laplacian as a quadratic form

The Laplacian can be written as

$$L = \sum_{i\sim j} E_{i,j} \quad \text{where} \quad [E_{i,j}]_{p,q} = \begin{cases} 1 & \text{if } p=q=i \text{ or } p=q=j \\ -1 & \text{if } (p,q)=(i,j) \text{ or } (p,q)=(j,i) \\ 0 & \text{otherwise.} \end{cases}$$

Laplacian as a quadratic form

The Laplacian can be written as

$$L = \sum_{i \sim j} E_{i,j} \quad \text{where} \quad [E_{i,j}]_{p,q} = \begin{cases} 1 & \text{if } p = q = i \text{ or } p = q = j \\ -1 & \text{if } (p,q) = (i,j) \text{ or } (p,q) = (j,i) \\ 0 & \text{otherwise.} \end{cases}$$

Therefore we have the fundamental identity that for any $f \in \mathbb{R}^n$,

$$f^\top L f = \sum_{i \sim j} (f(i) - f(j))^2.$$

Laplacian as a quadratic form

The Laplacian can be written as

$$L = \sum_{i \sim j} E_{i,j} \quad \text{where} \quad [E_{i,j}]_{p,q} = \begin{cases} 1 & \text{if } p = q = i \text{ or } p = q = j \\ -1 & \text{if } (p,q) = (i,j) \text{ or } (p,q) = (j,i) \\ 0 & \text{otherwise.} \end{cases}$$

Therefore we have the fundamental identity that for any $f \in \mathbb{R}^n$,

$$f^\top L f = \sum_{i \sim j} (f(i) - f(j))^2.$$

Equivalently (and confusingly),

$$f^\top L f = \frac{1}{2} \sum_{(i,j) \in E} (f(i) - f(j))^2.$$

Laplacian as a quadratic form

The Laplacian can be written as

$$L = \sum_{i \sim j} E_{i,j} \quad \text{where} \quad [E_{i,j}]_{p,q} = \begin{cases} 1 & \text{if } p = q = i \text{ or } p = q = j \\ -1 & \text{if } (p,q) = (i,j) \text{ or } (p,q) = (j,i) \\ 0 & \text{otherwise.} \end{cases}$$

Therefore we have the fundamental identity that for any $f \in \mathbb{R}^n$,

$$f^\top L f = \sum_{i \sim j} (f(i) - f(j))^2.$$

Equivalently (and confusingly),

$$f^\top L f = \frac{1}{2} \sum_{(i,j) \in E} (f(i) - f(j))^2.$$

Exercise: Prove that L is a psd matrix.

Weighted graphs

Let $\mathcal{G}$ be a weighted, undirected graph with edge weights $(w_{i,j})_{i,j}$. Note that $w_{i,j=w_{j,i}}$, and if $i \not\sim j$, then $w_{i,j} = 0$.

- The **adjacency matrix** of $\mathcal{G}$ is the matrix $A \in (\mathbb{R}^+)^{n \times n}$ with

$$A = \begin{cases} w_{i,j} & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

Weighted graphs

Let $\mathcal{G}$ be a weighted, undirected graph with edge weights $(w_{i,j})_{i,j}$. Note that $w_{i,j=w_{j,i}}$, and if $i \not\sim j$, then $w_{i,j} = 0$.

- The **adjacency matrix** of $\mathcal{G}$ is the matrix $A \in (\mathbb{R}^+)^{n \times n}$ with

$$A = \begin{cases} w_{i,j} & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

- The **degree matrix** of $\mathcal{G}$ is $D = \operatorname{diag}(d(1), d(2), \ldots, d(n))$, where

$$d(i) = \sum_{j \neq j} w_{i,j}.$$

Weighted graphs

Let $\mathcal{G}$ be a weighted, undirected graph with edge weights $(w_{i,j})_{i,j}$. Note that $w_{i,j=w_{j,i}}$, and if $i \not\sim j$, then $w_{i,j} = 0$.

- The **adjacency matrix** of $\mathcal{G}$ is the matrix $A \in (\mathbb{R}^+)^{n \times n}$ with

$$A = \begin{cases} w_{i,j} & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

- The **degree matrix** of $\mathcal{G}$ is $D = \operatorname{diag}(d(1), d(2), \ldots, d(n))$, where

$$d(i) = \sum_{j \neq j} w_{i,j}.$$

- The **Laplacian matrix** of $\mathcal{G}$ is

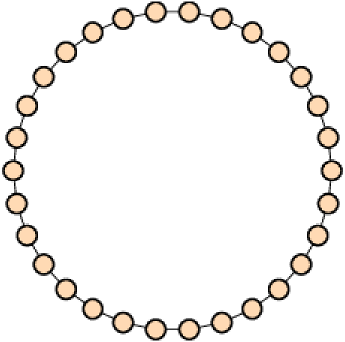
$$L = D - A.$$

The normalized Laplacian

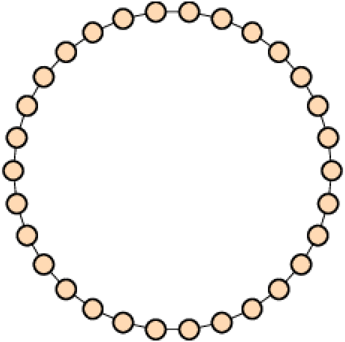
When the degree distribution is uneven, it is often much better to work with the **normalized Laplacian**

$$\tilde{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}.$$

Example: cycle graph

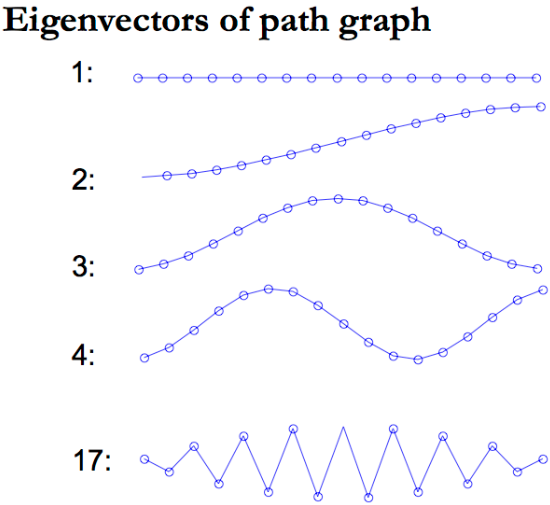


Example: cycle graph



$$f_k(v_i) = \sin(2\pi ki/n) \qquad k = 1, 2, \ldots \lfloor n/2 \rfloor$$
$$g_k(v_i) = \cos(2\pi ki/n) \qquad k = 0, 1, 2, \ldots, \lfloor (n-1)/2 \rfloor,$$

Example: path graph



Connectivity

Theorem

The multiplicity of 0 in the spectrum of L (i.e., the number of zero eigenvalues) is the number of connected components of $\mathcal{G}$.

Connectivity

Theorem

The multiplicity of 0 in the spectrum of L (i.e., the number of zero eigenvalues) is the number of connected components of $\mathcal{G}$.

Fiedler vector

Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of L, and $v_1, v_2, \ldots, v_n$ be the corresponding normalized eigenvectors.

Fiedler vector

Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of L, and $v_1, v_2, \ldots, v_n$ be the corresponding normalized eigenvectors.

- By the above, $\lambda_1 = 0$ and $v_1 = \frac{1}{\sqrt{n}}\mathbf{1}$ for any graph.

Fiedler vector

Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of L, and $v_1, v_2, \ldots, v_n$ be the corresponding normalized eigenvectors.

- By the above, $\lambda_1 = 0$ and $v_1 = \frac{1}{\sqrt{n}}\mathbf{1}$ for any graph.
- The second eigenvector v_2, is called the **Fiedler vector**, and is particularly informative about how to cluster the graph.

Cheeger's inequality

Let $S \subset V$, $= V \setminus S$, and $E(S, \overline{S}) = \sum_{i \in S} \sum_{j \in \overline{s}} w_{i,j}$.

Cheeger's inequality

Let $S \subset V$, $= V \setminus S$, and $E(S,\overline{S}) = \sum_{i \in S} \sum_{j \in \overline{s}} w_{i,j}$. Further for any $W \subseteq V$, let $d(W) = \sum_{i \in W} d(i)$.

- The **conductance** of S is defined as

$$\phi(S) = d(V)\,\frac{E(S,\overline{S})}{d(S)\,d(\overline{S})}.$$

Cheeger's inequality

Let $S \subset V$, $= V \setminus S$, and $E(S, \overline{S}) = \sum_{i \in S} \sum_{j \in \overline{s}} w_{i,j}$. Further for any $W \subseteq V$, let $d(W) = \sum_{i \in W} d(i)$.

- The **conductance** of S is defined as

$$\phi(S) = d(V) \, \frac{E(S, \overline{S})}{d(S) \, d(\overline{S})}.$$

- The conductance of the whole graph is $\phi_{\mathcal{G}} = \min_{S \subset V} \phi(S)$.

Cheeger's inequality

Let $S \subset V$, $= V \setminus S$, and $E(S, \overline{S}) = \sum_{i \in S} \sum_{j \in \overline{s}} w_{i,j}$. Further for any $W \subseteq V$, let $d(W) = \sum_{i \in W} d(i)$.

- The **conductance** of S is defined as

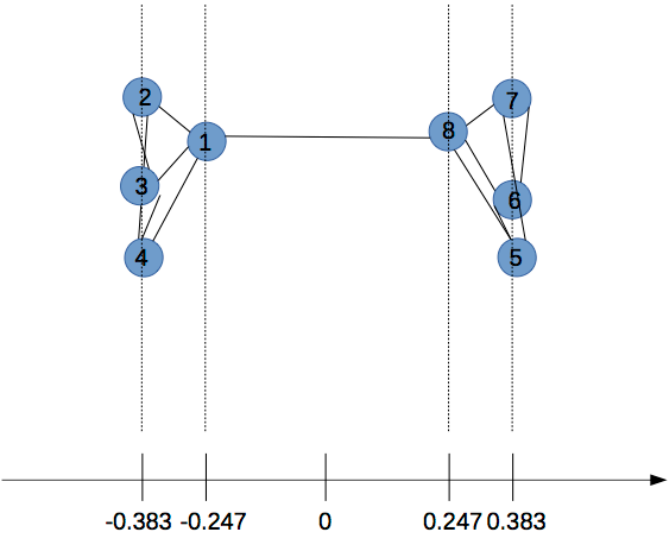
$$\phi(S) = d(V)\,\frac{E(S, \overline{S})}{d(S)\,d(\overline{S})}.$$

- The conductance of the whole graph is $\phi_{\mathcal{G}} = \min_{S \subset V} \phi(S)$.
- **Cheeger's inequality** states that

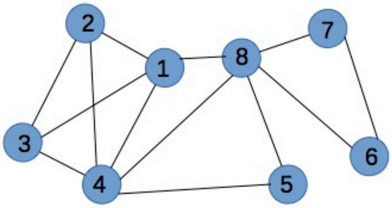
$$\frac{\phi_{\mathcal{G}}^2}{2d_{\text{max}}} \leq \lambda_2 \leq \phi_{\mathcal{G}},$$

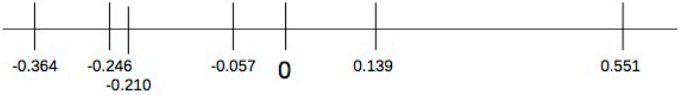
where d_{max} is the maximum degree of any vertex in $\mathcal{G}$.

Example

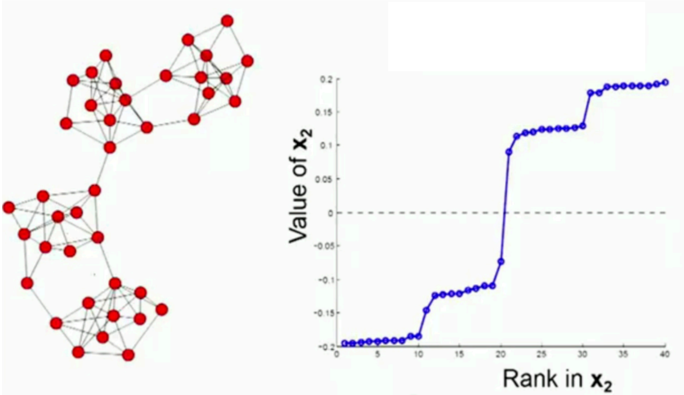


Example





Example



The first few eigenvectors can be used for clustering $\rightarrow$ spectral graph partitioning

The Laplace–Beltrami operator

The graph Laplacian is the discrete analog of the Laplace-Beltrami operator.

- The Laplacian operator on $\mathbb{R}^d$ is

$$\Delta = \nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \ldots + \frac{\partial^2}{\partial x_d^2}.$$

The Laplace–Beltrami operator

The graph Laplacian is the discrete analog of the Laplace-Beltrami operator.

- The Laplacian operator on $\mathbb{R}^d$ is

$$\Delta = \nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \ldots + \frac{\partial^2}{\partial x_d^2}.$$

- More generally, the Laplace–Beltrami operator on a d dimensional Riemannian manifold with metric tensor g is

$$\Delta = \frac{1}{\sqrt{\det g}} \sum_{i,j=1}^{d} \partial_i \sqrt{\det g} g^{i,j} \, \partial_j.$$

The graph Laplacian can be regarded as a discretization of these operators.

Discretization of Laplacian

- In $\mathbb{R}$, the (finite difference) discretization of $\nabla = \frac{\partial}{\partial x}$ is derived from

$$(\nabla f)(x) = \left(\frac{\partial}{\partial x} f\right)(x) = \frac{f(x+h/2) - f(x-h/2)}{h}.$$

Discretization of Laplacian

- In $\mathbb{R}$, the (finite difference) discretization of $\nabla = \frac{\partial}{\partial x}$ is derived from

$$(\nabla f)(x) = \left(\frac{\partial}{\partial x} f\right)(x) = \frac{f(x+h/2) - f(x-h/2)}{h}.$$

- The discretization of Δ is derived from

$$(\Delta f)(x) = (\nabla(\nabla f))(x) =$$

Discretization of Laplacian

- In $\mathbb{R}$, the (finite difference) discretization of $\nabla = \frac{\partial}{\partial x}$ is derived from

$$(\nabla f)(x) = \left(\frac{\partial}{\partial x} f\right)(x) = \frac{f(x+h/2) - f(x-h/2)}{h}.$$

- The discretization of Δ is derived from

$$(\Delta f)(x) = (\nabla(\nabla f))(x) = \frac{(\nabla f)(x+h/2) - (\nabla f)(x-h/2)}{h} =$$

Discretization of Laplacian

- In $\mathbb{R}$, the (finite difference) discretization of $\nabla = \frac{\partial}{\partial x}$ is derived from

$$(\nabla f)(x) = \left(\frac{\partial}{\partial x} f\right)(x) = \frac{f(x+h/2) - f(x-h/2)}{h}.$$

- The discretization of Δ is derived from

$$(\Delta f)(x) = (\nabla(\nabla f))(x) = \frac{(\nabla f)(x+h/2) - (\nabla f)(x-h/2)}{h} =$$

$$= \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}.$$

Discretization of Laplacian

- In $\mathbb{R}$, the (finite difference) discretization of $\nabla = \frac{\partial}{\partial x}$ is derived from

$$(\nabla f)(x) = \left(\frac{\partial}{\partial x} f\right)(x) = \frac{f(x+h/2) - f(x-h/2)}{h}.$$

- The discretization of Δ is derived from

$$(\Delta f)(x) = (\nabla(\nabla f))(x) = \frac{(\nabla f)(x+h/2) - (\nabla f)(x-h/2)}{h} =$$

$$= \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}.$$

If we regard f as a vector, $\boldsymbol{f} = (\ldots, f(x-h), f(x), f(x+h), \ldots)^\top$, then the latter is just $-L\boldsymbol{f}/h^2$, where L is the Laplacian of the line graph.

Discretization of Laplacian

- In $\mathbb{R}$, the (finite difference) discretization of $\nabla = \frac{\partial}{\partial x}$ is derived from

$$(\nabla f)(x) = \left(\frac{\partial}{\partial x} f\right)(x) = \frac{f(x+h/2) - f(x-h/2)}{h}.$$

- The discretization of Δ is derived from

$$(\Delta f)(x) = (\nabla(\nabla f))(x) = \frac{(\nabla f)(x+h/2) - (\nabla f)(x-h/2)}{h} =$$

$$= \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}.$$

If we regard f as a vector, $\boldsymbol{f} = (\ldots, f(x-h), f(x), f(x+h), \ldots)^\top$, then the latter is just $-L\boldsymbol{f}/h^2$, where L is the Laplacian of the line graph. Similarly for grids on $\mathbb{R}^d$.

Discretization of Laplacian

- In $\mathbb{R}$, the (finite difference) discretization of $\nabla = \frac{\partial}{\partial x}$ is derived from

$$(\nabla f)(x) = \left(\frac{\partial}{\partial x} f\right)(x) = \frac{f(x+h/2) - f(x-h/2)}{h}.$$

- The discretization of Δ is derived from

$$(\Delta f)(x) = (\nabla(\nabla f))(x) = \frac{(\nabla f)(x+h/2) - (\nabla f)(x-h/2)}{h} =$$

$$= \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}.$$

If we regard f as a vector, $\boldsymbol{f} = (\ldots, f(x-h), f(x), f(x+h), \ldots)^\top$, then the latter is just $-L\boldsymbol{f}/h^2$, where L is the Laplacian of the line graph. Similarly for grids on $\mathbb{R}^d$. $\langle f, \Delta f \rangle$ is a natural measure of roughness of f $\rightarrow$ sheds new light on L as a quadratic form.

The heat equation

The flow of heat in a homogenous medium is governed by the equation

$$\frac{\partial}{\partial t} f(\mathbf{x}, t) = \kappa \Delta f(\mathbf{x}, t).$$

The heat equation

The flow of heat in a homogenous medium is governed by the equation

$$\frac{\partial}{\partial t} f(\mathbf{x}, t) = \kappa \Delta f(\mathbf{x}, t).$$

Δ is a negative definite self-adjoint operator.

The heat equation

The flow of heat in a homogenous medium is governed by the equation

$$\frac{\partial}{\partial t} f(\mathbf{x}, t) = \kappa \Delta f(\mathbf{x}, t).$$

Δ is a negative definite self-adjoint operator. Solution to this is

$$f(\mathbf{x}, t) = e^{\kappa t \Delta} f(\mathbf{x}, 0) \qquad \text{where} \qquad e^T := I + T + \frac{1}{2}T^2 + \frac{1}{6}T^3 + \ldots.$$

The heat equation

The flow of heat in a homogenous medium is governed by the equation

$$\frac{\partial}{\partial t} f(\mathbf{x}, t) = \kappa \Delta f(\mathbf{x}, t).$$

Δ is a negative definite self-adjoint operator. Solution to this is

$$f(\mathbf{x}, t) = e^{\kappa t \Delta} f(\mathbf{x}, 0) \qquad \text{where} \qquad e^T := I + T + \frac{1}{2} T^2 + \frac{1}{6} T^3 + \ldots .$$

In particular, if our domain $\mathcal{M}$ is compact, then the eigenfunctions of Δ, i.e., $\Delta g_i = \lambda_i g_i$ form a basis for $\mathcal{M}$ and

$$f(\mathbf{x}, 0) = \sum_i \alpha_i g_i \qquad\qquad f(\mathbf{x}, 0) = \sum_i e^{\lambda_i \kappa t} \alpha_i g_i.$$

The heat equation

The flow of heat in a homogenous medium is governed by the equation

$$\frac{\partial}{\partial t} f(\mathbf{x}, t) = \kappa \Delta f(\mathbf{x}, t).$$

Δ is a negative definite self-adjoint operator. Solution to this is

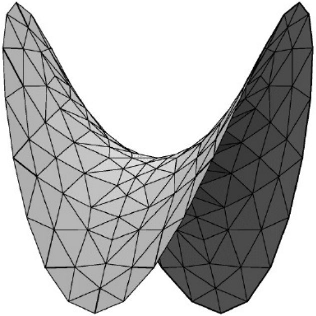
$$f(\mathbf{x}, t) = e^{\kappa t \Delta} f(\mathbf{x}, 0) \qquad \text{where} \qquad e^T := I + T + \frac{1}{2} T^2 + \frac{1}{6} T^3 + \ldots.$$

In particular, if our domain $\mathcal{M}$ is compact, then the eigenfunctions of Δ,
i.e., $\Delta g_i = \lambda_i g_i$ form a basis for $\mathcal{M}$ and

$$f(\mathbf{x}, 0) = \sum_i \alpha_i g_i \qquad\qquad f(\mathbf{x}, 0) = \sum_i e^{\lambda_i \kappa t} \alpha_i g_i.$$

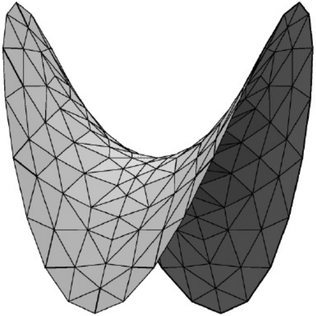
The long time behavior of the system is determined by the low $|\lambda_i|$ modes!!!

Laplacian Eigenmaps [Belkin&Niyogi]



- Turn dimensionality reduction into a graph problem by forming knn-mesh, possibly weighted by $w_{i,j} = \exp(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / (2\sigma^2))$

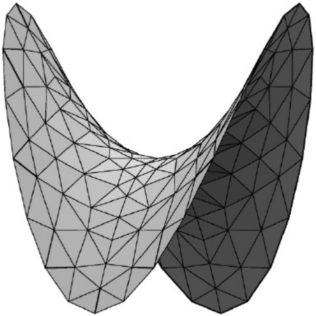
Laplacian Eigenmaps [Belkin&Niyogi]



- Turn dimensionality reduction into a graph problem by forming knn-mesh, possibly weighted by $w_{i,j} = \exp(- \|\mathbf{x}_i - \mathbf{x}_j\|^2 / (2\sigma^2))$
- Embed according to first p non-zero e-value e-vectors:

$$\phi \colon V \to \mathbb{R}^p \qquad i \mapsto \begin{pmatrix} v_1(i) \\ \vdots \\ v_{p+1}(i) \end{pmatrix}$$

Laplacian Eigenmaps [Belkin&Niyogi]



- Turn dimensionality reduction into a graph problem by forming knn-mesh, possibly weighted by $w_{i,j} = \exp(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / (2\sigma^2))$
- Embed according to first p non-zero e-value e-vectors:

$$\phi\colon V \to \mathbb{R}^p \qquad i \mapsto \begin{pmatrix} v_1(i) \\ \vdots \\ v_{p+1}(i) \end{pmatrix}$$

- Intuition: these are the smoothest functions on the graph, and they give global coordinates

Laplacian Eigenmaps: detail

Formulate the problem as minimizing the strain

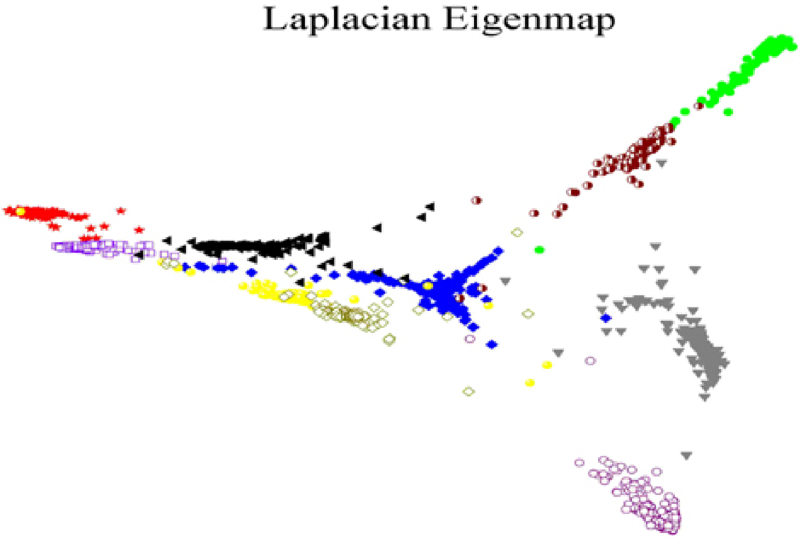
$$\mathcal{E} = \sum_{i,j} w_{i,j} \left\| \mathbf{y}_i - \mathbf{y}_j \right\|^2 = 2\mathrm{tr}(Y^\top L Y).$$

Laplacian Eigenmaps: detail

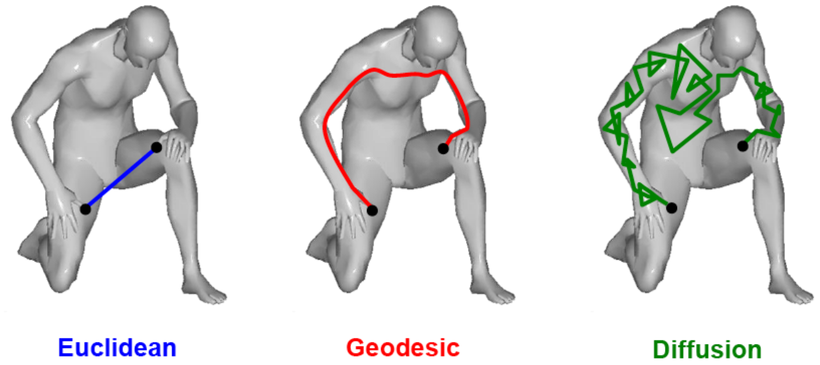
Formulate the problem as minimizing the strain

$$\mathcal{E} = \sum_{i,j} w_{i,j} \left\| \mathbf{y}_i - \mathbf{y}_j \right\|^2 = 2\mathrm{tr}(Y^\top L Y).$$

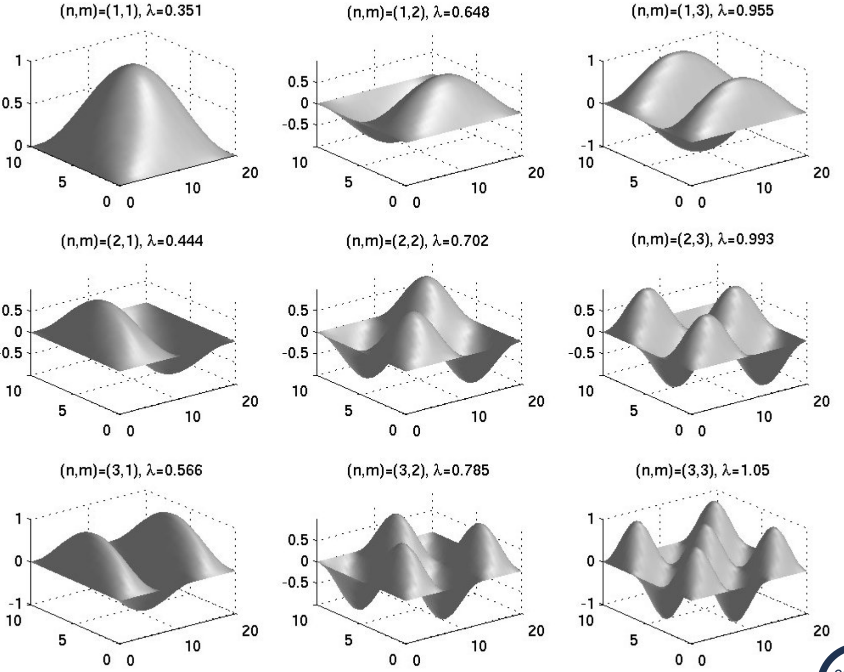
Adding the additional constraint $Y^\top D Y = I$, after some algebra, this leads to the generalized eigenvalue problem $LY = DY$.

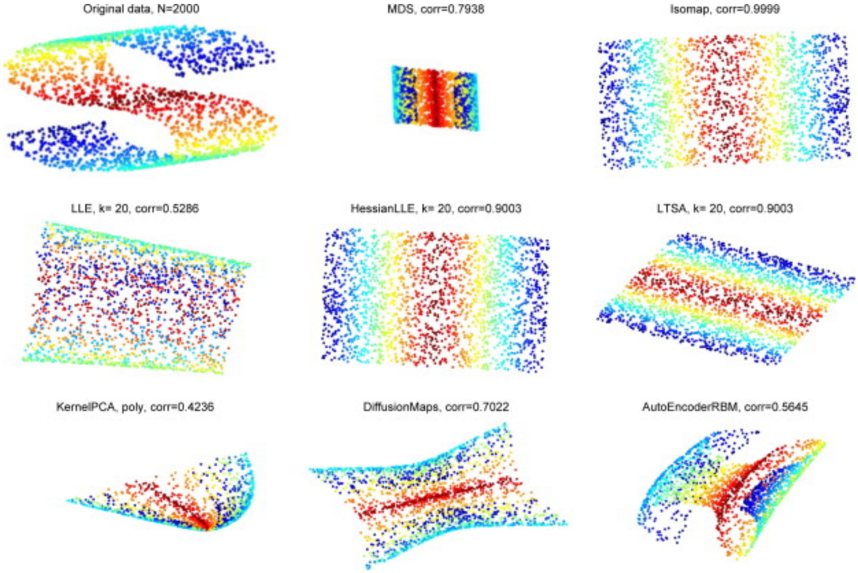

Laplacian Eigenmap

Three different metrics



Laplacian eigenmaps corresponds to PCA w.r.t. the diffusion metric on the manifold, because the diffusion (heat) kernel is exactly $e^{-\beta L}$ [K and Lafferty, 2001].


(n,m)=(1,1), λ=0.351
(n,m)=(1,2), λ=0.648
(n,m)=(1,3), λ=0.955
(n,m)=(2,1), λ=0.444
(n,m)=(2,2), λ=0.702
(n,m)=(2,3), λ=0.993
(n,m)=(3,1), λ=0.566
(n,m)=(3,2), λ=0.785
(n,m)=(3,3), λ=1.05
1
0.5
0
-0.5
-1
10
5
0
20


Original data, N=2000
MDS, corr=0.7938
Isomap, corr=0.9999
LLE, k= 20, corr=0.5286
HessianLLE, k= 20, corr=0.9003
LTSA, k= 20, corr=0.9003
KernelPCA, poly, corr=0.4236
DiffusionMaps, corr=0.7022
AutoEncoderRBM, corr=0.5645