StatML

Visualization and dimensionality reduction 2

13.3.2014

Aasa Feragen
aasa@diku.dk

After today's lecture you should

- Be familiar with Multidimensional Scaling (MDS)
  - Classical definition
  - Low-distortion interpretation of PCA
  - MDS for non-Euclidean data
- Be familiar with basic manifold learning techniques
  - Isomap
  - What problems does Isomap (try to) solve?
  - What are its strengths and weaknesses
- Be familiar with kernel PCA:
  - Its definition and relation to standard PCA
  - How to compute it
  - Applications to nonlinear PCA and non-Euclidean data

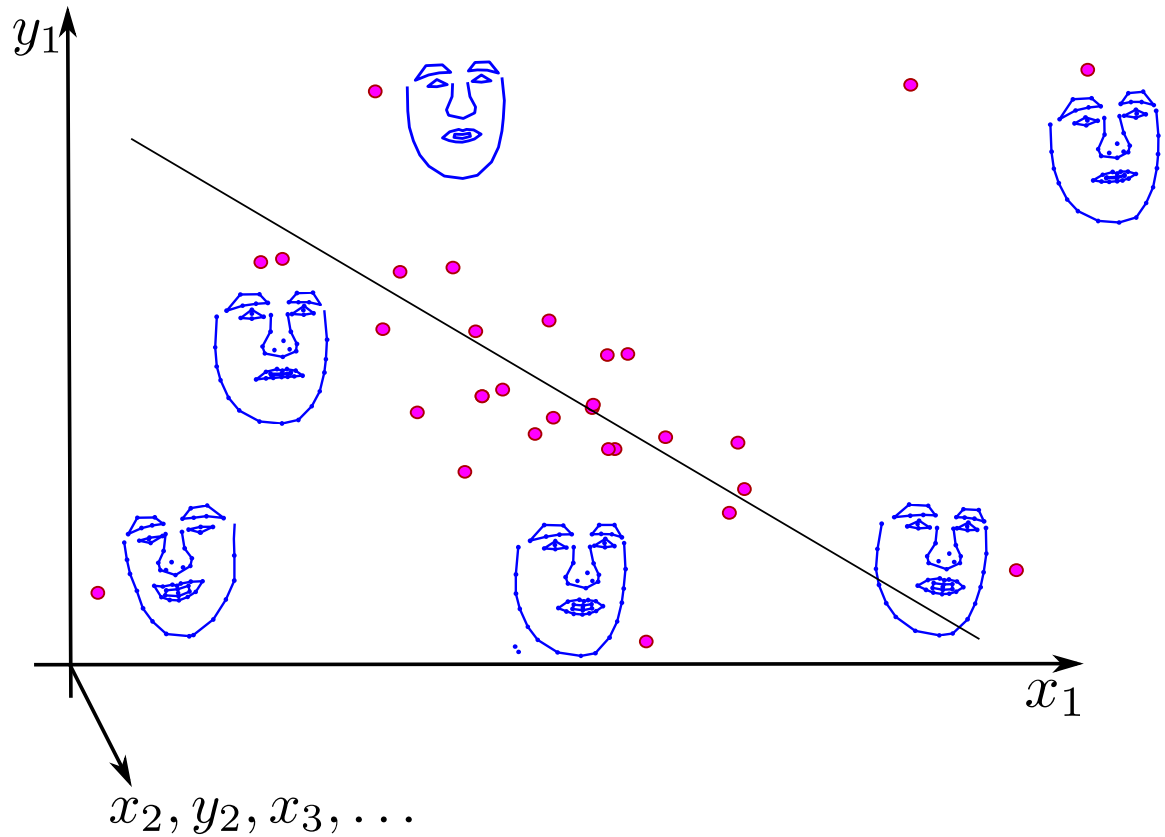
Optional reading material

- Dimensionality Reduction: A comparative review (van der Maaten, Postma, van den Herik, 2009)
- A global geometric framework for nonlinear dimensionality reduction, Tenenbaun, da Silva and Langford, Science, 2000

  (Isomap; the following discussing papers include Locally Linear Embedding)

Last time: Principal Component Analysis (PCA)

- Linear model for latent variable

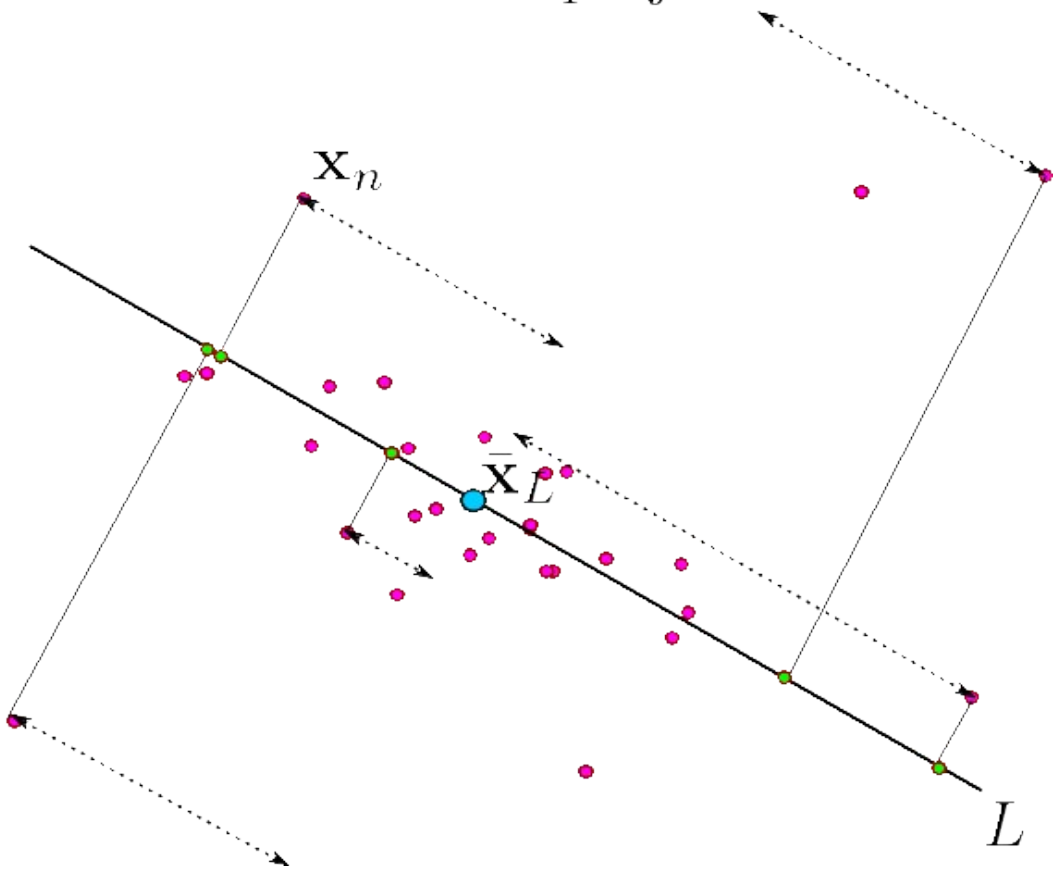


PCA definition 1: Variance maximization

Find M-dimensional hyperplane L which maximizes projected variance

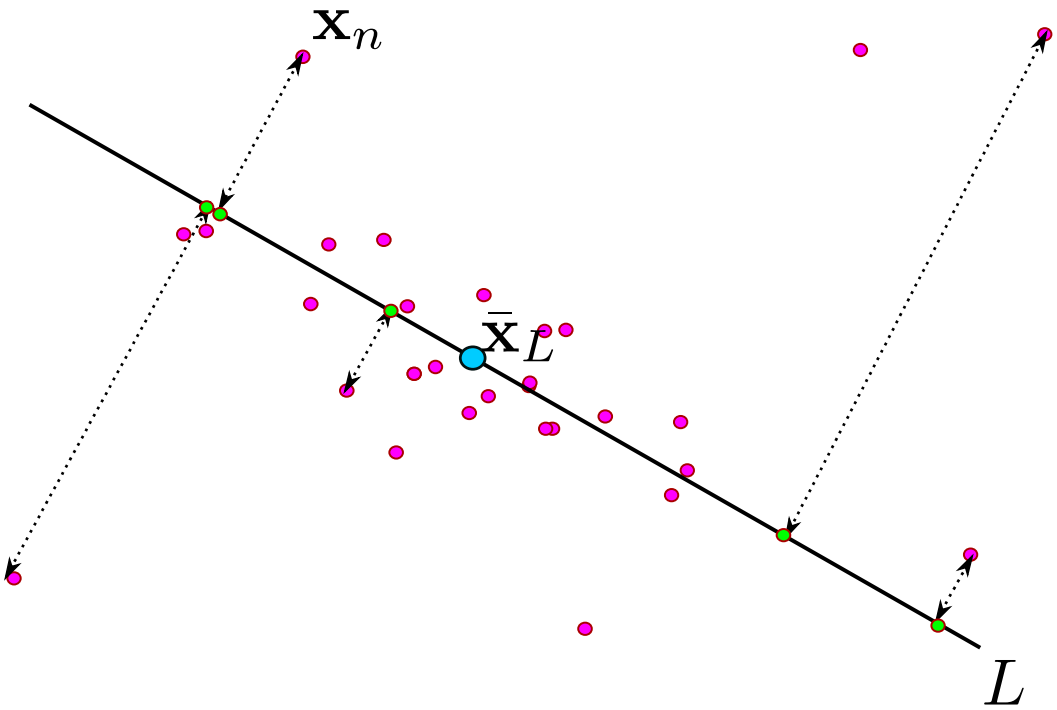
$\sum_{n=1}^{N} \|\mathrm{pr}_L \mathbf{x}_n - \bar{\mathbf{x}}_L\|^2$

$\bar{\mathbf{x}}_L$ mean of $\{\mathrm{pr}_L(\mathbf{x}_n)\}$



PCA definition 2: Error minimization

Find M-dimensional hyperplane L which minimizes quadratic projection error $\sum_{n=1}^{N} \|\mathbf{x}_n - \mathrm{pr}_L \mathbf{x}_n\|^2$

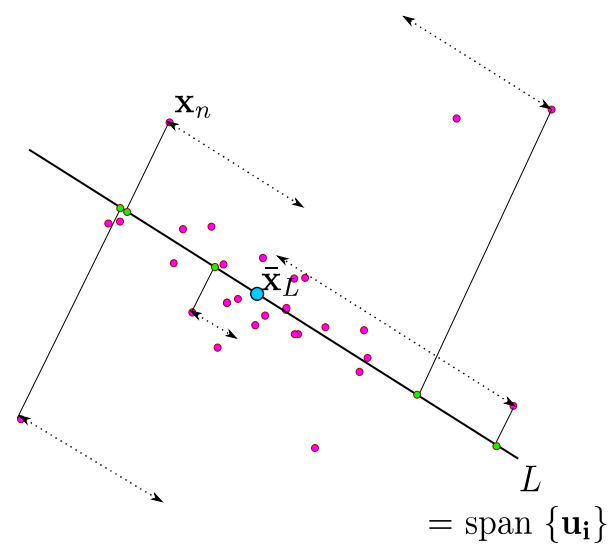


Computing principal components

To compute the M-dimensional hyperplane minimizing projected variance:

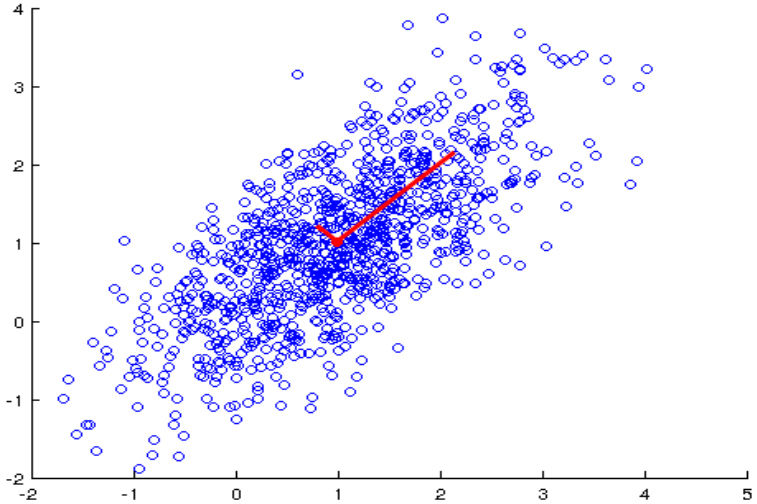
1. Compute covariance matrix $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x_n} - \bar{\mathbf{x}})(\mathbf{x_n} - \bar{\mathbf{x}})^{\mathbf{T}}$
2. Compute its eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M \geq \ldots \geq \lambda_D$ and their eigenvectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_M}, \ldots \mathbf{u_D}$
3. Optimal hyperplane is span $\{\mathbf{u_i}\}_{\mathbf{i=1}}^{\mathbf{M}}$
4. Projected variance is $\lambda_1 + \lambda_2 + \ldots + \lambda_M$

Look familiar?



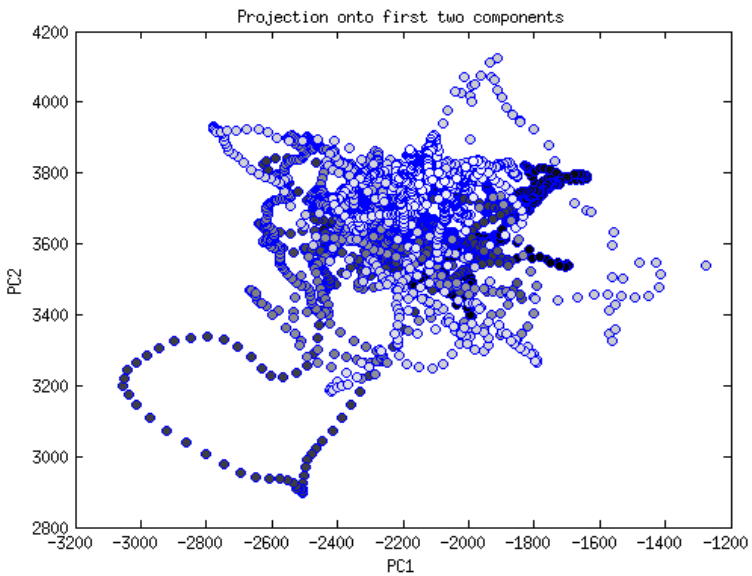
Computing principal components

- Eigenvalue decomposition shows that PCA is equivalent to
  - Fitting a Gaussian distribution to your data using mean and covariance
  - Using the eigenvectors of the covariance as principal directions

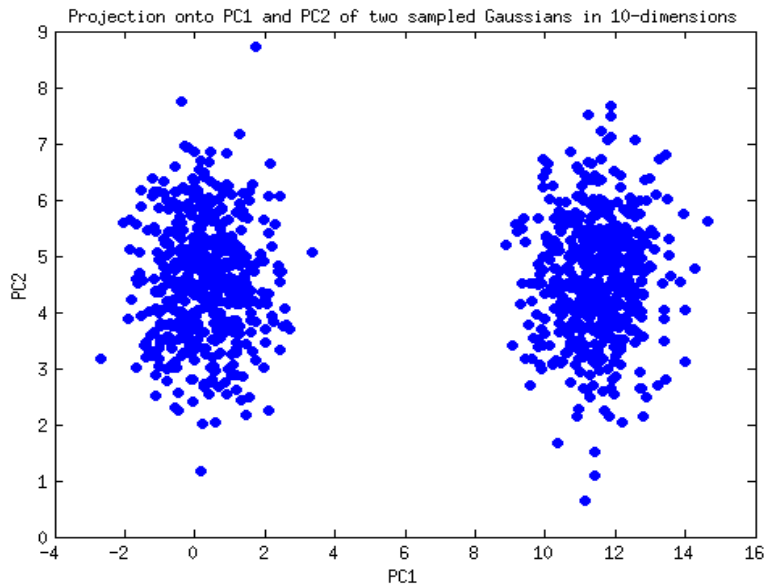


Multidimensional Scaling

- Last time we used projection onto principal components to visualize dataset structure



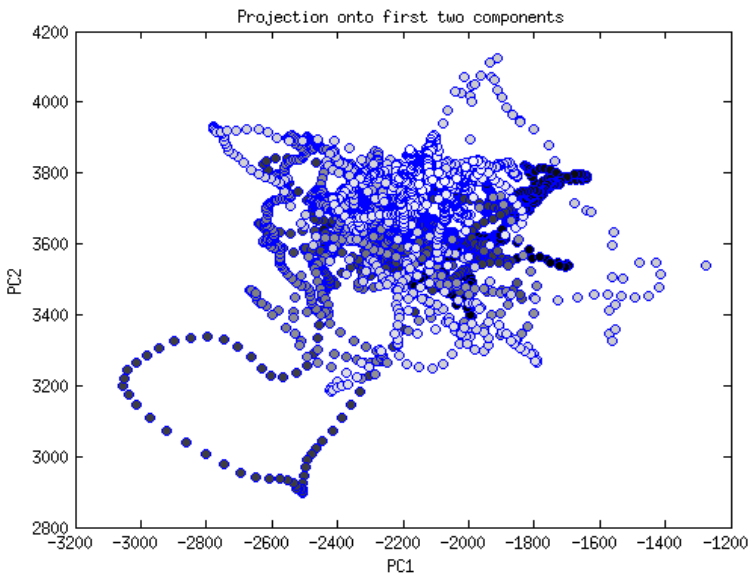
Face movement during movie



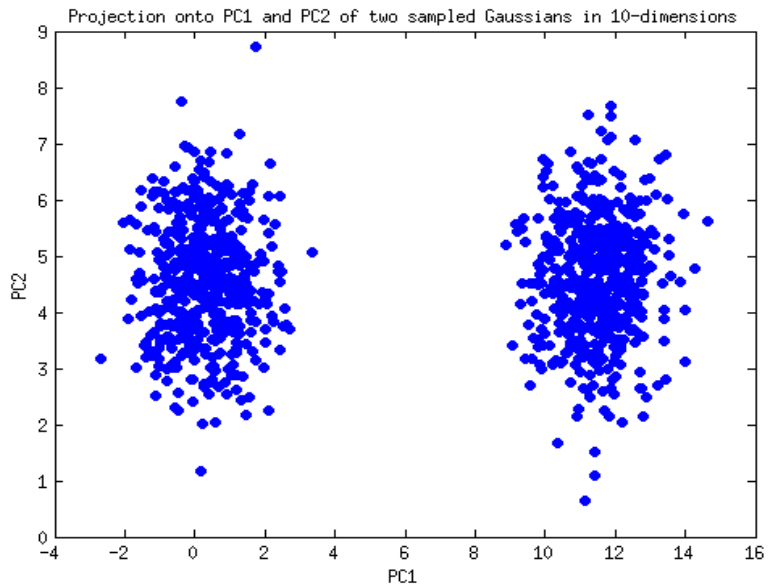
Two synthetic clusters in 10 dimensions

Multidimensional Scaling

- The strategy of projecting onto a lower-dimensional Euclidean space for visualization is referred to as “Multidimensional Scaling”



Face movement during movie



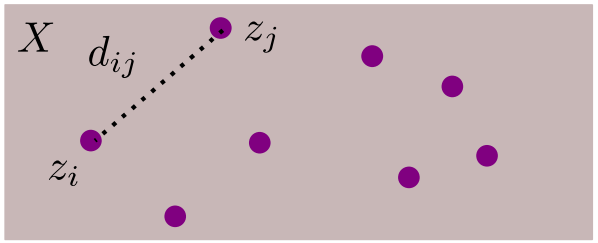
Two synthetic clusters in 10 dimensions

Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

for distances $d_{ij} = d(z_i, z_j)$

dataset $\{z_n\}_{n=1}^N \subset X$ general data space



Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

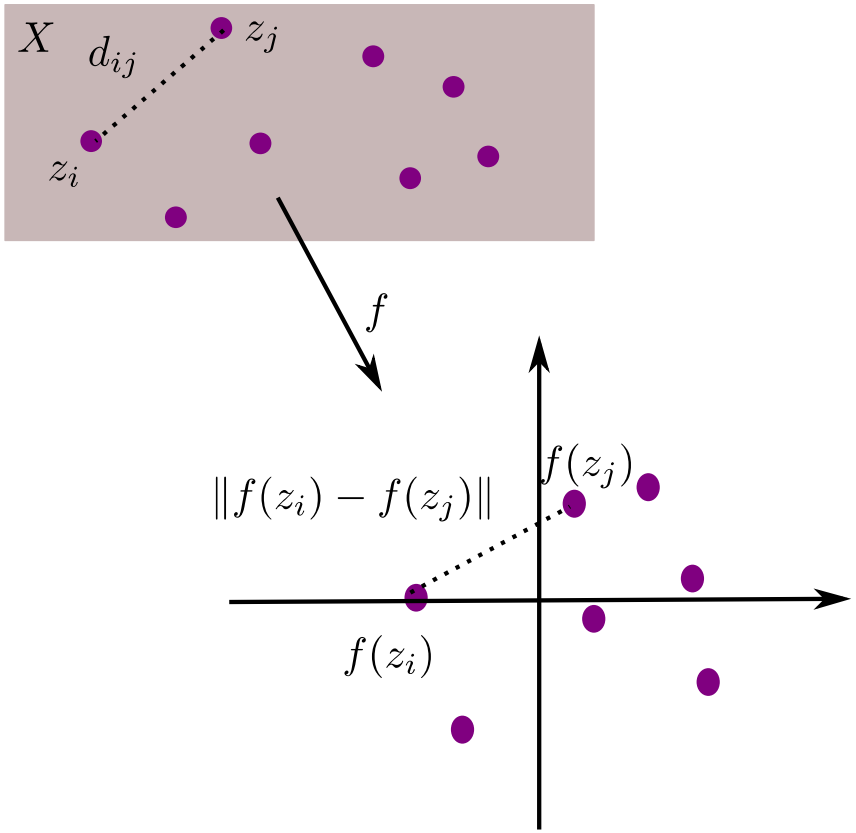
for distances $d_{ij} = d(z_i, z_j)$

dataset $\{z_n\}_{n=1}^N \subset X$ general data space

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.



Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

for distances $d_{ij} = d(z_i, z_j)$

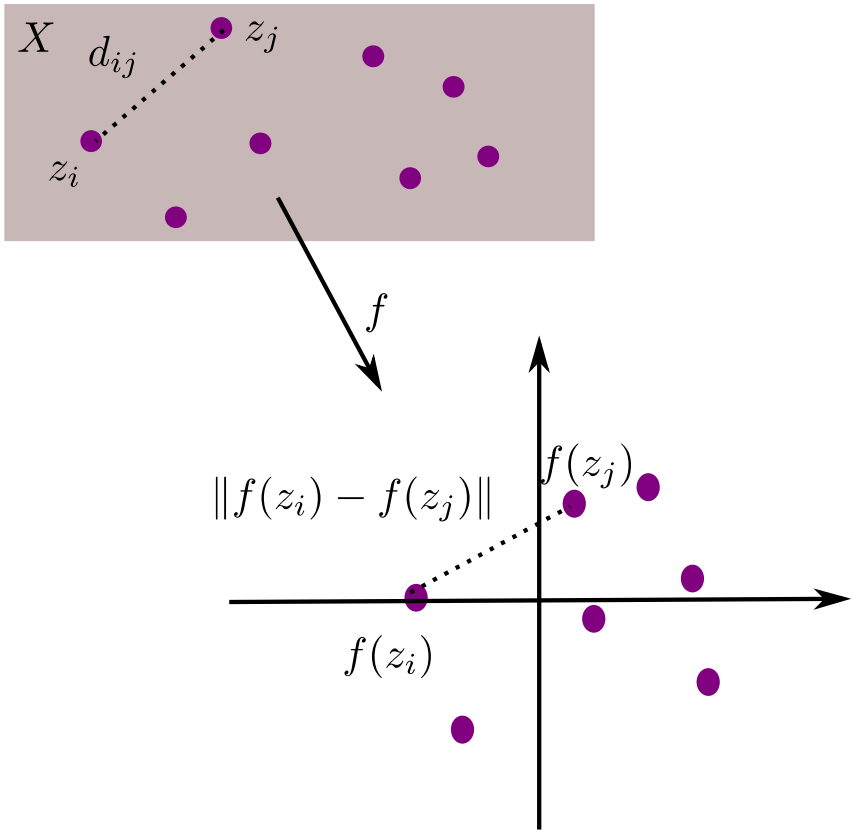
dataset $\{z_n\}_{n=1}^N \subset X$ general data space

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, distances are preserved as well as possible.



Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

for distances $d_{ij} = d(z_i, z_j)$

dataset $\{z_n\}_{n=1}^N \subset X$ general data space

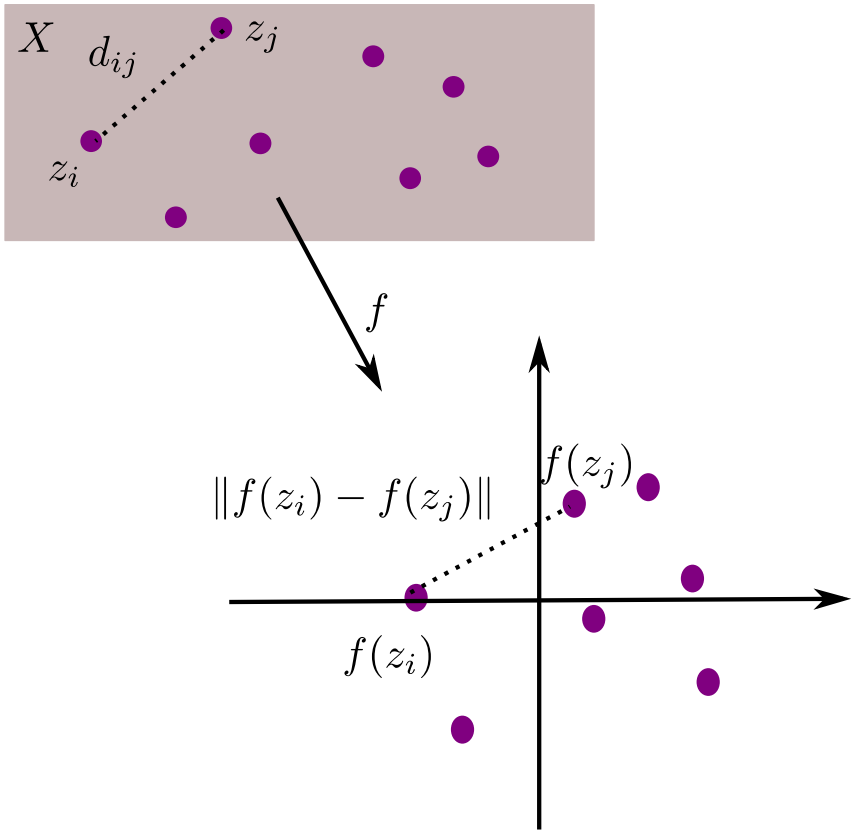
Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, distances are preserved as well as possible.

Assume $X = \mathbb{R}^k$ for $k >> 0$



Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

for distances $d_{ij} = d(z_i, z_j)$

dataset $\{z_n\}_{n=1}^N \subset X$ general data space

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

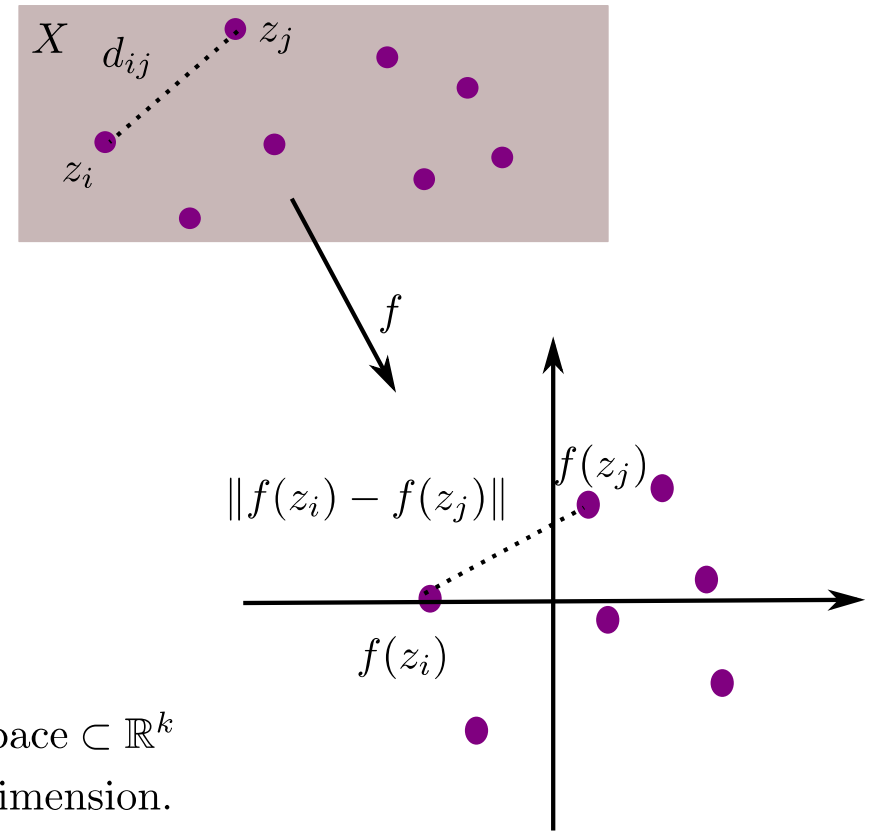
$$\Phi(Y) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, distances are preserved as well as possible.

Assume $X = \mathbb{R}^k$ for $k >> 0$

$f(z) = Mz$ linear projection onto linear subspace $\subset \mathbb{R}^k$ of low dimension.

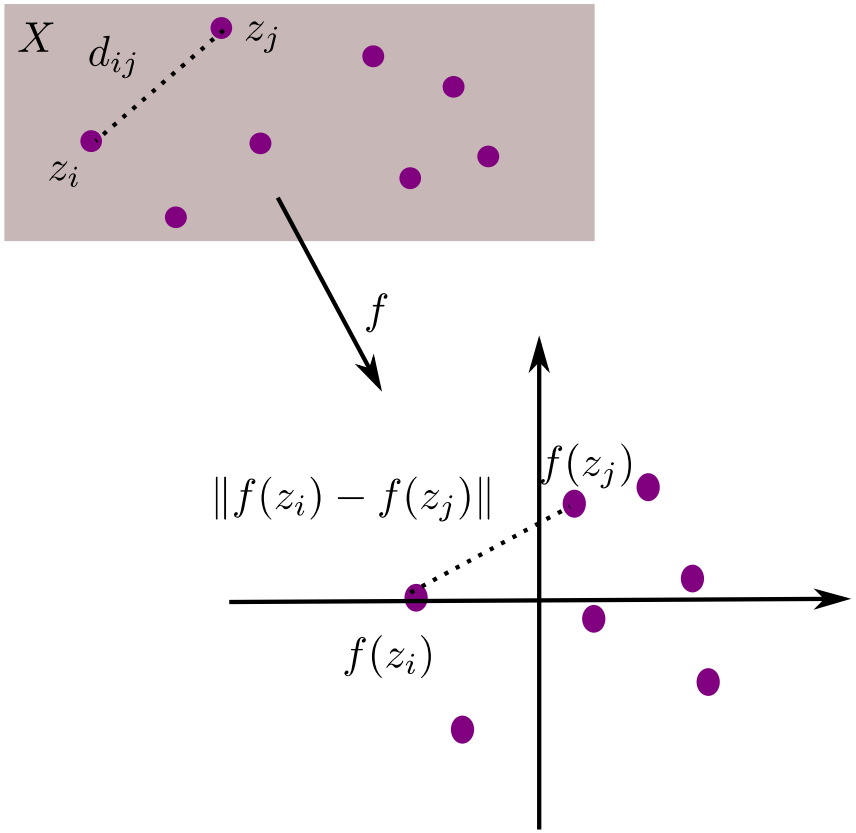


Classical Multidimensional Scaling

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.



Classical Multidimensional Scaling

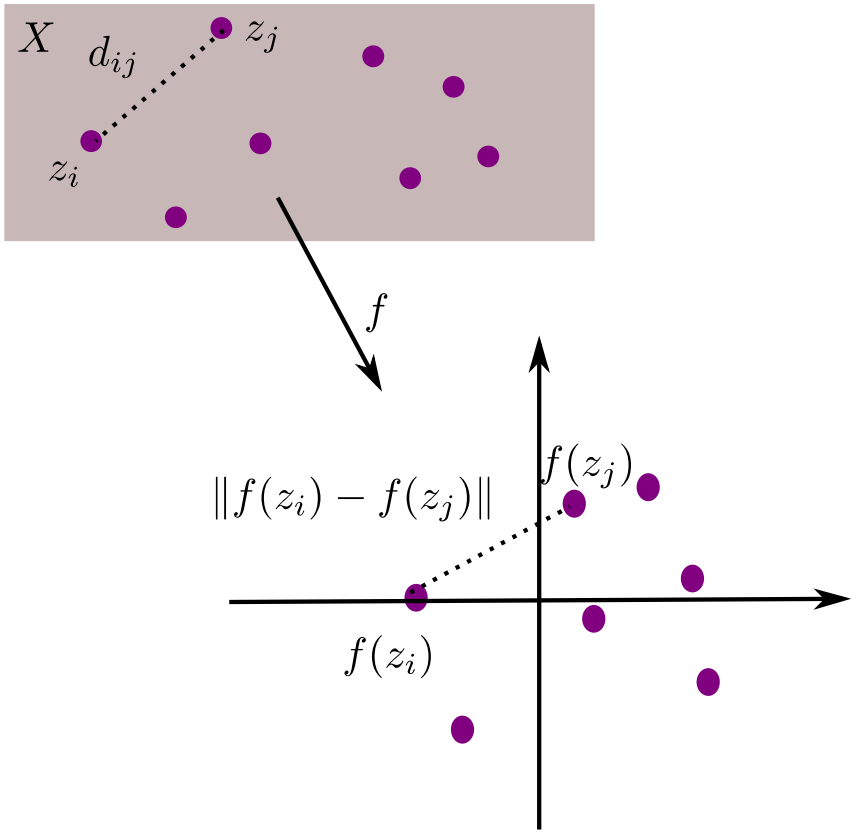
Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, we seek

$$\operatorname{argmin} \sum_{i,j} (d_{ij}^2 - \|Mz_i - Mz_j\|^2)$$



Classical Multidimensional Scaling

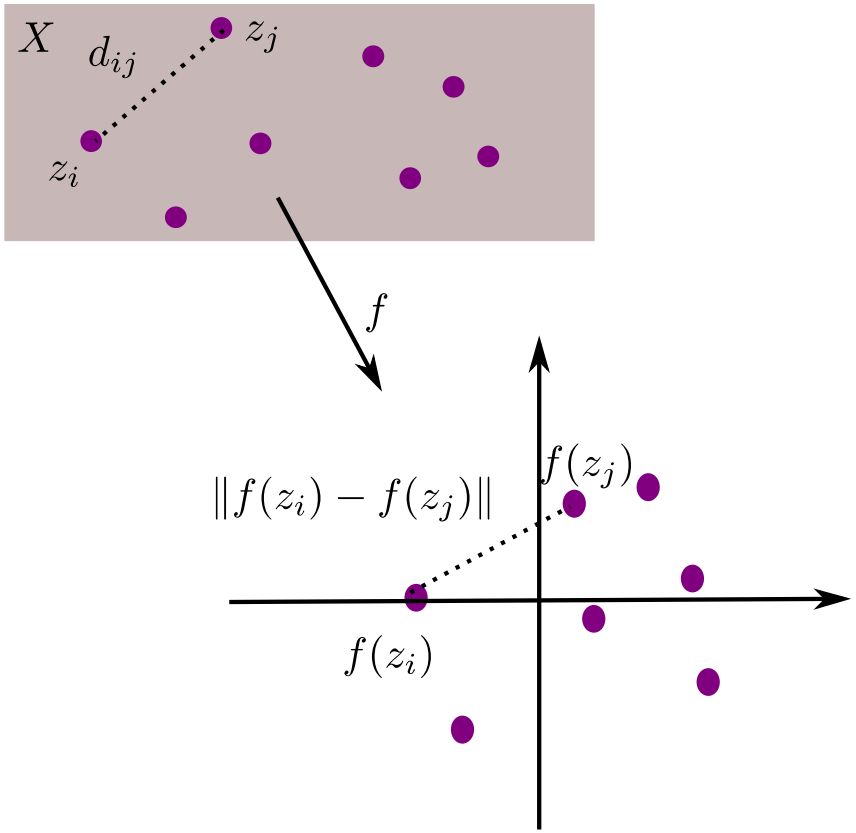
Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, we seek

$$\begin{aligned} &\operatorname{argmin} \textstyle\sum_{i,j} (d_{ij}^2 - \|Mz_i - Mz_j\|^2) \\ &= \operatorname{argmin} \left(\textstyle\sum_{i,j} d_{ij}^2 - \sum_{i,j} \|Mz_i - Mz_j\|^2 \right) \end{aligned}$$



Classical Multidimensional Scaling

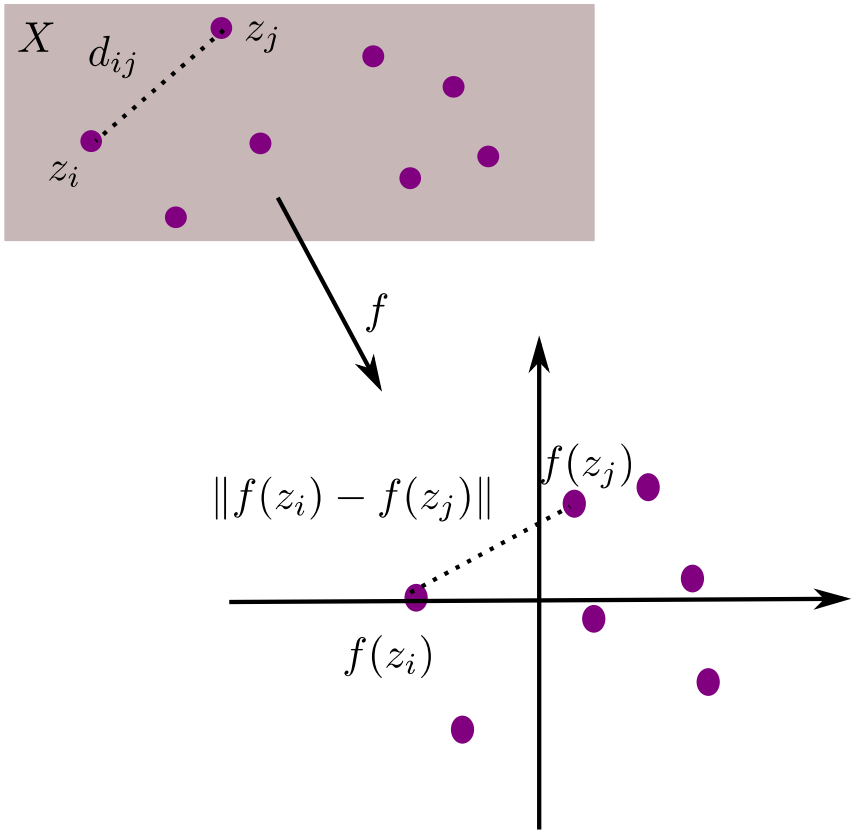
Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, we seek

$$\begin{aligned} &\operatorname{argmin} \textstyle\sum_{i,j} (d_{ij}^2 - \|Mz_i - Mz_j\|^2) \\ &= \operatorname{argmin} \left(\textstyle\sum_{i,j} d_{ij}^2 - \sum_{i,j} \|Mz_i - Mz_j\|^2 \right) \\ &= \operatorname{argmax} \textstyle\sum_{i,j} \|Mz_i - Mz_j\|^2 \end{aligned}$$



Multidimensional Scaling

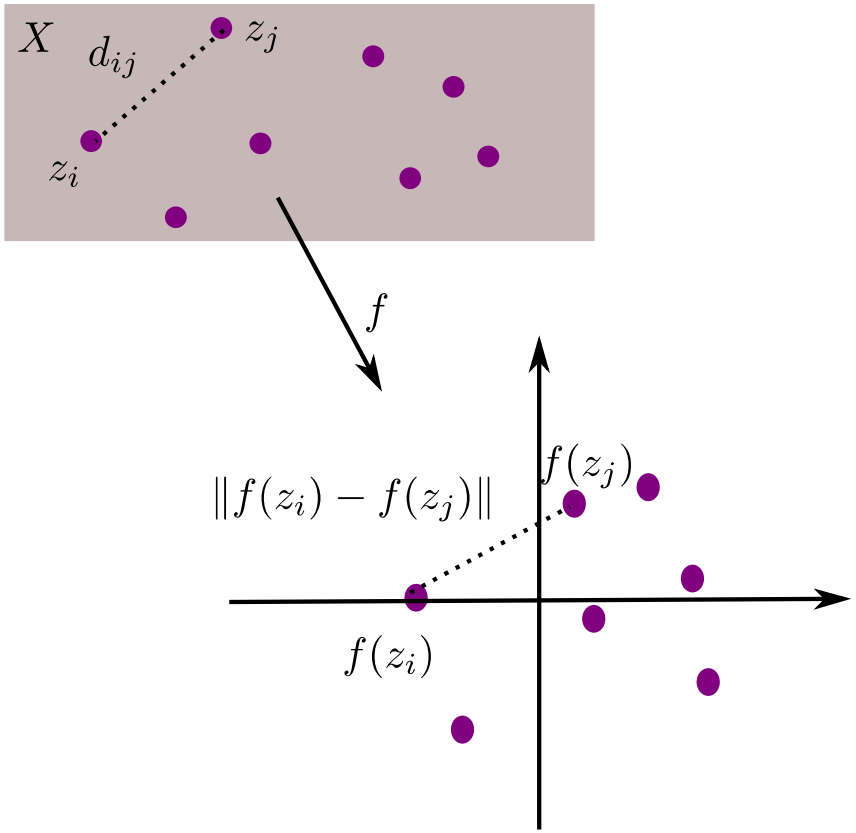
Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, we seek

$$\begin{aligned}
&\operatorname{argmin} \textstyle\sum_{i,j} (d_{ij}^2 - \|Mz_i - Mz_j\|^2) \\
&= \operatorname{argmin} \left(\textstyle\sum_{i,j} d_{ij}^2 - \sum_{i,j} \|Mz_i - Mz_j\|^2 \right) \\
&= \operatorname{argmax} \textstyle\sum_{i,j} \|Mz_i - Mz_j\|^2 \\
&= \operatorname{argmax} 2N \textstyle\sum_i \|Mz_i - \frac{1}{N} \sum_j Mz_j\|^2
\end{aligned}$$



Classical Multidimensional Scaling

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

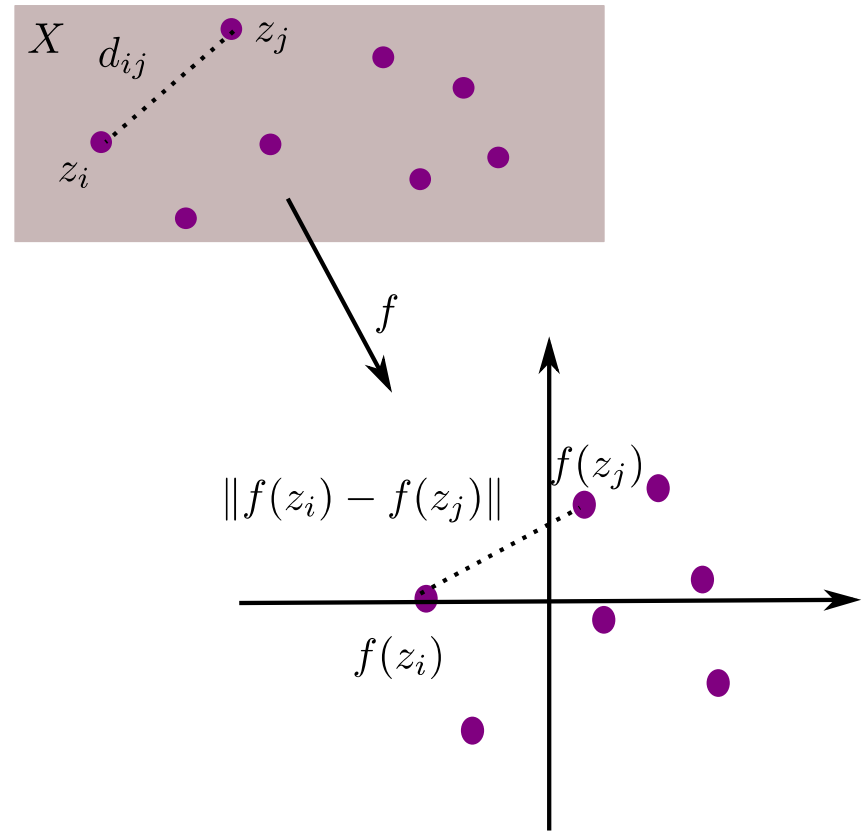
$$\Phi(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

That is, we seek

$$\begin{aligned}
&\operatorname{argmin} \textstyle\sum_{i,j} (d_{ij}^2 - \|Mz_i - Mz_j\|^2) \\
&= \operatorname{argmin} \left(\textstyle\sum_{i,j} d_{ij}^2 - \sum_{i,j} \|Mz_i - Mz_j\|^2 \right) \\
&= \operatorname{argmax} \textstyle\sum_{i,j} \|Mz_i - Mz_j\|^2 \\
&= \operatorname{argmax} 2N \textstyle\sum_i \|Mz_i - \frac{1}{N} \sum_j Mz_j\|^2
\end{aligned}$$

equivalent to maximizing projected variance.



Classical Multidimensional Scaling

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

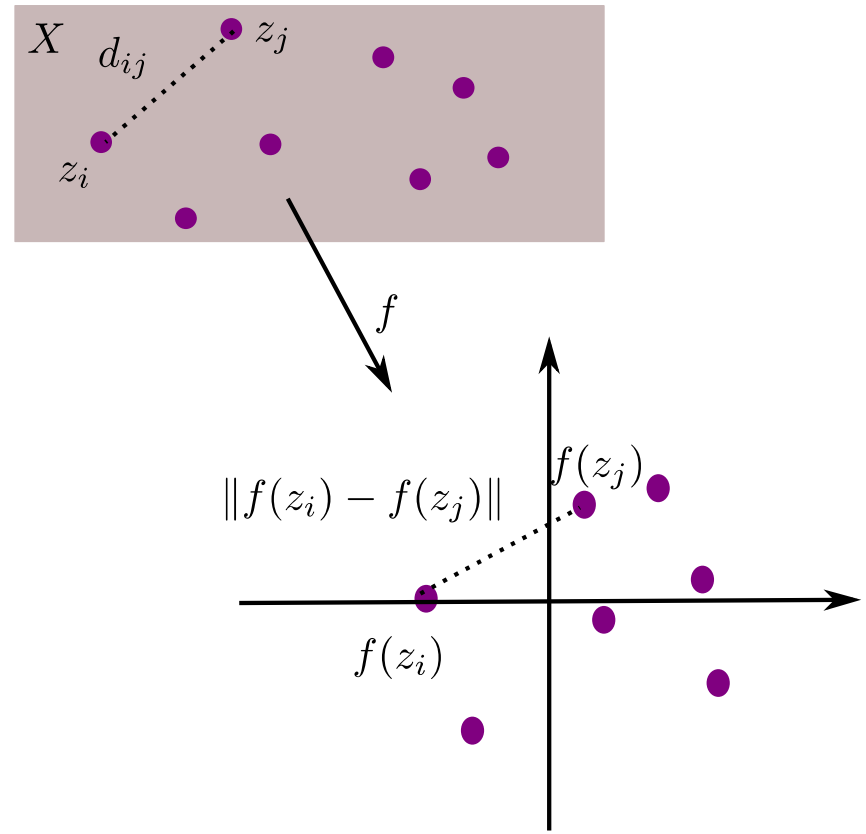
is minimized.

That is, we seek

$$\begin{aligned}
&\operatorname{argmin} \textstyle\sum_{i,j} (d_{ij}^2 - \|Mz_i - Mz_j\|^2) \\
&= \operatorname{argmin} \left(\textstyle\sum_{i,j} d_{ij}^2 - \sum_{i,j} \|Mz_i - Mz_j\|^2 \right) \\
&= \operatorname{argmax} \textstyle\sum_{i,j} \|Mz_i - Mz_j\|^2 \\
&= \operatorname{argmax} 2N \textstyle\sum_i \|Mz_i - \frac{1}{N} \sum_j Mz_j\|^2
\end{aligned}$$

equivalent to maximizing projected variance.

Familiar?



Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

for distances $d_{ij} = d(z_i, z_j)$

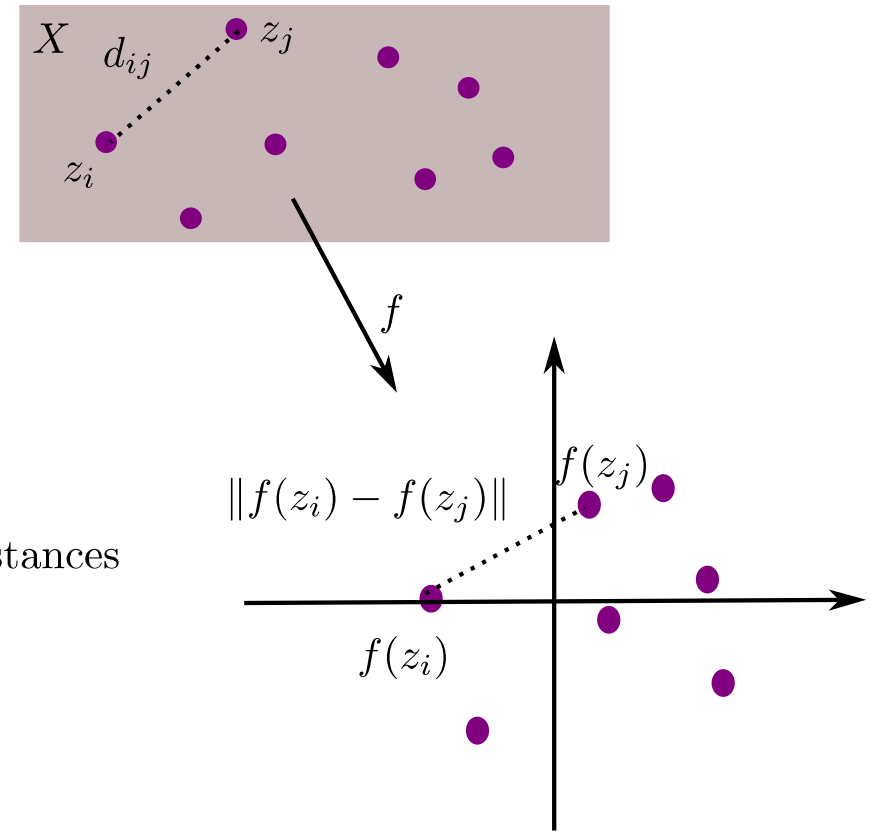
dataset $\{z_n\}_{n=1}^N \subset X$ general data space

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

New interpretation of PCA: Preserving squared distances



Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

for distances $d_{ij} = d(z_i, z_j)$

dataset $\{z_n\}_{n=1}^N \subset X$ general data space

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

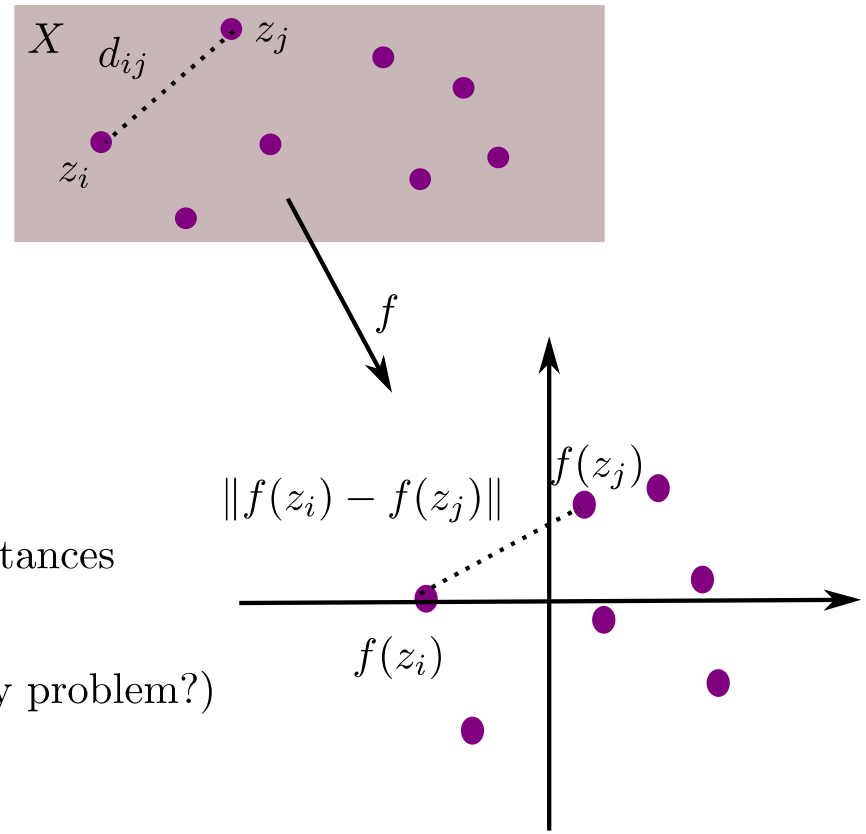
$$\Phi(Y) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

is minimized.

New interpretation of PCA: Preserving squared distances

Revealing problem with PCA:

Preserves long distances better than short ones (why problem?)



Classical Multidimensional Scaling

Input: Distance matrix $D = (d_{ij})$

for distances $d_{ij} = d(z_i, z_j)$

dataset $\{z_n\}_{n=1}^N \subset X$ general data space

Goal: Find mapping $f\colon X \to \mathbb{R}^d$ for small d such that

$$\Phi(Y) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij}^2 - \|f(z_i) - f(z_j)\|^2)$$

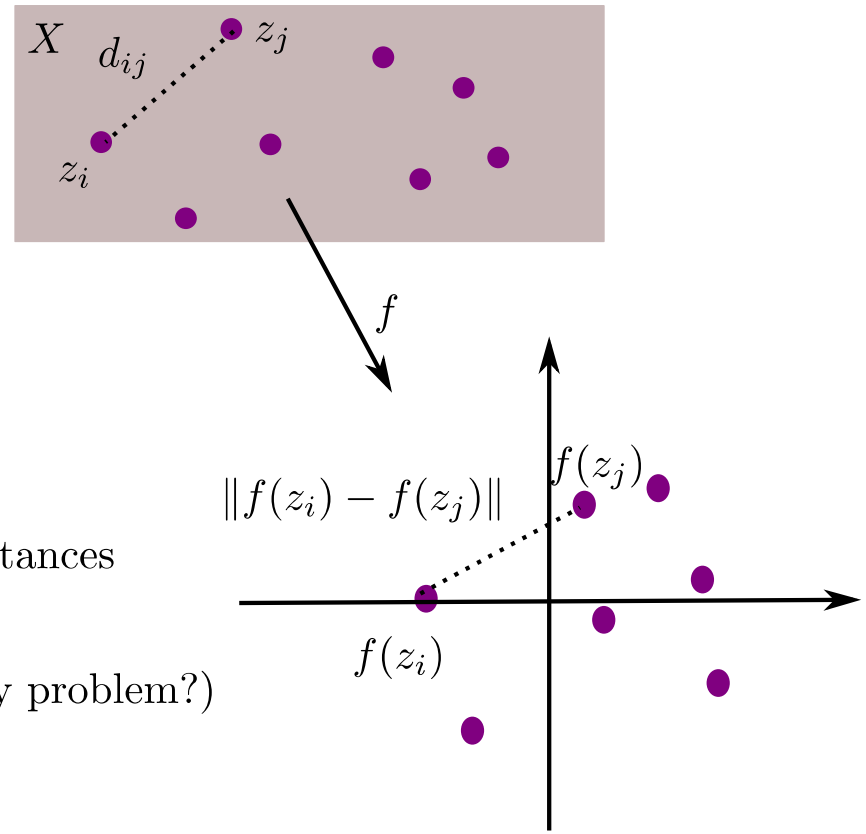
is minimized.

New interpretation of PCA: Preserving squared distances

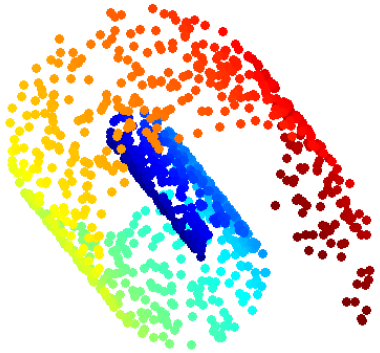
Revealing problem with PCA:

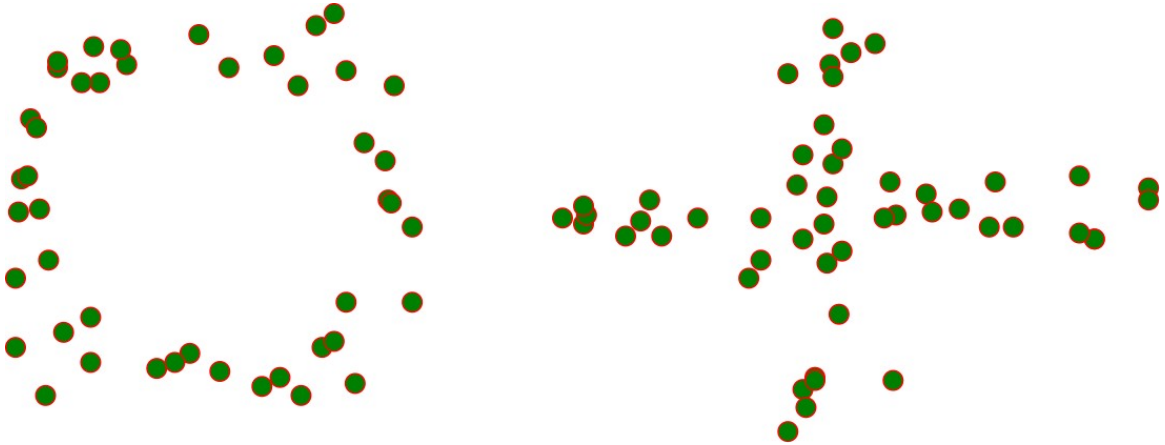
Preserves long distances better than short ones (why problem?)

MDS can only preserve linear distances well (why?)



What PCA and MDS cannot do



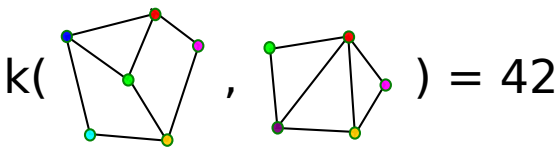


Lectures with Christian: Kernels and kernel methods

Definition

Given an input data space X, a *kernel* is a "similarity score"

$$k\colon X \times X \to \mathbb{R}$$



such that the *kernel matrix*

$$K = (K_{ij}) = \begin{pmatrix} k(z_1, z_1) & k(z_1, z_2) & \dots & k(z_1, z_N) \\ k(z_2, z_1) & k(z_2, z_2) & \dots & k(z_2, z_N) \\ \vdots & & \ddots & \vdots \\ k(z_N, z_1) & k(z_N, z_2) & \dots & k(z_N, z_N) \end{pmatrix}$$

is symmetric and positive definite for any dataset $\{z_1, \dots, z_N\} \subset X$

Lectures with Christian: Kernels and kernel methods

You learned from Christian:

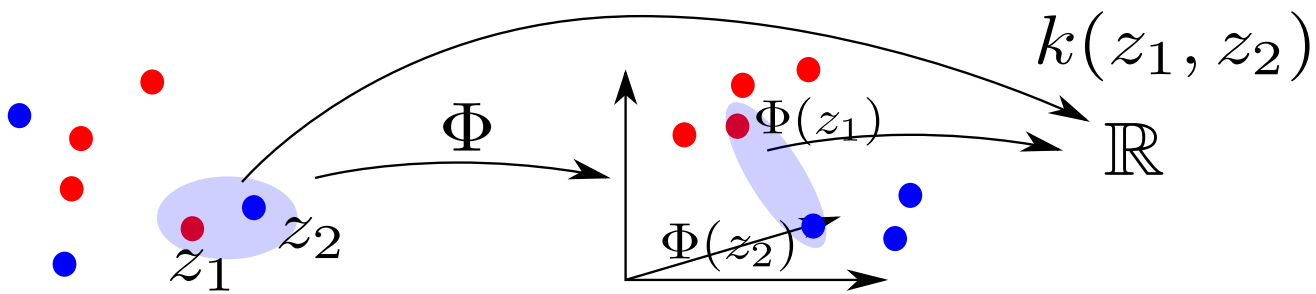
If $k\colon X \times X \to \mathbb{R}$ is a kernel, then there exists a Hilbert feature space $\mathcal{H}$ and a feature map

$$\Phi\colon X \to \mathcal{H}$$

such that k defines an inner product on the linear space $\mathcal{H}$:

$$k(z_i, z_j) = \langle \Phi(z_i), \Phi(z_j) \rangle.$$

Here, $\langle \cdot, \cdot \rangle$ is another way of writing "dot product" in $\mathcal{H}$



set of points z_n

Lectures with Christian: Kernels and kernel methods

You learned from Christian:

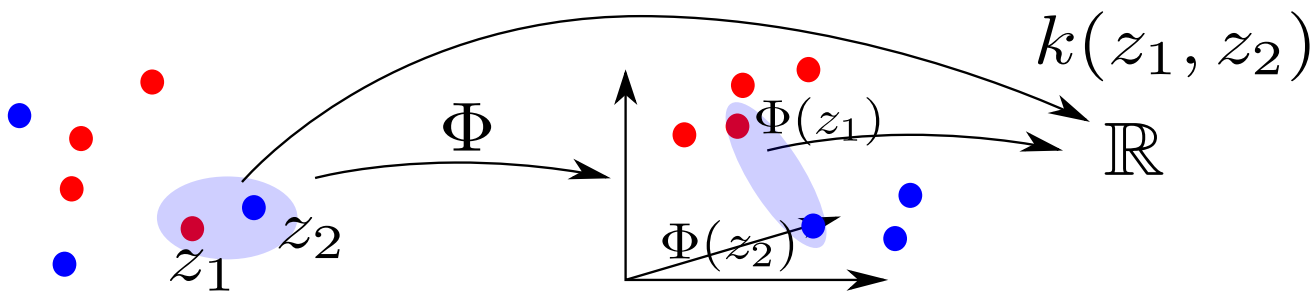
If $k\colon X \times X \to \mathbb{R}$ is a kernel, then there exists a Hilbert feature space $\mathcal{H}$ and a feature map

$$\Phi\colon X \to \mathcal{H}$$

such that k defines an inner product on the linear space $\mathcal{H}$:

$$k(z_i, z_j) = \langle \Phi(z_i), \Phi(z_j) \rangle.$$

Here, $\langle \cdot, \cdot \rangle$ is another way of writing "dot product" in $\mathcal{H}$



set of points z_n

Why is this cool?

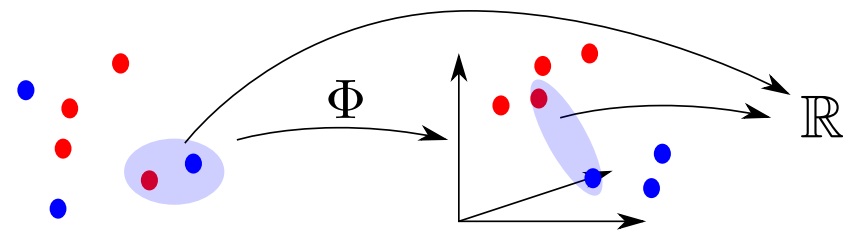
$\mathcal{H}$ is linear! We know how to do linear analysis (sorta)!

The nonlinear Φ lets us do nonlinear analysis with linear methods!

Lectures with Christian: Kernels and kernel methods

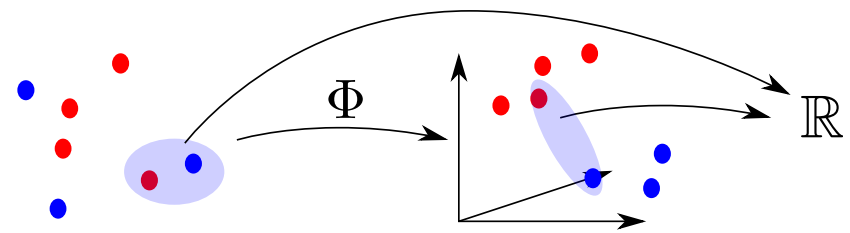
Kernel methods:

- We know how to do statistics in $\mathcal{H}$
- We do not know (or want to deal with) Φ (or $\mathcal{H}$)
- **Instead:** Rewrite statistics in $\mathcal{H}$ in terms of the kernel value.
- **How?**



Lectures with Christian: Kernels and kernel methods

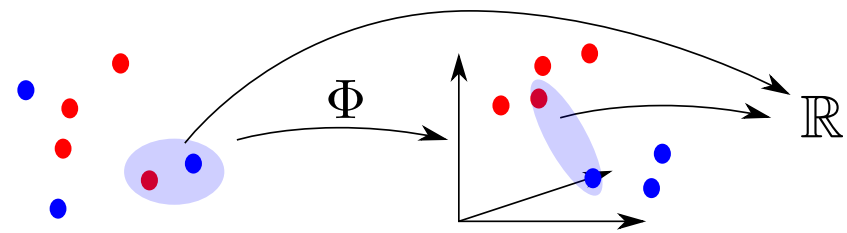
Kernel methods:



- We know how to do statistics in $\mathcal{H}$
- We do not know (or want to deal with) Φ (or $\mathcal{H}$)
- **Instead:** Rewrite statistics in $\mathcal{H}$ in terms of the kernel value.
- **How?** Rewrite original statistics, or functions used in original statistics, through inner products.

Lectures with Christian: Kernels and kernel methods

Kernel methods:

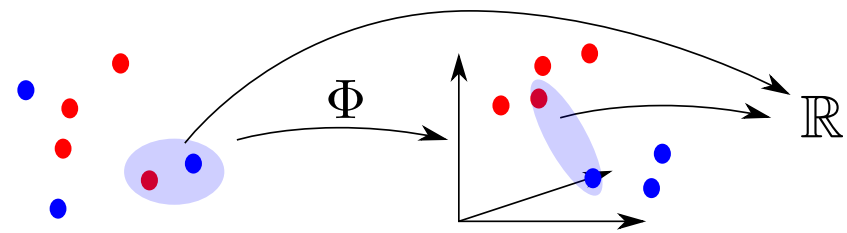


- We know how to do statistics in $\mathcal{H}$
- We do not know (or want to deal with) Φ (or $\mathcal{H}$)
- **Instead:** Rewrite statistics in $\mathcal{H}$ in terms of the kernel value.
- **How?** Rewrite original statistics, or functions used in original statistics, through inner products.
- **Example:** Distance $\|x_1 - x_2\|$:

$$\|x_1 - x_2\|^2 = \langle x_1 - x_2, x_1 - x_2 \rangle = \langle x_1, x_1 \rangle - 2\langle x_1, x_2 \rangle + \langle x_2, x_2 \rangle$$

Lectures with Christian: Kernels and kernel methods

Kernel methods:



- We know how to do statistics in $\mathcal{H}$
- We do not know (or want to deal with) Φ (or $\mathcal{H}$)
- **Instead:** Rewrite statistics in $\mathcal{H}$ in terms of the kernel value.
- **How?** Rewrite original statistics, or functions used in original statistics, through inner products.
- **Example:** Distance $\|x_1 - x_2\|$:

$$\|x_1 - x_2\|^2 = \langle x_1 - x_2, x_1 - x_2 \rangle = \langle x_1, x_1 \rangle - 2\langle x_1, x_2 \rangle + \langle x_2, x_2 \rangle$$

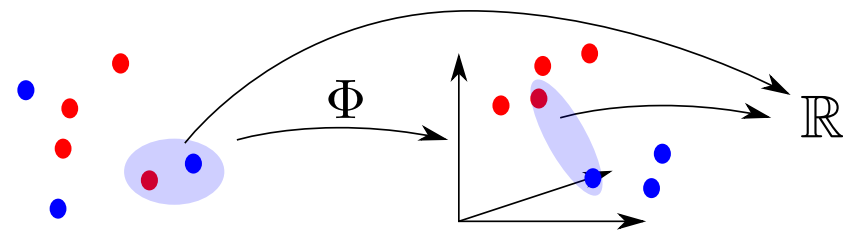
so the kernel defines a distance between the original data points:

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\|^2 = \langle \Phi(z_1) - \Phi(z_2), \Phi(z_1) - \Phi(z_2) \rangle$$

$$= \langle \Phi(z_1), \Phi(z_1) \rangle - 2\langle \Phi(z_1), \Phi(z_2) \rangle + \langle \Phi(z_2), \Phi(z_2) \rangle$$

Lectures with Christian: Kernels and kernel methods

Kernel methods:



- We know how to do statistics in $\mathcal{H}$
- We do not know (or want to deal with) Φ (or $\mathcal{H}$)
- **Instead:** Rewrite statistics in $\mathcal{H}$ in terms of the kernel value.
- **How?** Rewrite original statistics, or functions used in original statistics, through inner products.
- **Example:** Distance $\|x_1 - x_2\|$:

$$\|x_1 - x_2\|^2 = \langle x_1 - x_2, x_1 - x_2 \rangle = \langle x_1, x_1 \rangle - 2\langle x_1, x_2 \rangle + \langle x_2, x_2 \rangle$$

so the kernel defines a distance between the original data points:

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\|^2 = \langle \Phi(z_1) - \Phi(z_2), \Phi(z_1) - \Phi(z_2) \rangle$$

$$= \langle \Phi(z_1), \Phi(z_1) \rangle - 2\langle \Phi(z_1), \Phi(z_2) \rangle + \langle \Phi(z_2), \Phi(z_2) \rangle$$

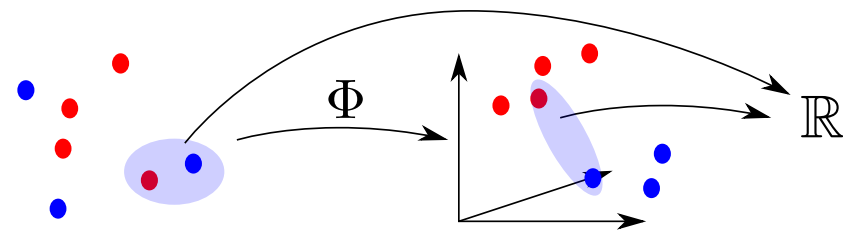
Today's problem: Can we do PCA in $\mathcal{H}$?

Sanity check:

- Why is this a good idea?

Lectures with Christian: Kernels and kernel methods

Kernel methods:



- We know how to do statistics in $\mathcal{H}$
- We do not know (or want to deal with) Φ (or $\mathcal{H}$)
- **Instead:** Rewrite statistics in $\mathcal{H}$ in terms of the kernel value.
- **How?** Rewrite original statistics, or functions used in original statistics, through inner products.
- **Example:** Distance $\|x_1 - x_2\|$:

$$\|x_1 - x_2\|^2 = \langle x_1 - x_2, x_1 - x_2 \rangle = \langle x_1, x_1 \rangle - 2\langle x_1, x_2 \rangle + \langle x_2, x_2 \rangle$$

so the kernel defines a distance between the original data points:

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\|^2 = \langle \Phi(z_1) - \Phi(z_2), \Phi(z_1) - \Phi(z_2) \rangle$$

$$= \langle \Phi(z_1), \Phi(z_1) \rangle - 2\langle \Phi(z_1), \Phi(z_2) \rangle + \langle \Phi(z_2), \Phi(z_2) \rangle$$

Today's problem: Can we do PCA in $\mathcal{H}$?

Sanity check:

- Why is this a good idea?
  - Detect nonlinear structure
  - Reduce dimensionality when analysis done in $\mathcal{H}$

Kernel PCA

Conventional PCA:

- Data $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^D$
- Let's assume mean $\bar{\mathbf{x}} = \mathbf{0}$
- Compute covariance matrix

  $$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T$$

- PCs are eigenvectors, i.e. solutions $\mathbf{u}_i$ to

  $$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \; \mathbf{u}_i^T \mathbf{u}_i = 1$$

Kernel PCA

Conventional PCA:

- Data $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^D$
- Let's assume mean $\bar{\mathbf{x}} = \mathbf{0}$
- Compute covariance matrix

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T$$

- PCs are eigenvectors, i.e. solutions $\mathbf{u}_i$ to

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i,\ \mathbf{u}_i^T \mathbf{u}_i = 1$$

How do we turn this into a kernel procedure?

Kernel PCA

Conventional PCA:

- Data $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^D$
- Let's assume mean $\bar{\mathbf{x}} = \mathbf{0}$
- Compute covariance matrix

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T$$

- PCs are eigenvectors, i.e. solutions $\mathbf{u}_i$ to

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i,\ \mathbf{u}_i^T \mathbf{u}_i = 1$$

Kernel PCA:

- Data $\{z_n\}_{n=1}^N \subset X$, data space X
- Data mapped to features space $\{\Phi(z_n)\}_{n=1}^N \subset \mathcal{H}$
- Assume for now:
  - $\frac{1}{N}\sum_{n=1}^N \Phi(z_n) = \mathbf{0}$
  - $\dim(\mathcal{H}) = M < \infty$
- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^N \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Kernel PCA

Conventional PCA:

- Data $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^D$
- Let's assume mean $\bar{\mathbf{x}} = \mathbf{0}$
- Compute covariance matrix

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T$$

- PCs are eigenvectors, i.e. solutions $\mathbf{u}_i$ to

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i,\ \mathbf{u}_i^T \mathbf{u}_i = 1$$

Kernel PCA:

- Data $\{z_n\}_{n=1}^N \subset X$, data space X
- Data mapped to features space $\{\Phi(z_n)\}_{n=1}^N \subset \mathcal{H}$
- Assume for now:
  - $\frac{1}{N}\sum_{n=1}^N \Phi(z_n) = \mathbf{0}$
  - $\dim(\mathcal{H}) = M < \infty$
- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^N \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Task: Find $\mathbf{v_i}$ – or, more precisely, the projections $\Phi(z_n)^T \mathbf{v}_i$

Kernel PCA

We have

- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Kernel PCA

We have

- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Substitute $\mathbf{C}$ in:

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n) \underbrace{\Phi(z_n)^T \mathbf{v}_i}_{\text{scalar}} = \lambda_i \mathbf{v}_i$$

Kernel PCA

We have

- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Substitute $\mathbf{C}$ in:

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n) \underbrace{\Phi(z_n)^T \mathbf{v}_i}_{\text{scalar}} = \lambda_i \mathbf{v}_i$$

so $\quad \mathbf{v}_i = \sum_{n=1}^{N} a_{in}\Phi(z_n) \quad$ for some a_{in}.

Kernel PCA

We have

- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Substitute $\mathbf{C}$ in:

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n) \underbrace{\Phi(z_n)^T \mathbf{v}_i}_{\text{scalar}} = \lambda_i \mathbf{v}_i$$

so $\quad \mathbf{v}_i = \sum_{n=1}^{N} a_{in}\Phi(z_n) \quad$ for some a_{in}.

Task: Find a_{in}.

Kernel PCA

We have

- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Substitute $\mathbf{C}$ in:

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n) \underbrace{\Phi(z_n)^T \mathbf{v}_i}_{\text{scalar}} = \lambda_i \mathbf{v}_i \qquad \bullet$$

so $\quad \mathbf{v}_i = \sum_{n=1}^{N} a_{in}\Phi(z_n) \quad$ for some a_{in}. $\bullet$

Task: Find a_{in}.

From $\bullet$ and $\bullet$

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T \sum_{m=1}^{N} a_{im}\Phi(z_m) = \lambda_i \sum_{n=1}^{N} a_{in}\Phi(z_n)$$

Kernel PCA

We have

- Covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T$
- PCs solutions $\mathbf{v}_i$ to $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$, $\mathbf{v}_i \in \mathcal{H}$

Substitute $\mathbf{C}$ in:

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n) \underbrace{\Phi(z_n)^T \mathbf{v}_i}_{\text{scalar}} = \lambda_i \mathbf{v}_i \qquad \bullet$$

so $\quad \mathbf{v}_i = \sum_{n=1}^{N} a_{in}\Phi(z_n) \quad$ for some a_{in}. $\bullet$

Task: Find a_{in}.

From $\bullet$ and $\bullet$

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n)\Phi(z_n)^T \sum_{m=1}^{N} a_{im}\Phi(z_m) = \lambda_i \sum_{n=1}^{N} a_{in}\Phi(z_n)$$

rearranged:

$$\frac{1}{N}\sum_{n=1}^{N} \Phi(z_n) \sum_{m=1}^{N} a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i \sum_{m=1}^{N} a_{im}\Phi(z_m)$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Multiply both sides with $\Phi(z_l)^T$:

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_l)^T\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_l)^T\Phi(z_m)$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Multiply both sides with $\Phi(z_l)^T$:

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_l)^T\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_l)^T\Phi(z_m)$$

See any kernels?

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Multiply both sides with $\Phi(z_l)^T$:

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_l)^T\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_l)^T\Phi(z_m)$$

See any kernels?

$$\frac{1}{N}\sum_{n=1}^{N}k(z_l,z_n)\sum_{m=1}^{N}a_{im}k(z_n,z_m) = \lambda_i\sum_{m=1}^{N}a_{im}k(z_l,z_m)$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Multiply both sides with $\Phi(z_l)^T$:

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_l)^T\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_l)^T\Phi(z_m)$$

See any kernels?

$$\frac{1}{N}\sum_{n=1}^{N}k(z_l,z_n)\sum_{m=1}^{N}a_{im}k(z_n,z_m) = \lambda_i\sum_{m=1}^{N}a_{im}k(z_l,z_m)$$

In matrix form:

$$K^2\mathbf{a}_i = \lambda_i N K\mathbf{a}_i \qquad \mathbf{a}_i = (a_{1i}, a_{2i}, \ldots, a_{Ni})^T$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Multiply both sides with $\Phi(z_l)^T$:

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_l)^T\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_l)^T\Phi(z_m)$$

See any kernels?

$$\frac{1}{N}\sum_{n=1}^{N}k(z_l,z_n)\sum_{m=1}^{N}a_{im}k(z_n,z_m) = \lambda_i\sum_{m=1}^{N}a_{im}k(z_l,z_m)$$

In matrix form:

$$K^2\mathbf{a}_i = \lambda_i N K\mathbf{a}_i \qquad \mathbf{a}_i = (a_{1i}, a_{2i}, \ldots, a_{Ni})^T$$

Solve eigenvalue problem:

$$K\mathbf{a}_i = \lambda_i N\mathbf{a}_i \qquad (K \text{ positive definite} \Rightarrow \text{invertible})$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Multiply both sides with $\Phi(z_l)^T$:

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_l)^T\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_l)^T\Phi(z_m)$$

See any kernels?

$$\frac{1}{N}\sum_{n=1}^{N}k(z_l,z_n)\sum_{m=1}^{N}a_{im}k(z_n,z_m) = \lambda_i\sum_{m=1}^{N}a_{im}k(z_l,z_m)$$

In matrix form:

$$K^2\mathbf{a}_i = \lambda_i N K\mathbf{a}_i \qquad \mathbf{a}_i = (a_{1i}, a_{2i}, \ldots, a_{Ni})^T$$

Solve eigenvalue problem:

$$K\mathbf{a}_i = \lambda_i N\mathbf{a}_i \qquad (K \text{ positive definite} \Rightarrow \text{invertible})$$

Normalization:

$$1 = \mathbf{v}_i^T\mathbf{v}_i = \sum_{n=1}^{N}\sum_{m=1}^{N}a_{in}a_{im}\underbrace{\Phi(z_n)^T\Phi(z_m))}_{k(z_n,z_m)}\mathbf{a}_i^T K\mathbf{a}_i = \lambda_i N\mathbf{a}_i^T\mathbf{a}_i$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_m)$$

Multiply both sides with $\Phi(z_l)^T$:

$$\frac{1}{N}\sum_{n=1}^{N}\Phi(z_l)^T\Phi(z_n)\sum_{m=1}^{N}a_{im}\Phi(z_n)^T\Phi(z_m) = \lambda_i\sum_{m=1}^{N}a_{im}\Phi(z_l)^T\Phi(z_m)$$

See any kernels?

$$\frac{1}{N}\sum_{n=1}^{N}k(z_l,z_n)\sum_{m=1}^{N}a_{im}k(z_n,z_m) = \lambda_i\sum_{m=1}^{N}a_{im}k(z_l,z_m)$$

In matrix form:

$$K^2\mathbf{a}_i = \lambda_i N K\mathbf{a}_i \qquad \mathbf{a}_i = (a_{1i}, a_{2i}, \ldots, a_{Ni})^T$$

Solve eigenvalue problem:

$$K\mathbf{a}_i = \lambda_i N\mathbf{a}_i \qquad (K \text{ positive definite} \Rightarrow \text{invertible})$$

Normalization:

$$1 = \mathbf{v}_i^T\mathbf{v}_i = \sum_{n=1}^{N}\sum_{m=1}^{N}a_{in}a_{im}\underbrace{\Phi(z_n)^T\Phi(z_m)}_{k(z_n,z_m)})\mathbf{a}_i^T K\mathbf{a}_i = \lambda_i N\mathbf{a}_i^T\mathbf{a}_i$$

Projecting $\Phi(z)$ onto i^{th} eigenvector $\mathbf{v}_i$ gives:

$$y_i(z) = \Phi(z)^T\mathbf{v}_i = \sum_{n=1}^{N}a_{in}\Phi(z)^T\Phi(z_n) = \sum_{n=1}^{N}a_{in}k(z,z_n)$$

Kernel PCA: Translate to zero mean

If the $\Phi(z_n)$ do not have zero mean, write

$$\tilde{\Phi}(z_n) = \Phi(z_n) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_l)$$

Kernel PCA: Translate to zero mean

If the $\Phi(z_n)$ do not have zero mean, write

$$\tilde{\Phi}(z_n) = \Phi(z_n) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_l)$$

and obtain a kernel matrix for zero mean data in $\mathcal{H}$ as

$$\tilde{K}_{nm} = \tilde{\Phi}(z_n)^T\tilde{\Phi}(z_m)$$

$$= \Phi(z_n)^T\Phi(z_m) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_n)^T\Phi(z_l) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_l)^T\Phi(z_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\Phi(z_j)^T\Phi(z_l)$$

Kernel PCA: Translate to zero mean

If the $\Phi(z_n)$ do not have zero mean, write

$$\tilde{\Phi}(z_n) = \Phi(z_n) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_l)$$

and obtain a kernel matrix for zero mean data in $\mathcal{H}$ as

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\Phi}(z_n)^T\tilde{\Phi}(z_m) \\
&= \Phi(z_n)^T\Phi(z_m) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_n)^T\Phi(z_l) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_l)^T\Phi(z_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\Phi(z_j)^T\Phi(z_l) \\
&= k(z_n, z_m) - \frac{1}{N}\sum_{l=1}^{N}k(z_l, z_n) - \frac{1}{N}\sum_{l=1}^{N}k(z_m, z_l) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(z_j, z_l)
\end{aligned}$$

Kernel PCA: Translate to zero mean

If the $\Phi(z_n)$ do not have zero mean, write

$$\tilde{\Phi}(z_n) = \Phi(z_n) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_l)$$

and obtain a kernel matrix for zero mean data in $\mathcal{H}$ as

$$\tilde{K}_{nm} = \tilde{\Phi}(z_n)^T\tilde{\Phi}(z_m)$$

$$= \Phi(z_n)^T\Phi(z_m) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_n)^T\Phi(z_l) - \frac{1}{N}\sum_{l=1}^{N}\Phi(z_l)^T\Phi(z_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\Phi(z_j)^T\Phi(z_l)$$

$$= k(z_n, z_m) - \frac{1}{N}\sum_{l=1}^{N}k(z_l, z_n) - \frac{1}{N}\sum_{l=1}^{N}k(z_m, z_l) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(z_j, z_l)$$

In matrix form: $\tilde{K} = K - 1_N K - K 1_N + 1_N K 1_N$ where $(1_N)_{ij} = \frac{1}{N}$

Now apply the previous algorithm to $\tilde{K}$...

Kernel PCA algorithm summarized

1. Compute kernel matrix $K = (K_{ij})$

$$K_{ij} = k(z_i, z_j)$$

2. Translate features $\Phi(z_n)$ to zero mean:

$$\tilde{K} = K - 1_N K - K 1_N + 1_N K 1_N$$

3. Solve eigenvalue problem:

$$\tilde{K}\mathbf{a}_i = \lambda_i \mathbf{a}_i$$

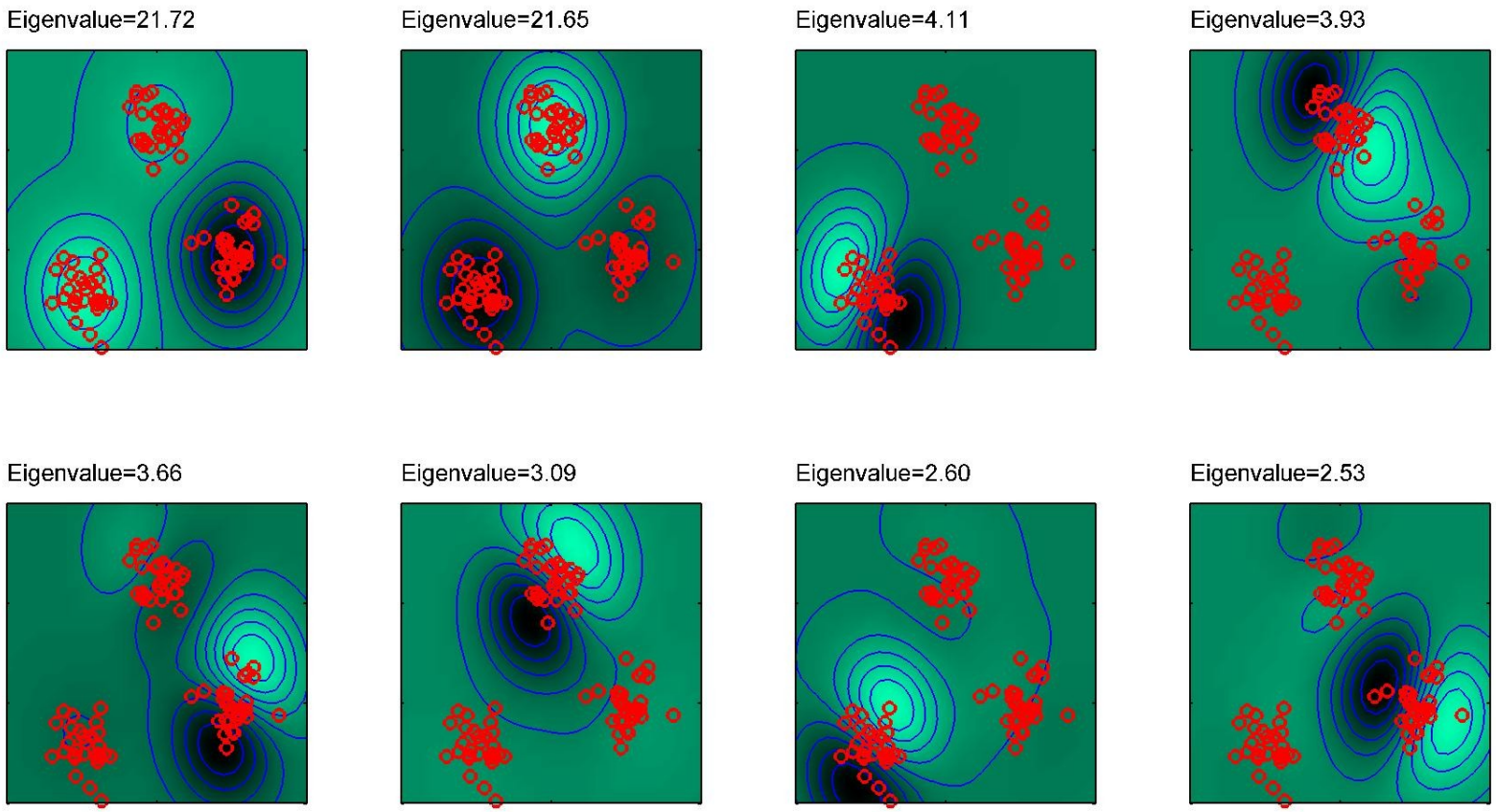
4. Normalize coefficients:

$$\mathbf{a}_i^T \mathbf{a}_i = \frac{1}{\lambda_i N}$$

5. Project data point z onto j^{th} PC:

$$y_j(z) = \sum_{n=1}^{N} a_{jn} k(z, z_n),\ n = 1, \ldots, N$$

Example 1: Finding nonlinear structure in synthetic data



Level sets indicate projection onto PC 1, 2, …, 8
(from Bishop)

Example 2: Looking inside the feature space – kernels on funky data

Graph kernels: Comparing graphs

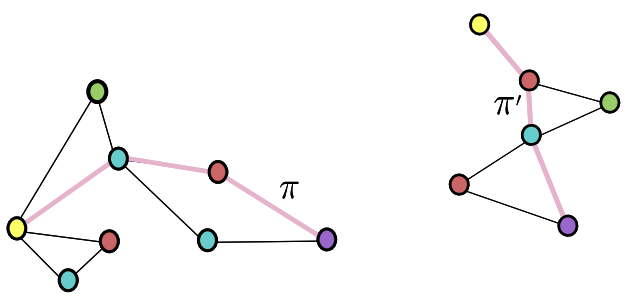
k( , ) = 42

Shortest path kernel: Decompose graphs into sets of shortest paths

$$k(G, G') = \sum_{\pi} \sum_{\pi'} k_p(\pi, \pi')$$

$*$ π and π' loop through all shortest paths in G and G'

$*$ $k_p(\pi_{vw}, \pi'_{v'w'}) = k_n(v, v') \cdot k_l(|\pi|, |\pi'|) \cdot k_n(w, w')$ kernel on paths.

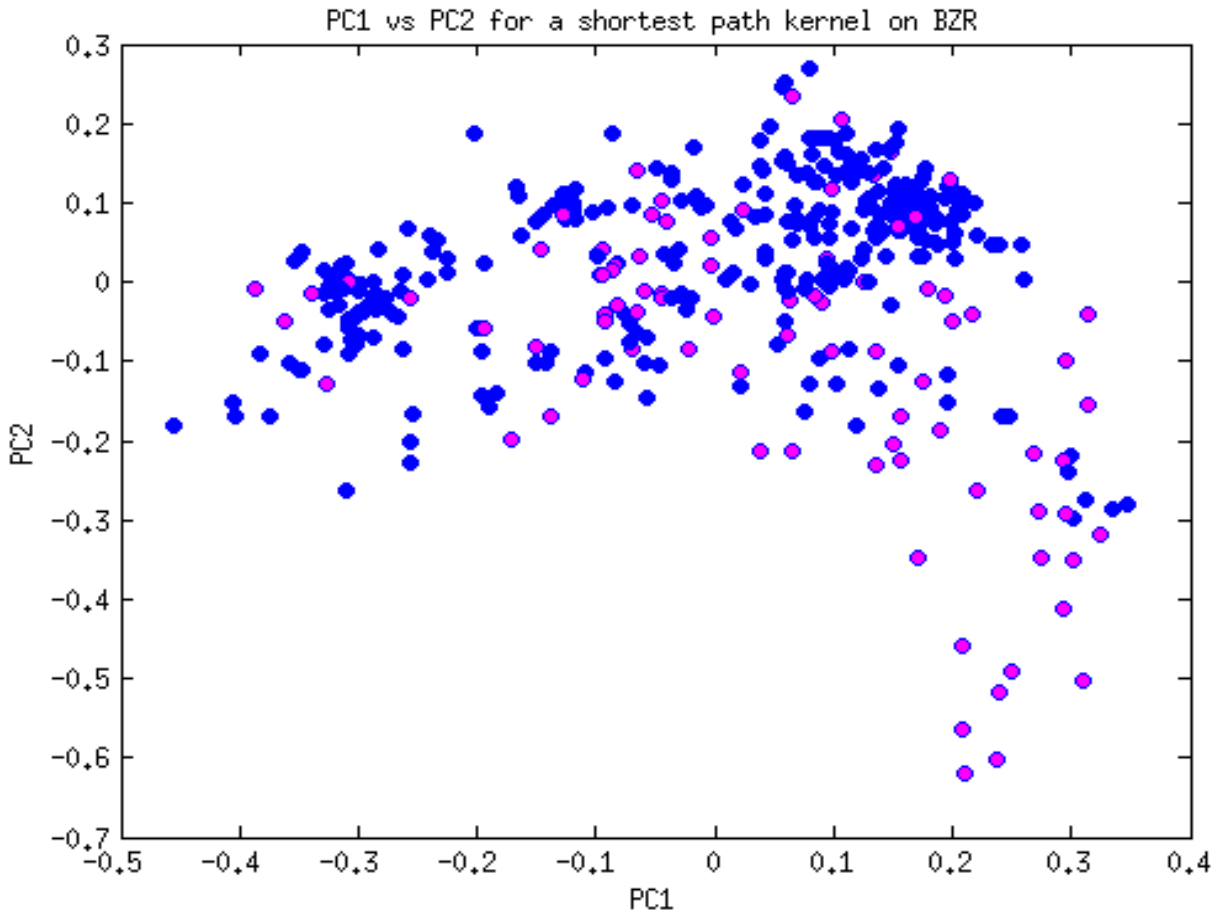


SP: $k(\pi, \pi') = k_n($ ●, ● $) \cdot k_l(3,3) \cdot k_n($ ●, ● $)$

- Dataset of 455 chemical compounds, represented as graphs, 3D node position and real-valued edge distances
- Two classes based on benzodiazepine receptor affinity
- SP-kernel (Gaussian node + path kernel) gets 83.9 +/- 0.9% classification accuracy
- Let's look at what the kernel does!

Borgwardt, Kriegel, ICDM'05

Example 2: Looking inside the feature space – kernels on funky data



From Kernel PCA to computing MDS

We saw earlier that kernels define distances

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\| = k(z_1, z_2) - 2k(z_1, z_2) + k(z_2, z_2)$$

From Kernel PCA to computing MDS

We saw earlier that kernels define distances

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\| = k(z_1, z_2) - 2k(z_1, z_2) + k(z_2, z_2)$$

We can similarly extract a kernel matrix K from a (Euclidean) distance matrix D

$$K = -\frac{1}{2}\left(D - \frac{(D1)1^T}{N} - \frac{1(D1^T)}{N} + \frac{1^T D 1}{N^2}\right)$$

where $K_{ij} = k(z_i, z_j)$ $D_{ij} = \|\Phi(z_i) - \Phi(z_j)\|$

From Kernel PCA to computing MDS

We saw earlier that kernels define distances

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\| = k(z_1, z_2) - 2k(z_1, z_2) + k(z_2, z_2)$$

We can similarly extract a kernel matrix K from a (Euclidean) distance matrix D

$$K = -\frac{1}{2}\left(D - \frac{(D1)1^T}{N} - \frac{1(D1^T)}{N} + \frac{1^T D 1}{N^2}\right)$$

where $K_{ij} = k(z_i, z_j)$ $D_{ij} = \|\Phi(z_i) - \Phi(z_j)\|$

This allows us to compute MDS directly from a distance matrix D!

1. Compute distance matrix D
2. Compute corresponding kernel matrix K
3. Perform kernel PCA to obtain MDS

MDS for non-Euclidean distances

We saw earlier that kernels define distances

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\| = k(z_1, z_2) - 2k(z_1, z_2) + k(z_2, z_2)$$

We can similarly extract a kernel matrix K from a (Euclidean) distance matrix D

$$K = -\frac{1}{2}\left(D - \frac{(D1)1^T}{N} - \frac{1(D1^T)}{N} + \frac{1^T D 1}{N^2}\right)$$

where $K_{ij} = k(z_i, z_j)$ $D_{ij} = \|\Phi(z_i) - \Phi(z_j)\|$

This allows us to compute MDS directly from a distance matrix D!

1. Compute distance matrix D
2. Compute corresponding kernel matrix K
3. Perform kernel PCA to obtain MDS

What happens if D is not a Euclidean distance matrix?

MDS for non-Euclidean distances

We saw earlier that kernels define distances

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\| = k(z_1, z_2) - 2k(z_1, z_2) + k(z_2, z_2)$$

We can similarly extract a kernel matrix K from a (Euclidean) distance matrix D

$$K = -\frac{1}{2}\left(D - \frac{(D1)1^T}{N} - \frac{1(D1^T)}{N} + \frac{1^T D1}{N^2}\right)$$

where $K_{ij} = k(z_i, z_j)$ $D_{ij} = \|\Phi(z_i) - \Phi(z_j)\|$

This allows us to compute MDS directly from a distance matrix D!

1. Compute distance matrix D
2. Compute corresponding kernel matrix K
3. Perform kernel PCA to obtain MDS

What happens if D is not a Euclidean distance matrix?

Kernel matrix K is not positive (semi)definite (negative eigenvalues)

MDS for non-Euclidean distances

We saw earlier that kernels define distances

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\| = k(z_1, z_2) - 2k(z_1, z_2) + k(z_2, z_2)$$

We can similarly extract a kernel matrix K from a (Euclidean) distance matrix D

$$K = -\frac{1}{2}\left(D - \frac{(D1)1^T}{N} - \frac{1(D1^T)}{N} + \frac{1^T D 1}{N^2}\right)$$

where $K_{ij} = k(z_i, z_j)$ $D_{ij} = \|\Phi(z_i) - \Phi(z_j)\|$

This allows us to compute MDS directly from a distance matrix D!

1. Compute distance matrix D
2. Compute corresponding kernel matrix K
3. Perform kernel PCA to obtain MDS

What happens if D is not a Euclidean distance matrix?

Kernel matrix K is not positive (semi)definite (negative eigenvalues)

Common hack: Throw away the negative eigenvalues and work only with the positive ones

MDS for non-Euclidean distances

We saw earlier that kernels define distances

$$d(z_1, z_2) = \|\Phi(z_1) - \Phi(z_2)\| = k(z_1, z_2) - 2k(z_1, z_2) + k(z_2, z_2)$$

We can similarly extract a kernel matrix K from a (Euclidean) distance matrix D

$$K = -\frac{1}{2}\left(D - \frac{(D1)1^T}{N} - \frac{1(D1^T)}{N} + \frac{1^T D 1}{N^2}\right)$$

where $K_{ij} = k(z_i, z_j)$ $D_{ij} = \|\Phi(z_i) - \Phi(z_j)\|$

This allows us to compute MDS directly from a distance matrix D!

1. Compute distance matrix D
2. Compute corresponding kernel matrix K
3. Perform kernel PCA to obtain MDS

What happens if D is not a Euclidean distance matrix?

Kernel matrix K is not positive (semi)definite (negative eigenvalues)

Common hack: Throw away the negative eigenvalues and work only with the positive ones

This is a hack and only God knows what it does! (although it seems to work ok)

Kernel PCA summary

We started out with assumptions:

- zero mean of $\Phi(z_n)$
- Finite-dimensional feature space $\mathcal{H}$.

The latter is OK because all our analysis takes place in $V = \text{span}_{n=1,\ldots,N}\{\Phi(z_n)\} \subset \mathcal{H}$, where $\dim(V) = M \leq N < \infty$.

Good about kernel PCA:

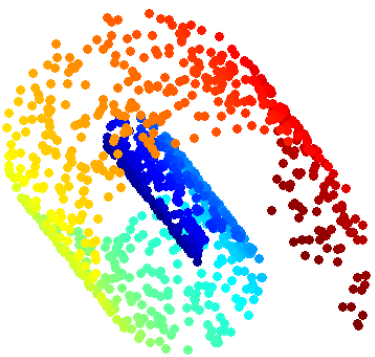
- Find nonlinear structures
- Insight into kernel – look inside $\mathcal{H}$!
- Extract approximate feature map

Bad about kernel PCA

- Runtime $\mathcal{O}(N)$ instead of $\mathcal{O}(D)$

Manifold learning: Isomap

- Isomap assumes that data can be "folded out" onto a low-dimensional space (data on a "manifold")
- Swiss roll: 2D structure in 3D
- Three steps:
  - Create neighborhood graph
  - Compute shortest path distances in path (approximates manifold distances)
  - Plug shortest path distances into MDS to "fold out" the manifold.

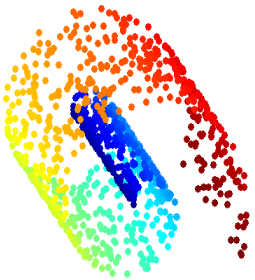


Tenenbaum et al, Science, 2000

From data to neighborhood graph

Input: A set of data points $\mathbf{x}_n \subset \mathbb{R}^d$

Assumption: A locally Euclidean structure (a flat manifold)

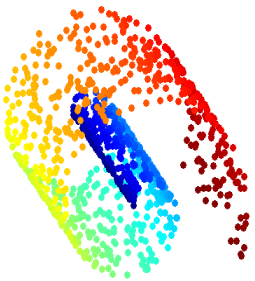


From data to neighborhood graph

Input: A set of data points $\mathbf{x}_n \subset \mathbb{R}^d$

Assumption: A locally Euclidean structure (a flat manifold)

If assumption holds: Distances between nearby points are good, long distances are bad



From data to neighborhood graph

Input: A set of data points $\mathbf{x}_n \subset \mathbb{R}^d$

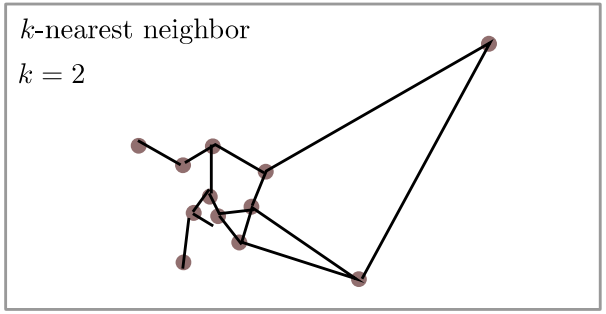
Assumption: A locally Euclidean structure (a flat manifold)

If assumption holds: Distances between nearby points are good, long distances are bad

Solution: Build neighborhood graph to approximate the manifold

Use shortest paths on graphs to approximate manifold shortest paths

From data to neighborhood graph



Input: A set of data points $\mathbf{x}_n \subset \mathbb{R}^d$

Assumption: A locally Euclidean structure (a flat manifold)

If assumption holds: Distances between nearby points are good, long distances are bad

Solution: Build neighborhood graph to approximate the manifold

Use shortest paths on graphs to approximate manifold shortest paths

Two approaches to neighborhood graph:

k-nearest neighbor graph

From data to neighborhood graph

Input: A set of data points $\mathbf{x}_n \subset \mathbb{R}^d$

Assumption: A locally Euclidean structure (a flat manifold)

If assumption holds: Distances between nearby points are good, long distances are bad

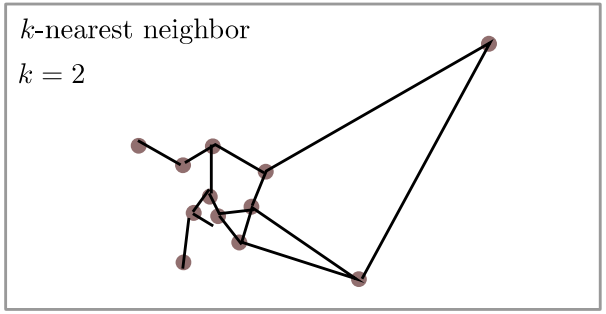
Solution: Build neighborhood graph to approximate the manifold

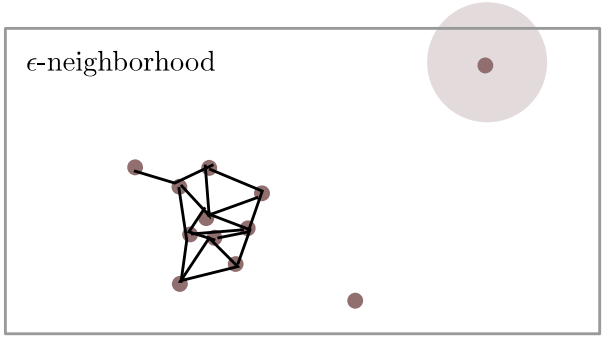
Use shortest paths on graphs to approximate manifold shortest paths

Two approaches to neighborhood graph:

k-nearest neighbor graph

ϵ-neighborhood graph





From data to neighborhood graph

Input: A set of data points $\mathbf{x}_n \subset \mathbb{R}^d$

Assumption: A locally Euclidean structure (a flat manifold)

If assumption holds: Distances between nearby points are good, long distances are bad

Solution: Build neighborhood graph to approximate the manifold

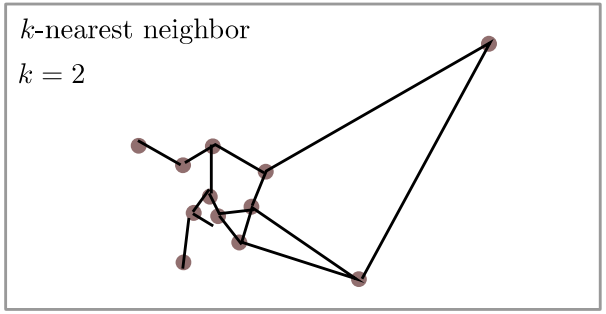
Use shortest paths on graphs to approximate manifold shortest paths

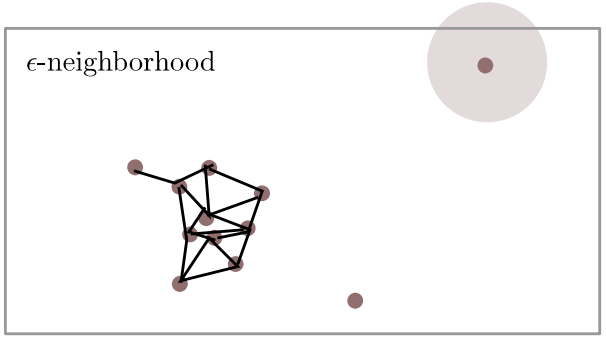
Two approaches to neighborhood graph:

k-nearest neighbor graph

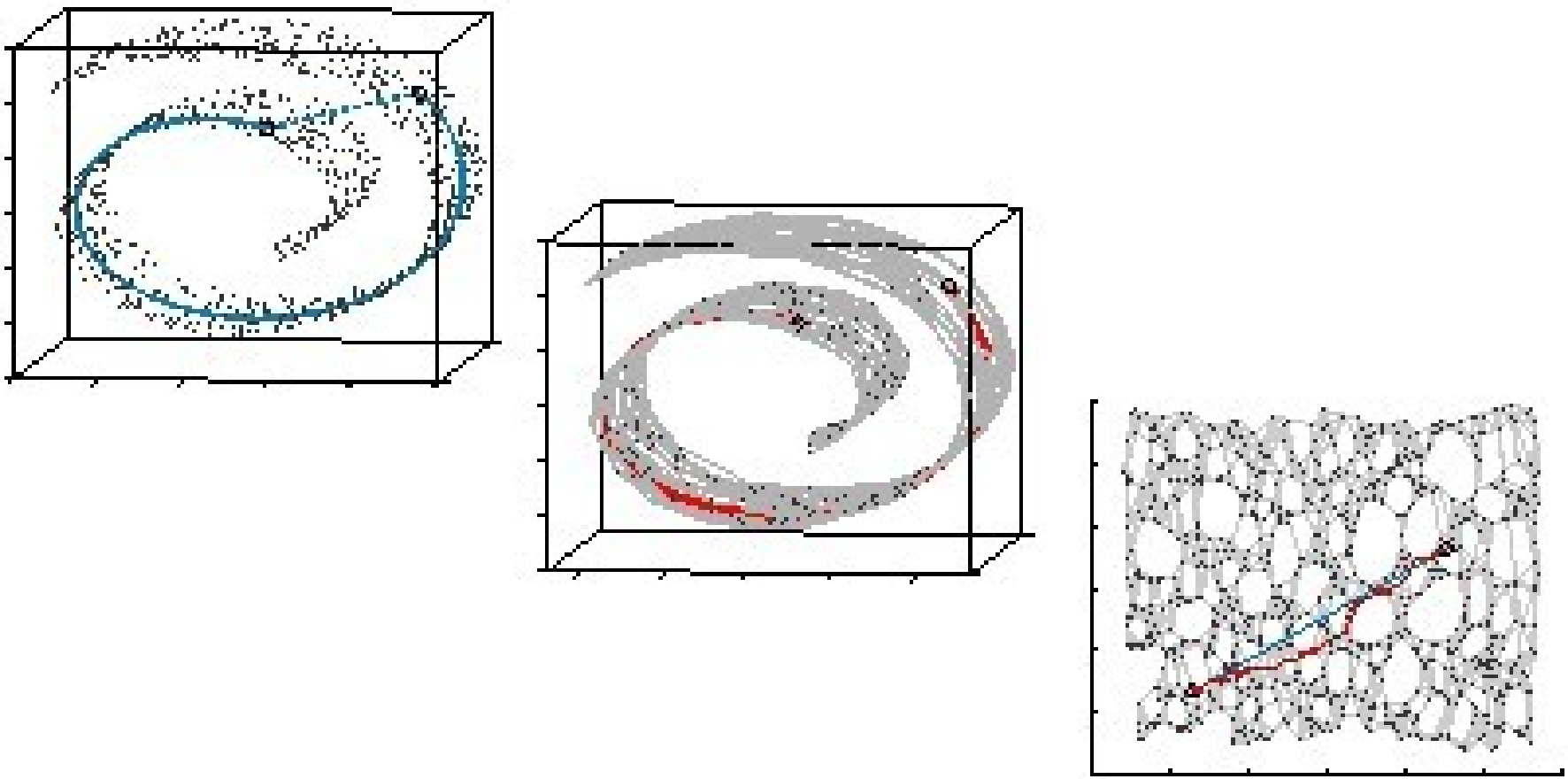
ϵ-neighborhood graph

What is right and wrong with the two approaches?



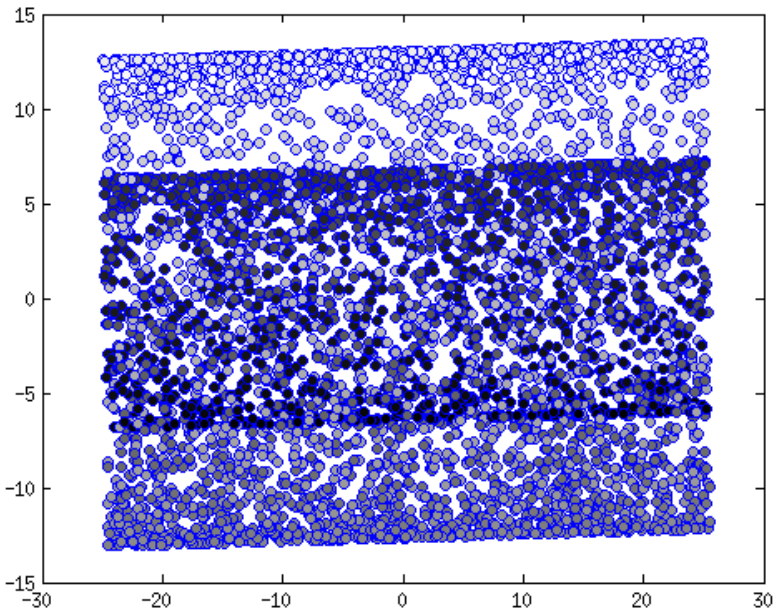


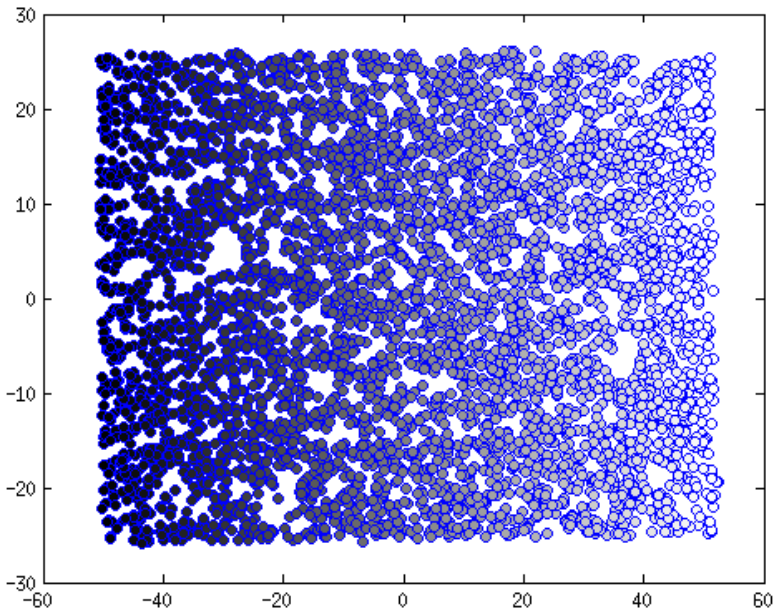
Example 1: The Swiss Roll



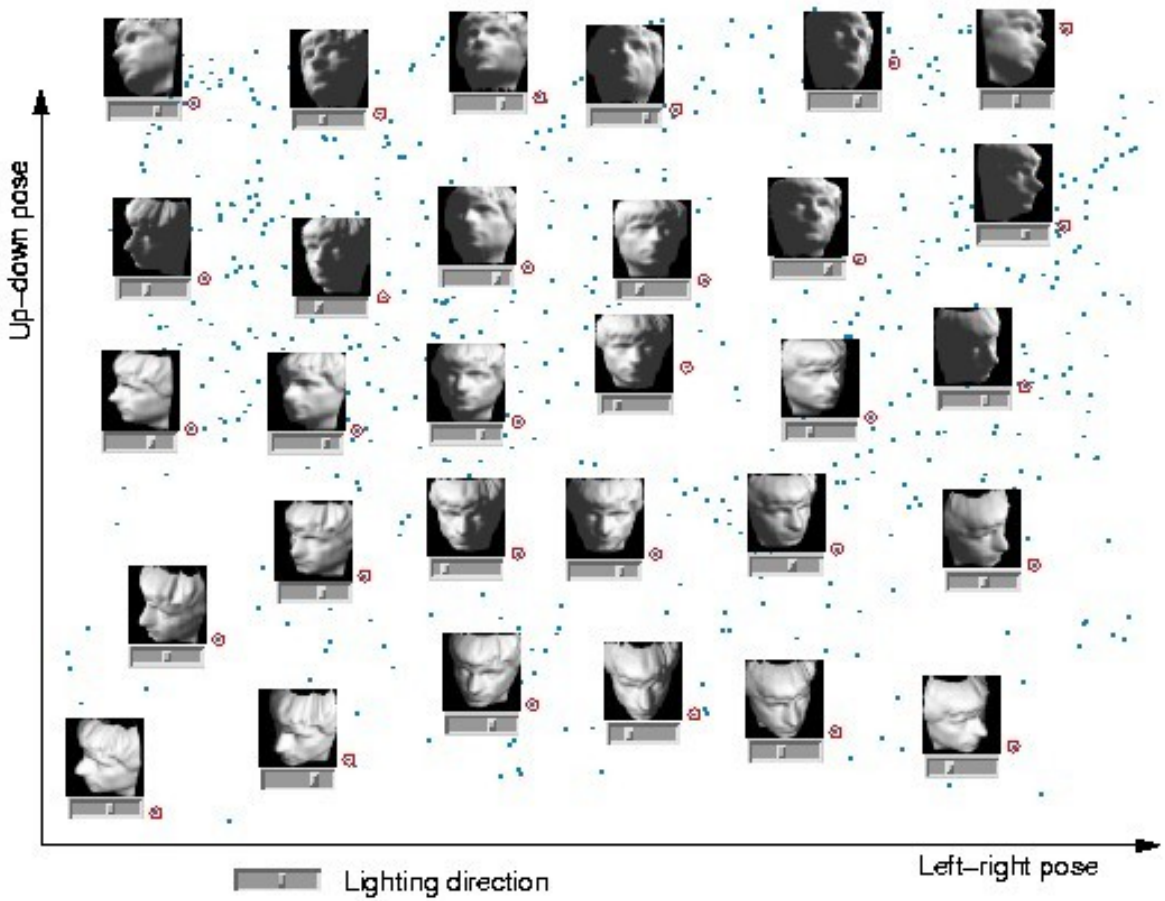
Tenenbaum et al, Science, 2000

Example 1: The swiss roll



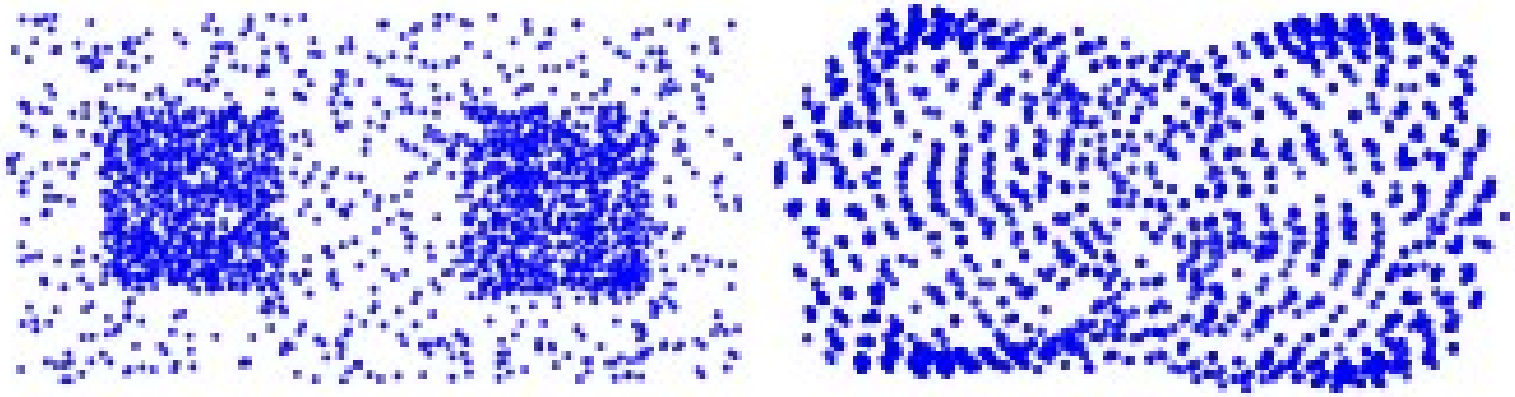


Example 2: Faces

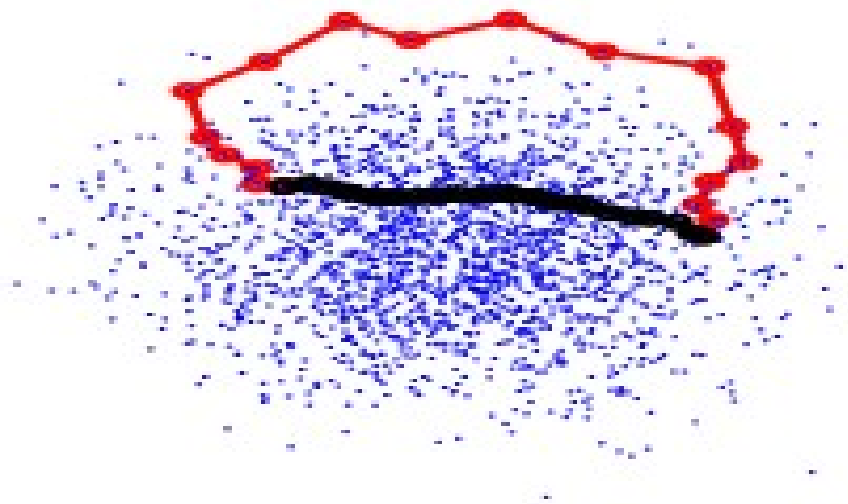


Tenenbaum et al, Science, 2000

The topological graph structure is not enough!



Isomap with unweighted kNN graph

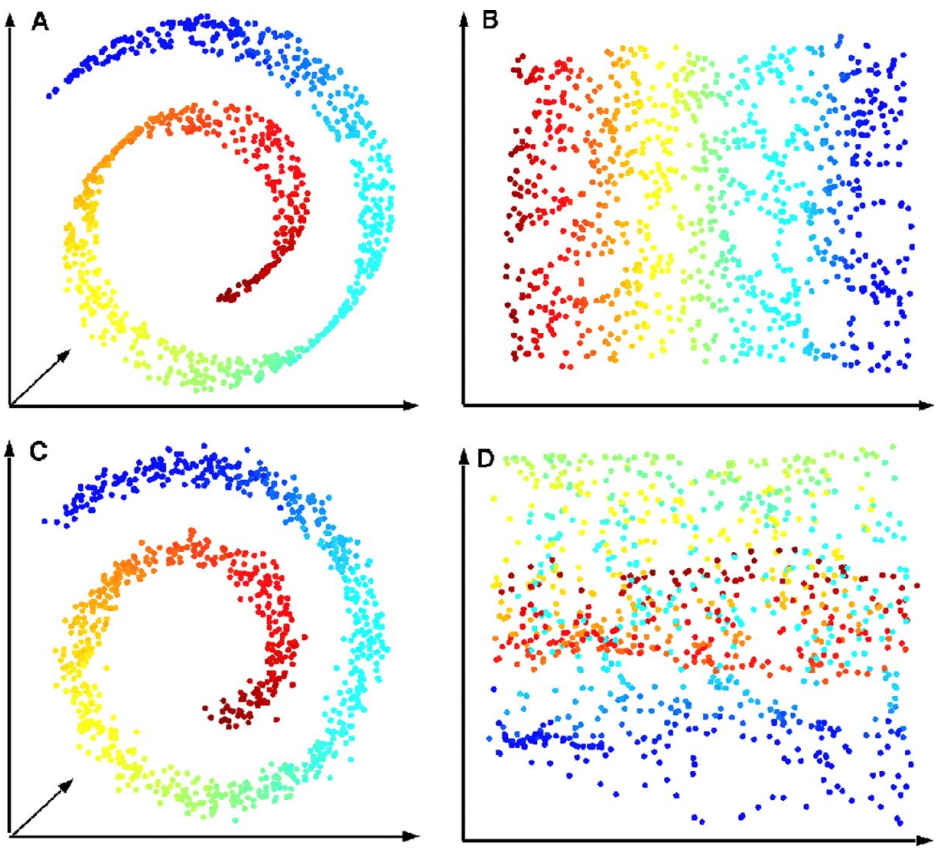


Shortest path in weighted (black) and unweighted KNN graph

Alamgir, von Luxburg, ICML'12

Isomap properties

- Easy to implement
- Easy to understand
- Often works quite well
- Problems:
  - Topological stability
  - Fails for disconnected graph
  - As MDS: Focus on preserving long distances



M Balasubramanian, and E L Schwartz Science 2002;295:7

Preserving short distances: Kruskal stress

Hillis et al, 2005

Whereas MDS optimizes a function

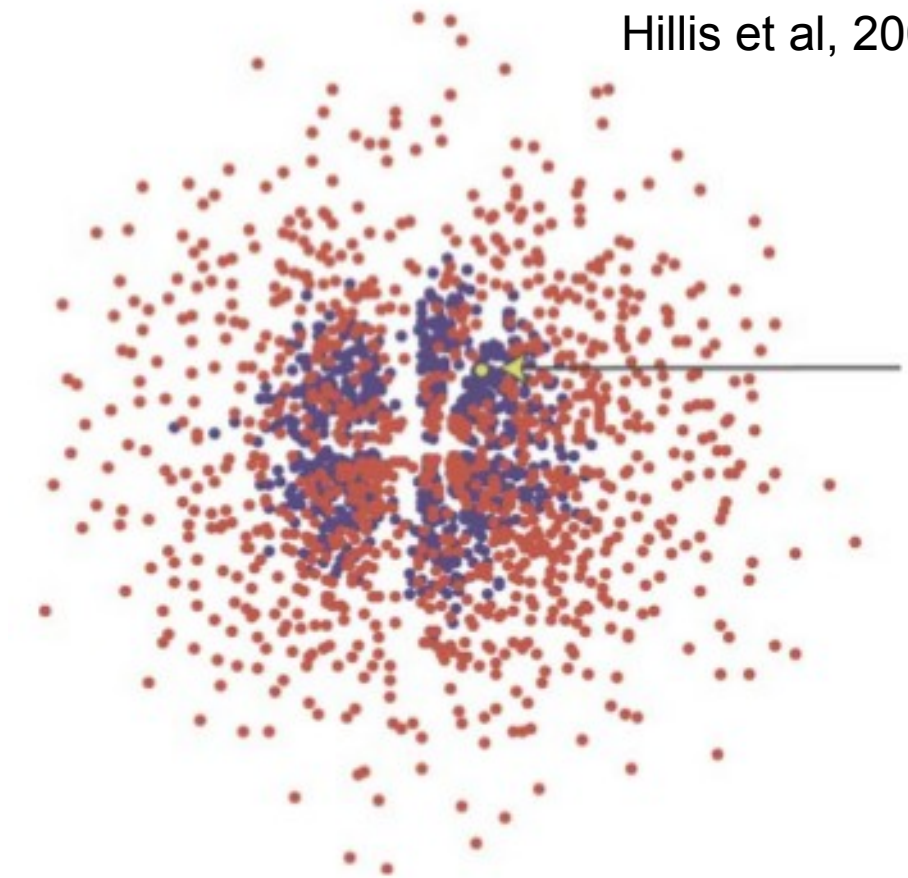
$$\sum_{ij} \left(d_{ij}^2 - \|f(z_i) - f(z_j)\|^2\right)$$

The Kruskal 1-stress optimizes

$$\left(\sum_{i \neq j} (d_{ij} - \|f(z_i) - f(z_j)\|)^2\right)^{1/2}$$

Computed using non-convex optimization

Computationally intensive,
Non-optimal solutions



Visual comparison of phylogenetic trees
Obtained by bootstrap (red) and
MCMC sampling (blue). True tree (yellow).
Distances are Robinson Foulds (unweighted
tree edit distance)

Summary: You should now:

- Be familiar with Multidimensional Scaling (MDS)
  - Classical definition
  - Low-distortion interpretation of PCA
  - MDS for non-Euclidean data
- Be familiar with basic manifold learning techniques
  - Isomap
  - What problems does Isomap (try to) solve?
  - What are its strengths and weaknesses
- Be familiar with kernel PCA:
  - Its definition and relation to standard PCA
  - How to compute it
  - Applications to nonlinear PCA and non-Euclidean data