Deep Learning Theory and Applications

Unsupervised Learning and Autoencoders

Yale

CPSC 452/552

CBB/AMTH 663

Outline

1. Unsupervised Learning
2. PCA
3. Non-linear dimensionality reduction
4. Autoencoders

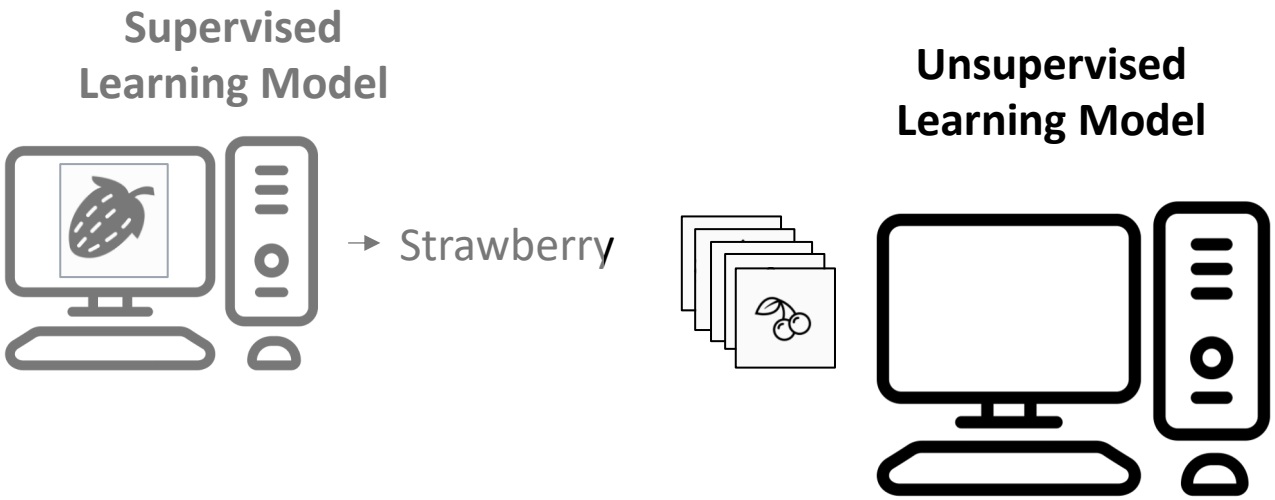
Two kinds of machine learning

Supervised learning

- Have a bunch of labelled data, want to label new data

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it



Noun project

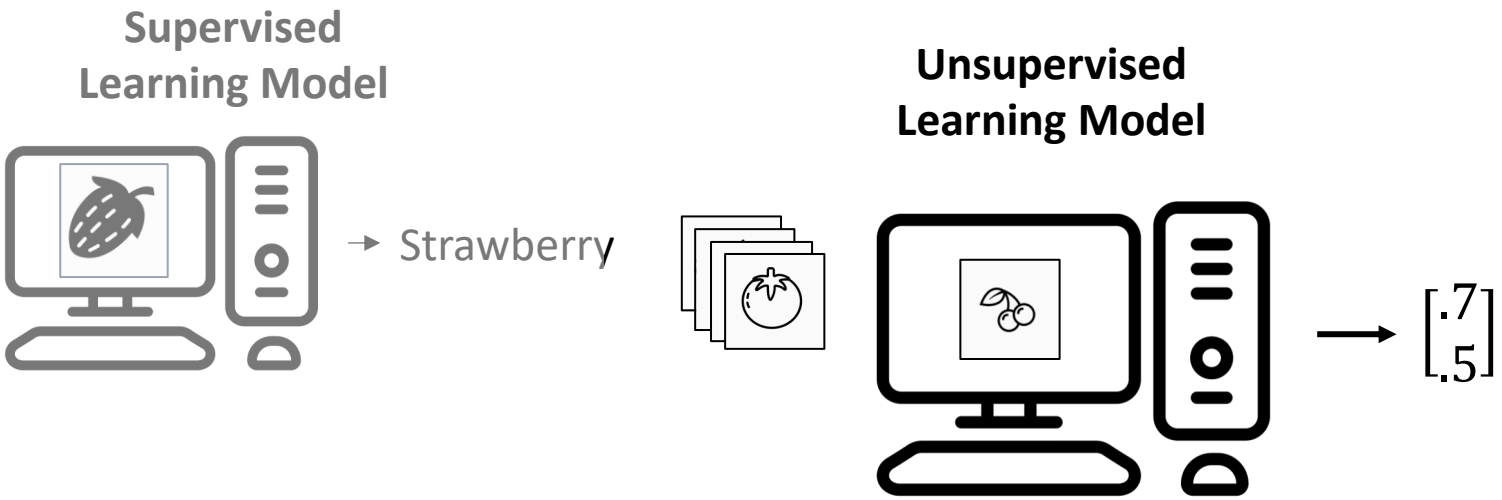
Two kinds of machine learning

Supervised learning

- Have a bunch of labelled data, want to label new data

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it



Noun project

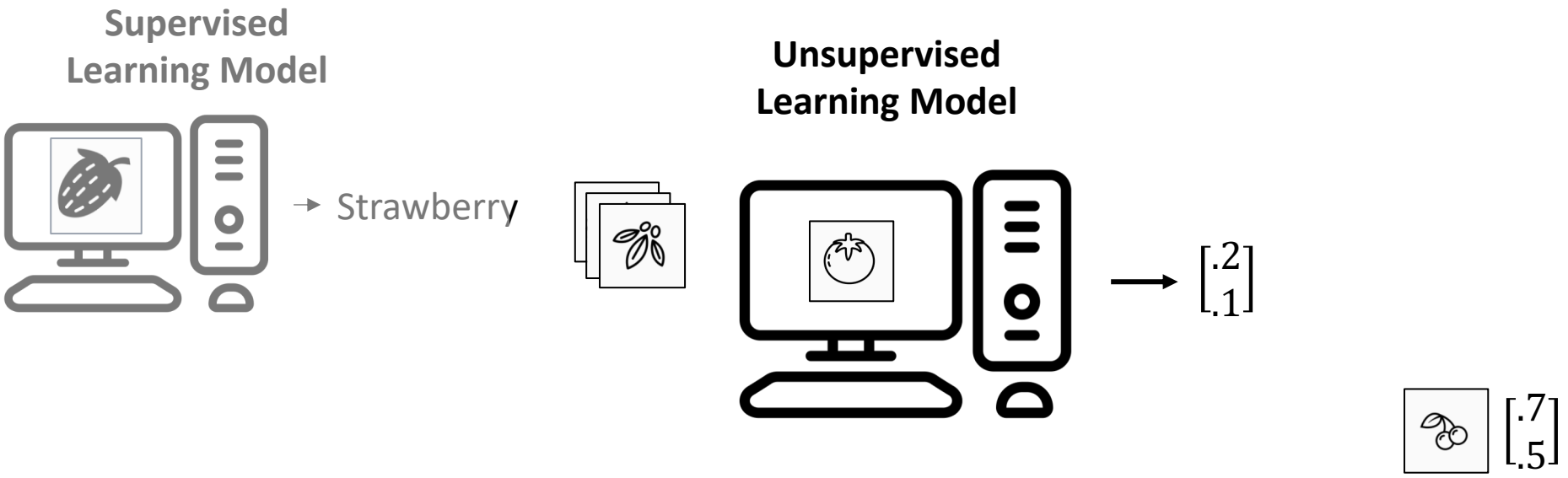
Two kinds of machine learning

Supervised learning

- Have a bunch of labelled data, want to label new data

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it



Noun project

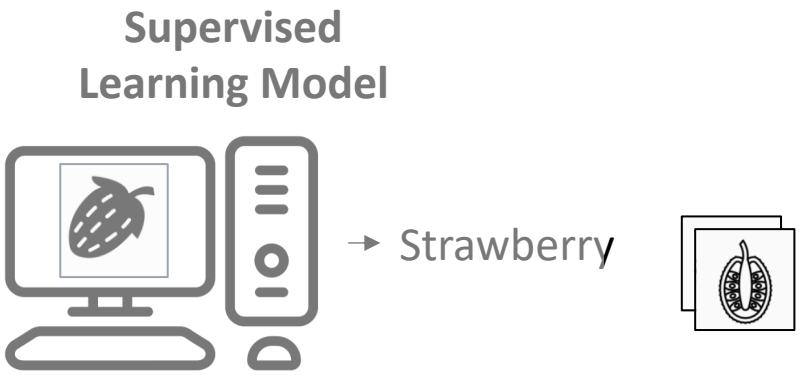
Two kinds of machine learning

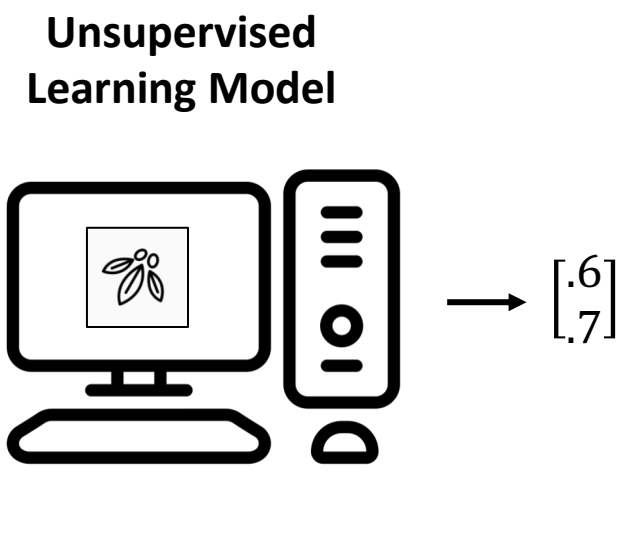
Supervised learning

- Have a bunch of labelled data, want to label new data

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it





Noun project

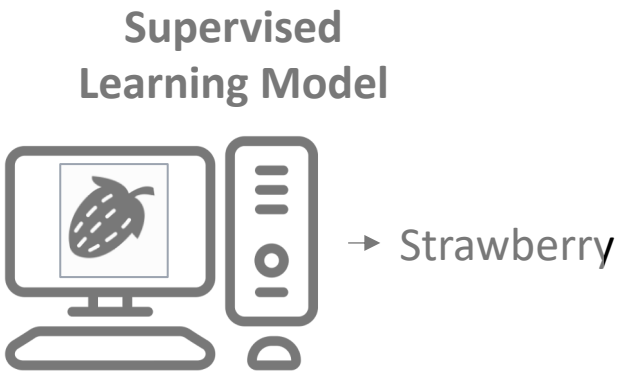
Two kinds of machine learning

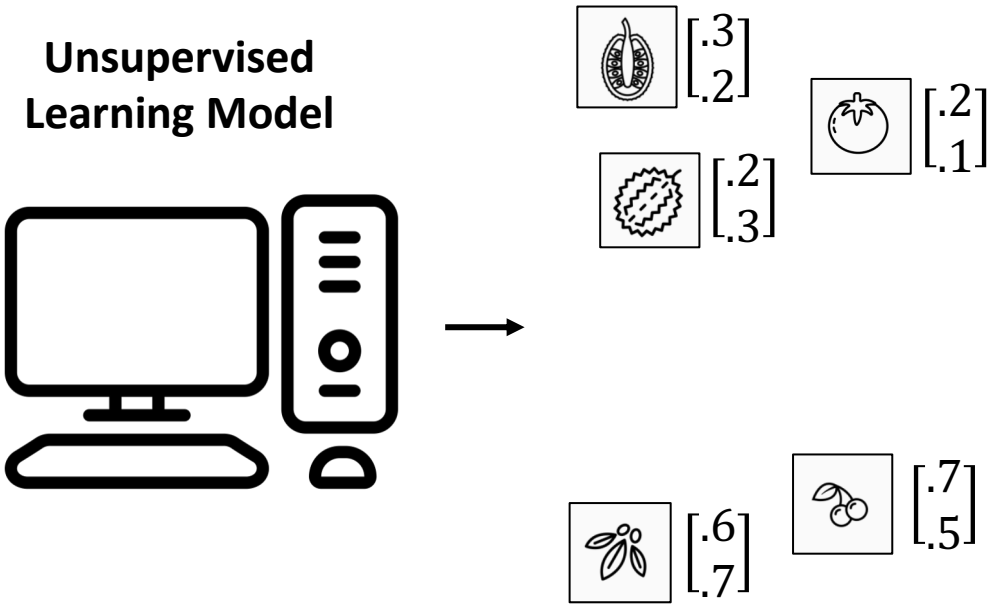
Supervised learning

- Have a bunch of labelled data, want to label new data

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it





Noun project

Two kinds of machine learning

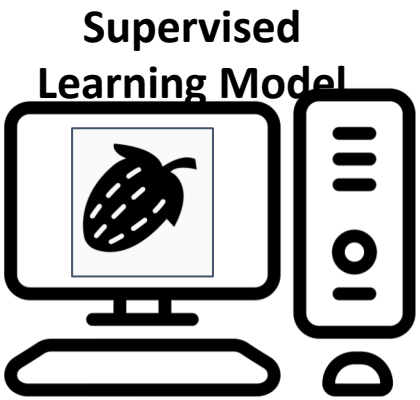
Supervised learning

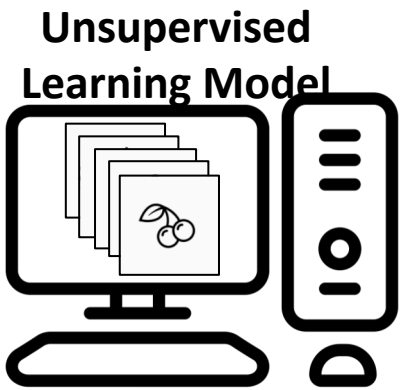
- Have a bunch of labelled data, want to label new data
- Learn a function $f(X) \rightarrow Y$ where all values of Y are known for some samples of X

Unsupervised learning

- Have a bunch of unlabeled data, want to organize it
- Learn an embedding
- Lower dimensional, easier to interpret (e.g. as clusters)

$$f(X) \rightarrow Y, X \in \mathbb{R}^n, Y \in \mathbb{R}^m, n \gg m$$





Noun project

Unsupervised learning

- Dataset contains only features and no labels
- Goal is to find hidden patterns in the unlabeled data
- Examples
  - Density estimation
  - Data generation
  - Clustering
  - Data denoising
  - Representation learning
- ***Semi-supervised learning*** attempts to combine information from both labeled and unlabeled data to deduce information

Representation learning

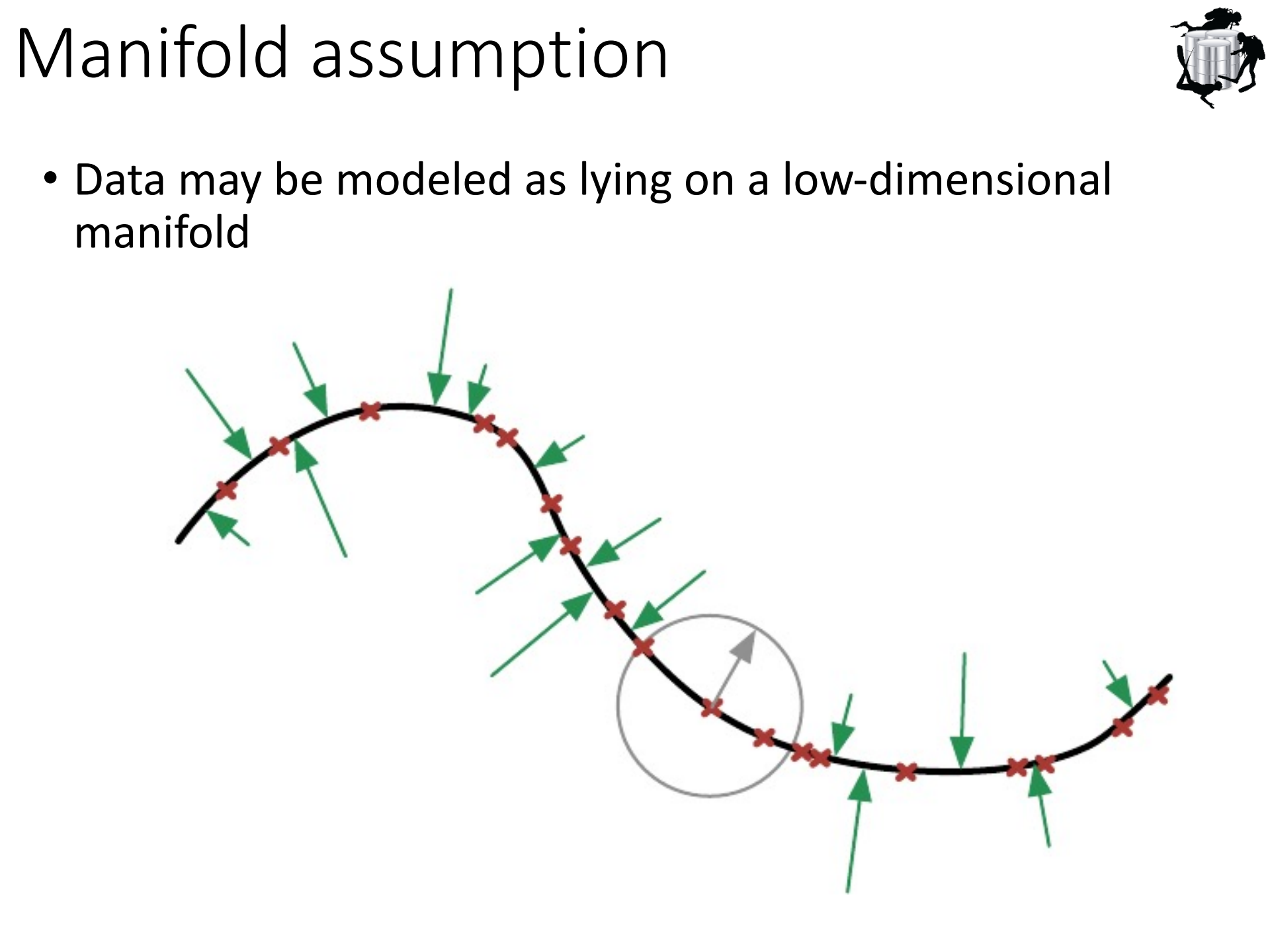
- Find the “best” representation of the data
  - I.e., find a representation of the data that preserves as much information as possible while obeying some penalty or constraint that simplifies the representation
- What information do we want to preserve?
- How do we define simple?
  - Low-dimensional
  - Sparse
  - Independent components
- Above criteria aren’t necessarily mutually exclusive
  - E.g., low-dimensional representations often have independent or weakly dependent components

Representation learning

Why is a simple representation useful?

- Interpretability
  - E.g. visualization
- Computational cost
  - Compression
- Performance
  - Preprocessing

Manifold assumption



- Data may be modeled as lying on a low-dimensional manifold

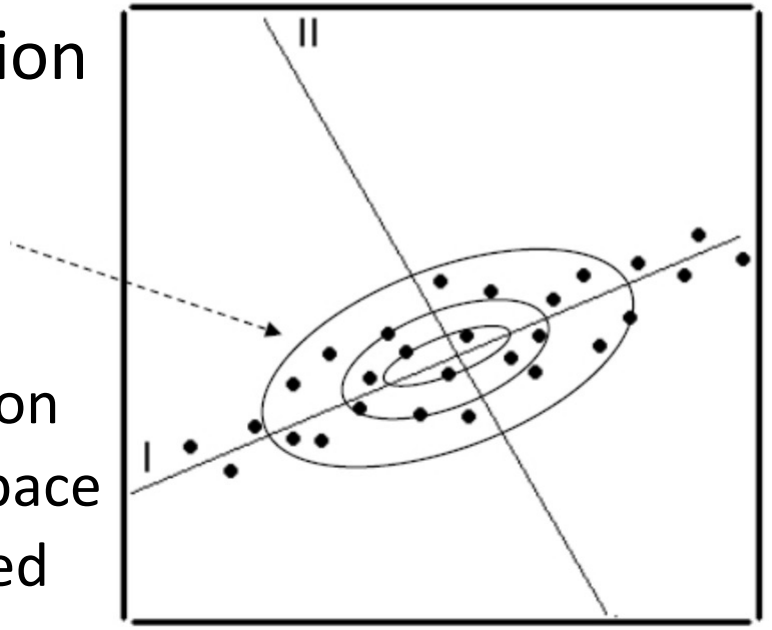
Example

- What is a good lower dimensional representation of this data?

- How can we find it?

Principal Component Analysis (PCA)

- PCA: the most popular method for dimensionality reduction
- Goal: find the directions in input space that explain the most variation
  - Data is re-represented by projecting along those directions
- Assumptions
  - The variation contains the information
  - The manifold model is a linear subspace
  - Components are linearly uncorrelated with each other



PCA: Maximize Variance

- N data vectors $\boldsymbol{x}_i$ with dimension d
  - $\overline{\boldsymbol{x}}$ the sample mean
- Goal is to reduce dimensionality to $k \ll d$
- Sample covariance matrix:

$$C = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$$

- Find direction $\boldsymbol{u}$ that maximizes the variance:

$$\arg\max_{\boldsymbol{u}} \boldsymbol{u}^T C \boldsymbol{u}$$

  - Typically, we require $\|\boldsymbol{u}\| \leq 1$
  - This is an eigendecomposition problem!
  - Choose $\boldsymbol{u}$ to be the first eigenvector of C

PCA: Maximize Variance

- N data vectors $\boldsymbol{x}_i$ with dimension d
  - $\overline{\boldsymbol{x}}$ the sample mean
- Goal is to reduce dimensionality to $k \ll d$
- Sample covariance matrix:

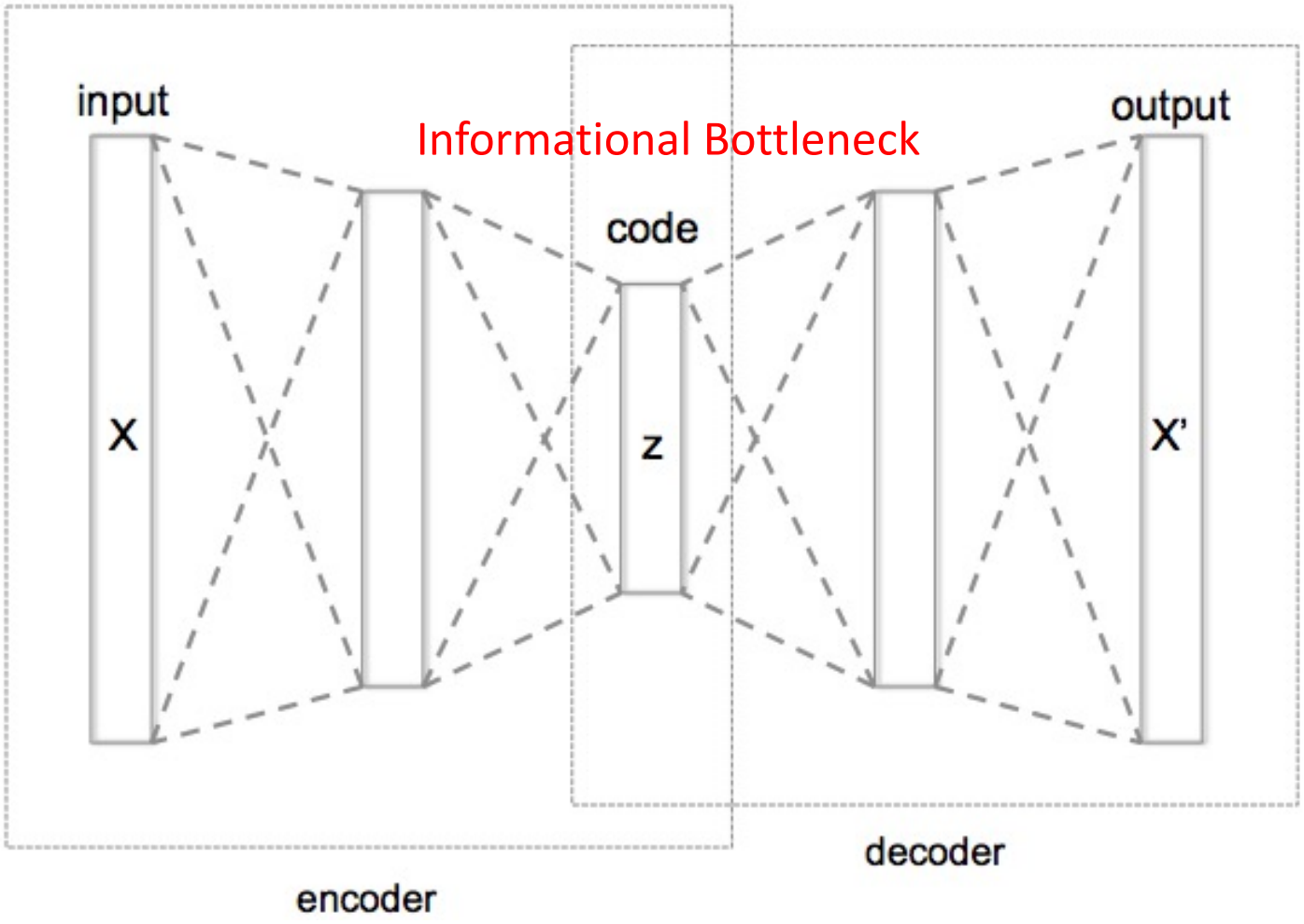
$$C = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T = \mathrm{U}\Lambda U^T$$

- U_k contains the first k eigenvectors
  - These give the directions of most explained variance
- Project the data points on this space: $\boldsymbol{z}_i = U_k^T \boldsymbol{x}_i$
  - Reconstruct with $\widehat{\boldsymbol{x}}_i = U_k U_k^T \boldsymbol{x}_i$

Autoencoder (AE)

- Neural network trained to copy its input x to its output
- Hidden layer z describes a ***code*** to represent the input
- Two parts
  - Encoder: $\mathrm{z} = f(x)$
  - Decoder: $x' = g(z)$
- Goal: minimize $L\left(x, g\left(f(x)\right)\right)$
  - L penalizes $g\left(f(x)\right)$ for being dissimilar from x
  - Example: mean squared error
- How do you train an autoencoder?
  - Backpropagation

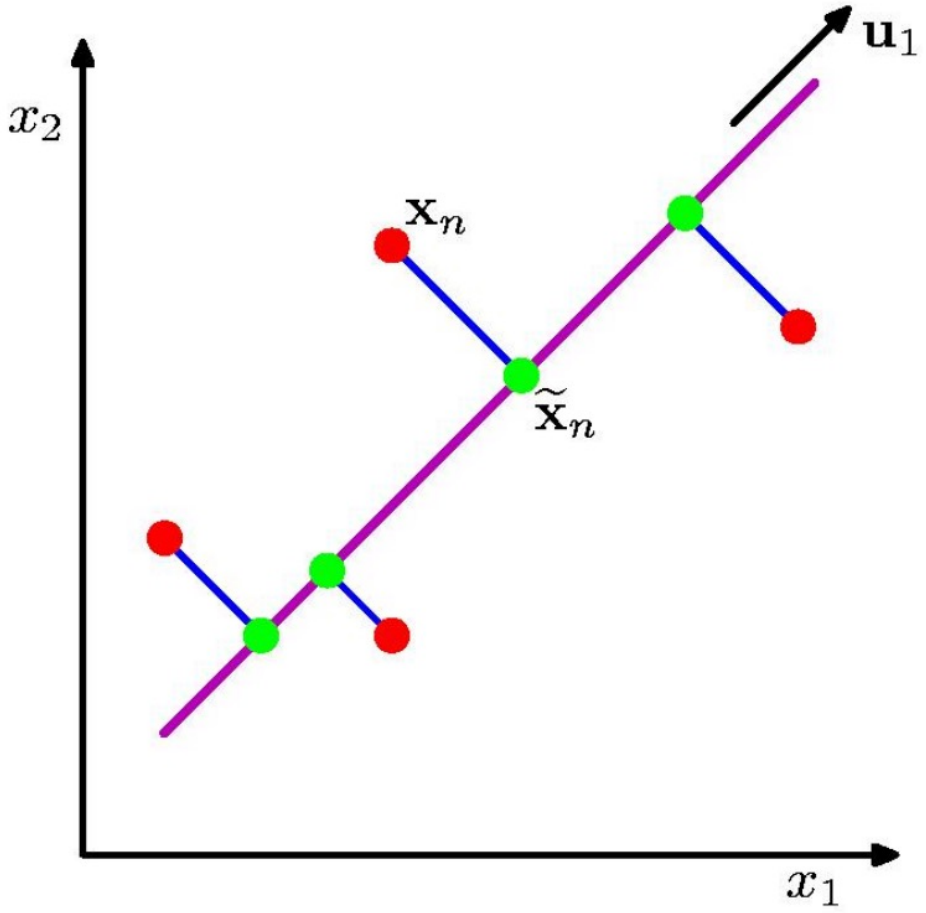
Autoencoder



[Hinton, Salakhutdinov, Science 2006]

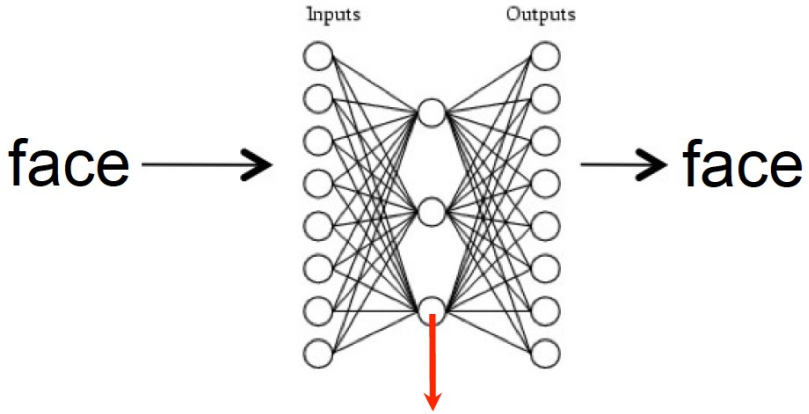
PCA: Two Views

1. Maximize variance (scatter of green points)
2. Minimize error (red-green distance per data point)



PCA as Minimizing Reconstruction Error

$$\text{face}_i = \sum_k c_{ik}\ \text{eigenface}_k$$



"eigenfaces"

Reconstruction

- Assume data come from d-dimensional vectors where nth vector is $x=[x_1,x_2 \ldots x_d]$
- We can represent a point in d-dimensional Euclidean space in terms of any d orthogonal vectors , call them $U=\{u_1,u_2 \ldots u_d\}$
- The Goal is:
- For m<d find the vectors U such that $E=\sum_{i=1}^{n} ||x - \tilde{x}||^2$ is minimized
- Here $\tilde{x} = \bar{x} - \sum_{i=1}^{m} z_i u_i$ (mean centered)
- 0 reconstruction error if m=d
- Otherwise error all due to missing $u_{m+1} \ldots u_d$ terms

Reconstruction

- $E=\sum_{i=M+1}^{d}\sum_{j=1}^{N} u_i'(x-\bar{x})^2$
- $E=\sum_{i=M+1}^{d}\sum_{j=1}^{N}(u_i'(x-\bar{x}))('(x-\bar{x})'\,u_i)$
- $E=\sum_{i=M+1}^{d} u_i'\sum u_i$

Covariance matrix

- How do you minimize this value? $\sum u_i$
- Pick the the smallest eigenvectors you can here (i.e. get rid of the low PCA components).

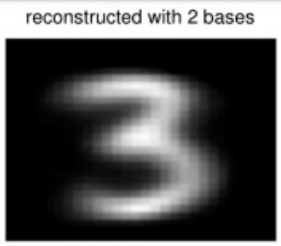
PCA on facial images

- PCA applied to 2429 19×19 images from CBCL dataset
- Reconstruction with only 3 components:

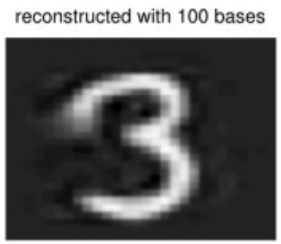
PCA on facial images

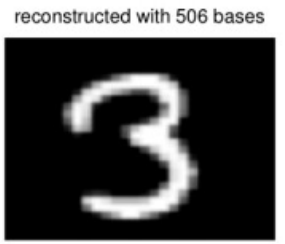
The principal components

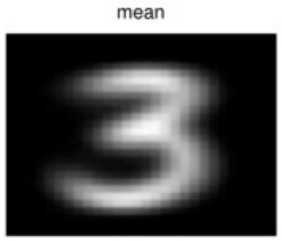
PCA on MNIST

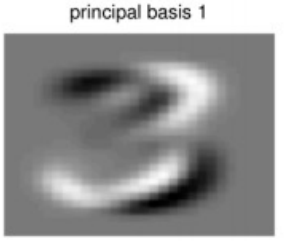

reconstructed with 2 bases


reconstructed with 10 bases

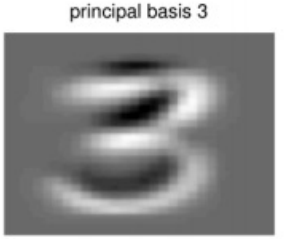

reconstructed with 100 bases


reconstructed with 506 bases


mean


principal basis 1

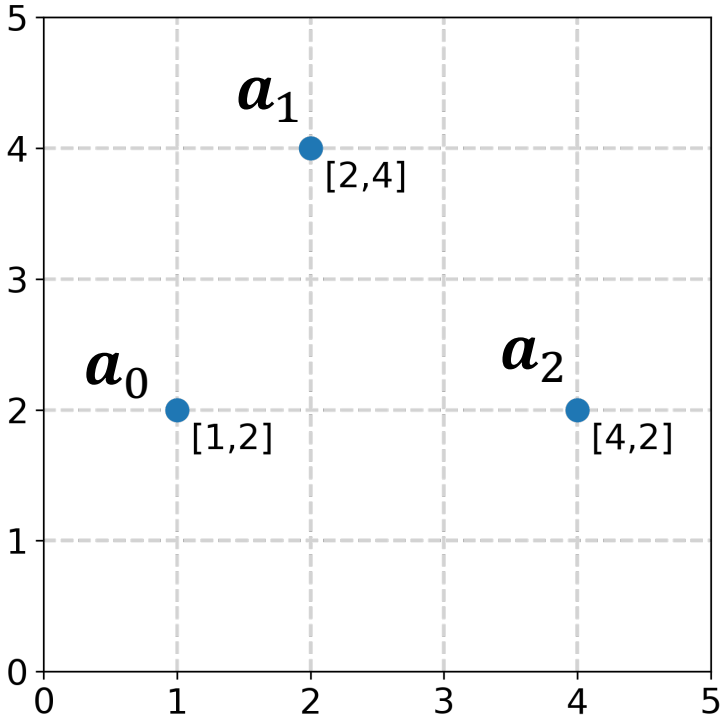

principal basis 2


principal basis 3

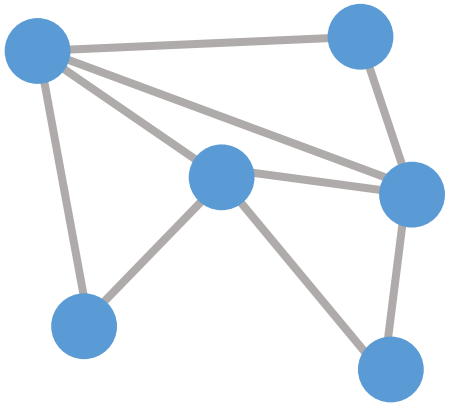
Non-linear dimensionality reduction

Representing Data as a Graph

Plot



Graph

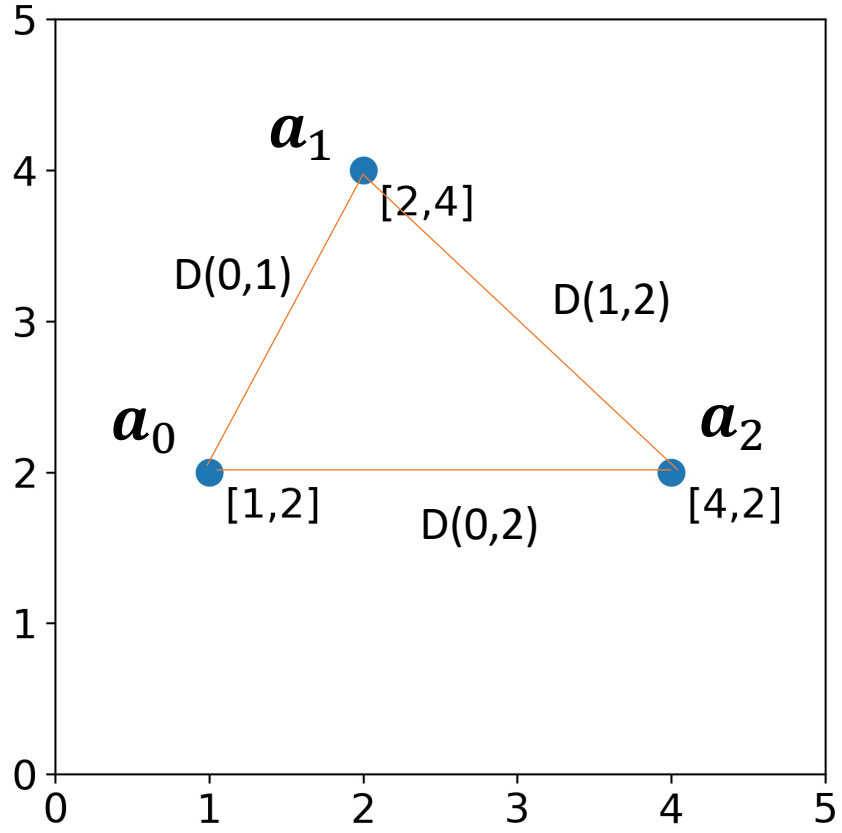


$G = (V, E)$

= V = vertices / nodes

= E = edges

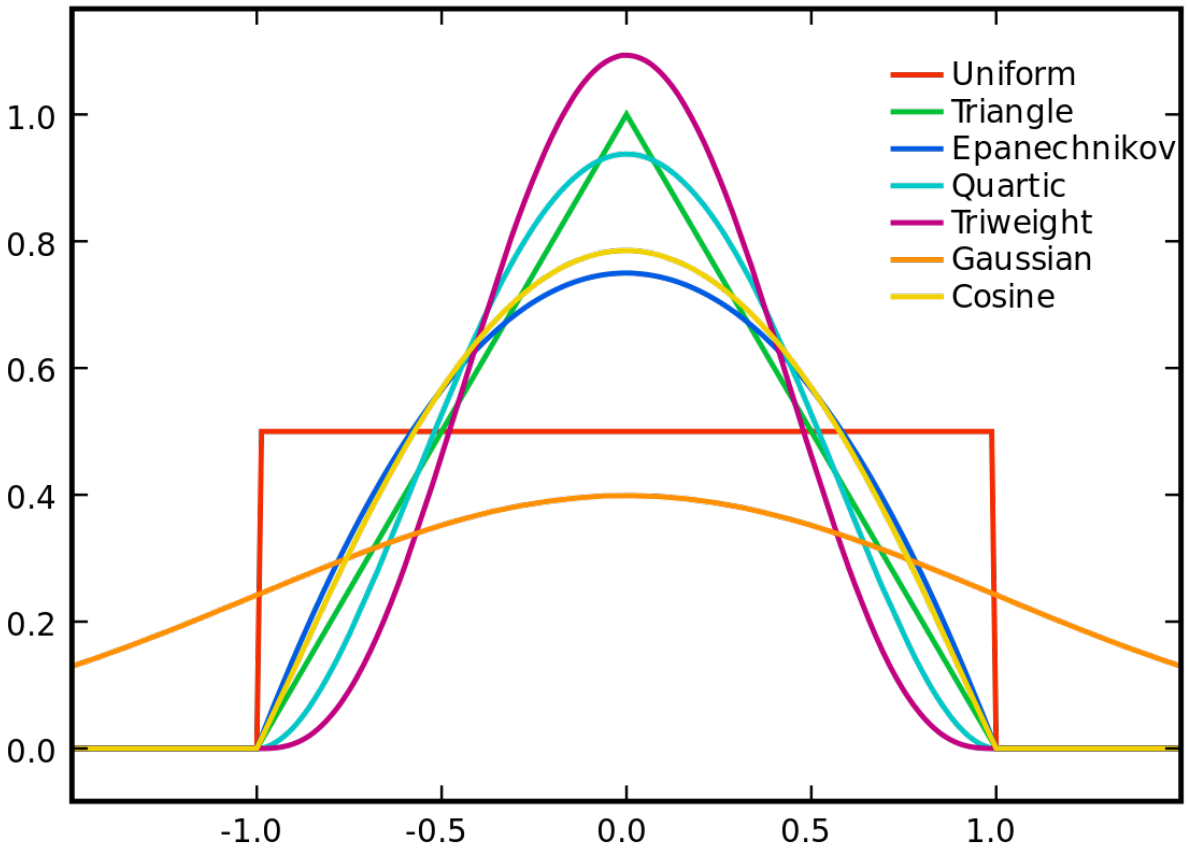
Fully connected Graph



	a_0	a_1	a_2
a_0	0	$\sqrt{5}$	3
a_1	$\sqrt{5}$	0	$\sqrt{8}$
a_2	3	$\sqrt{8}$	0

Distance matrix

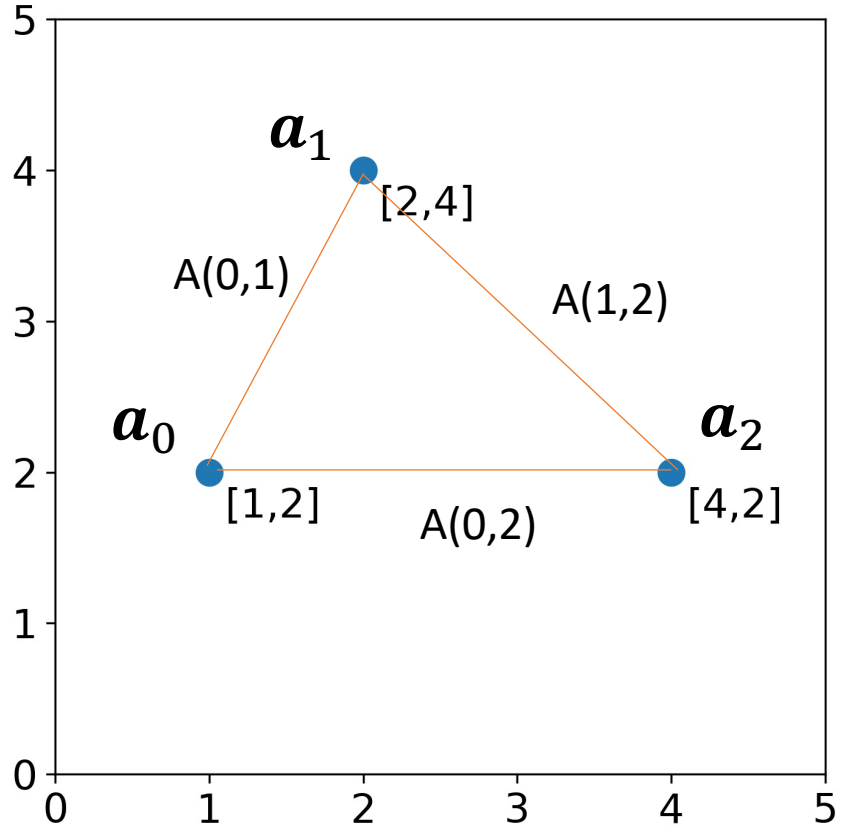
Distance to Affinity via Kernels



Real-valued
Non-negative
Symmetric

Affinities correlations in this hidden hypothetical space

Affinities

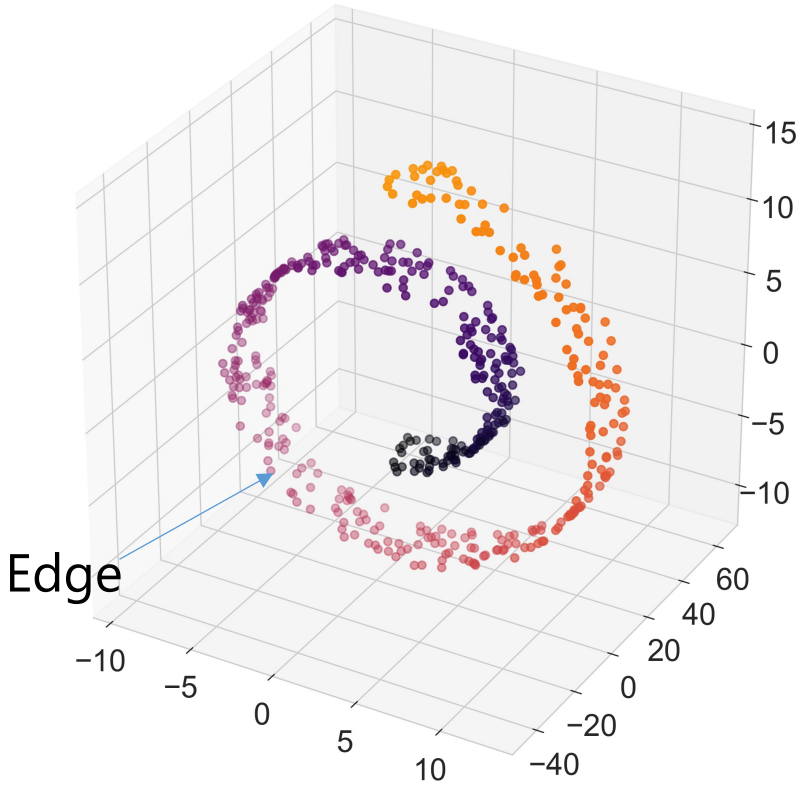


	a_0	a_1	a_2
a_0	1	0.032	0.0044
a_1	0.032	1	0.0075
a_2	0.0044	0.0075	1

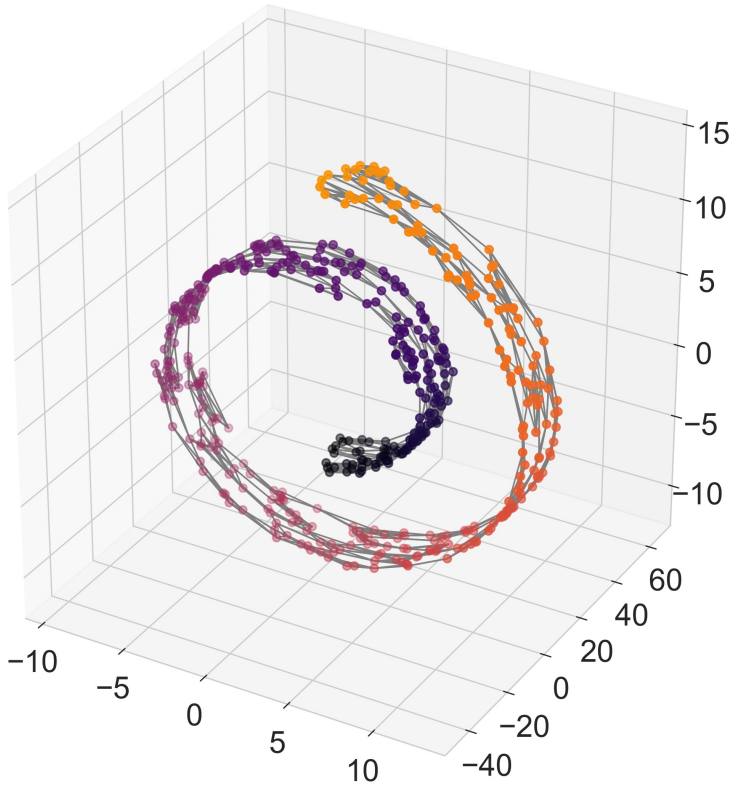
Affinity or Adjacency Matrix

Nearest Neighbors Graph

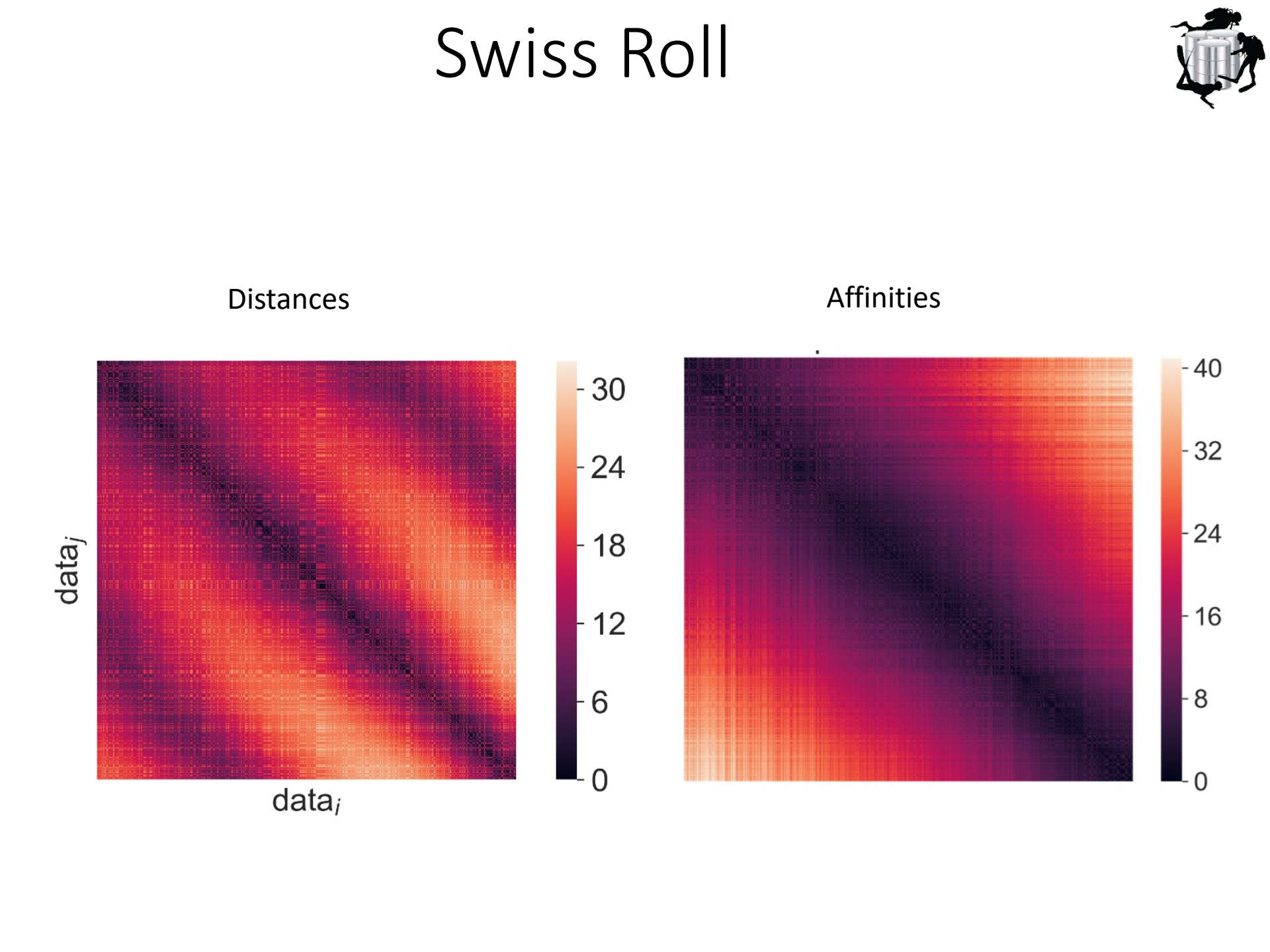
Data



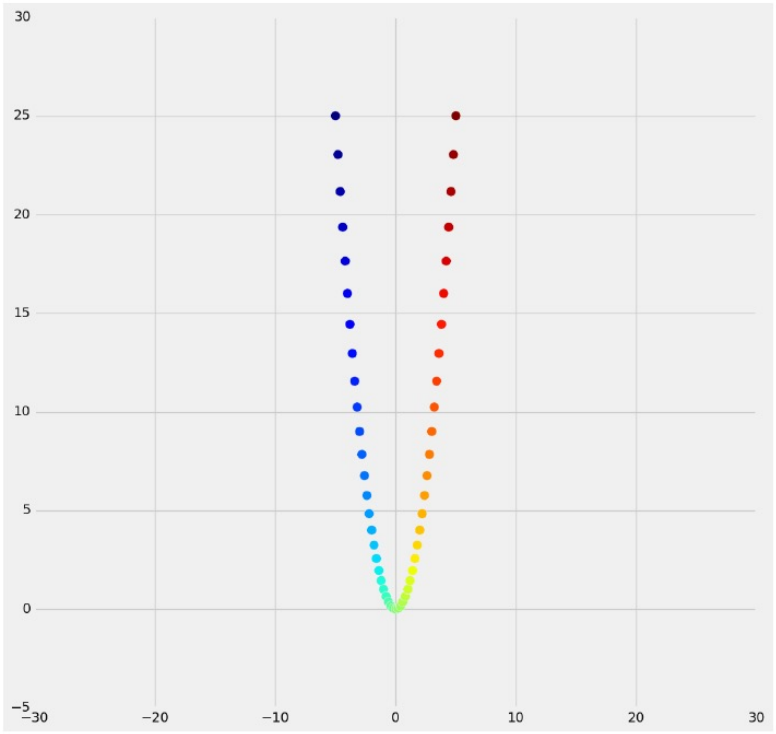
Nearest Neighbor Graph

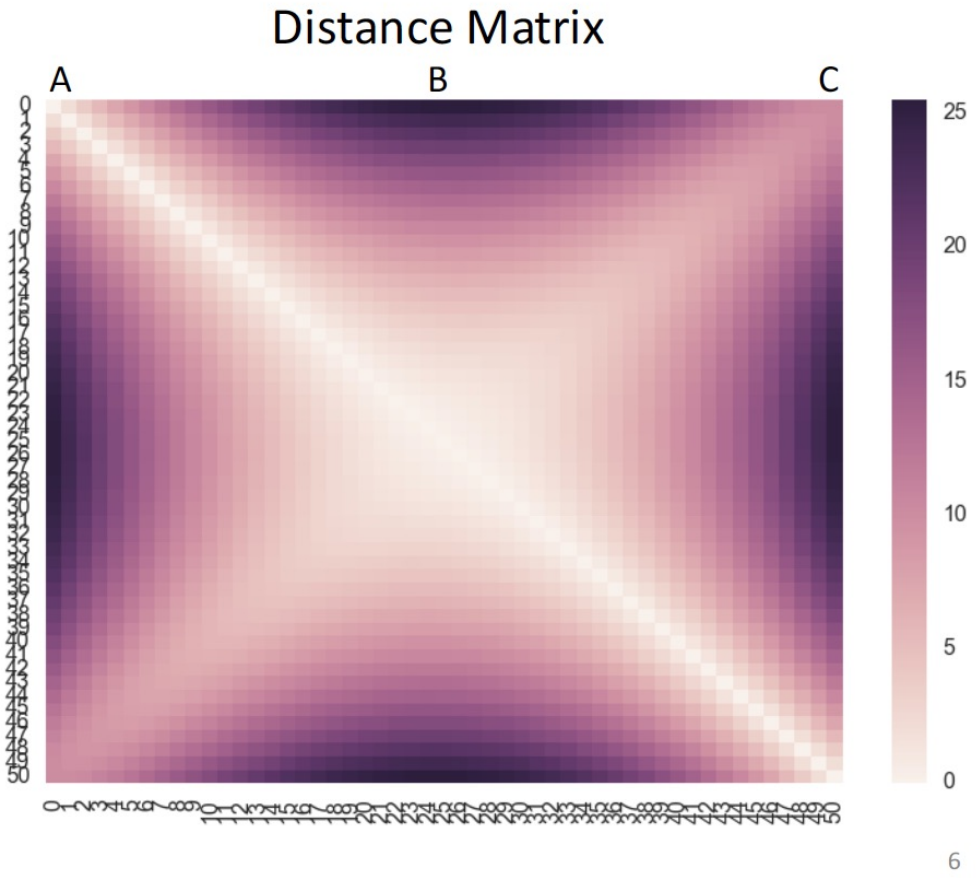


Swiss Roll

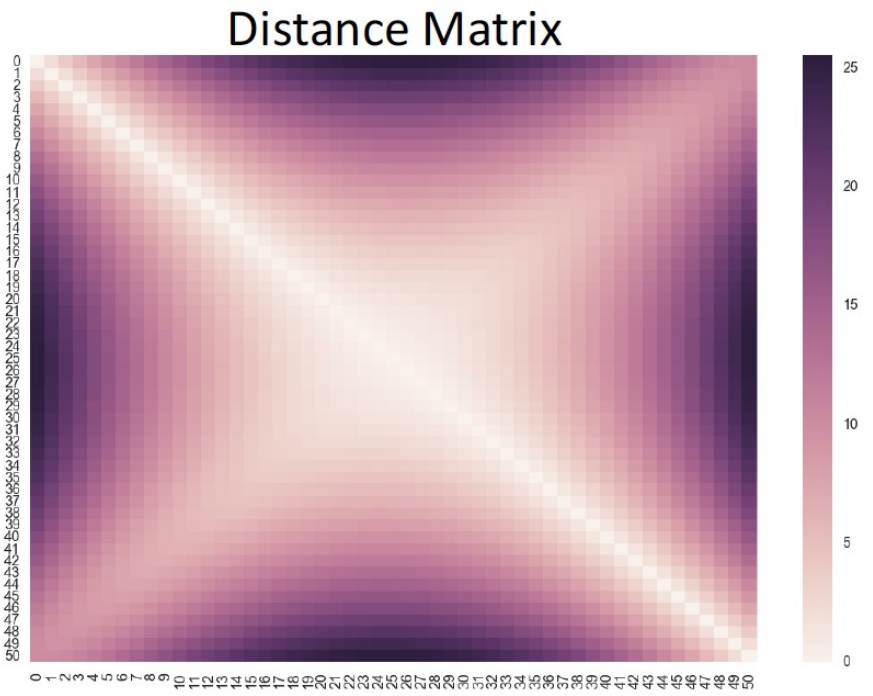


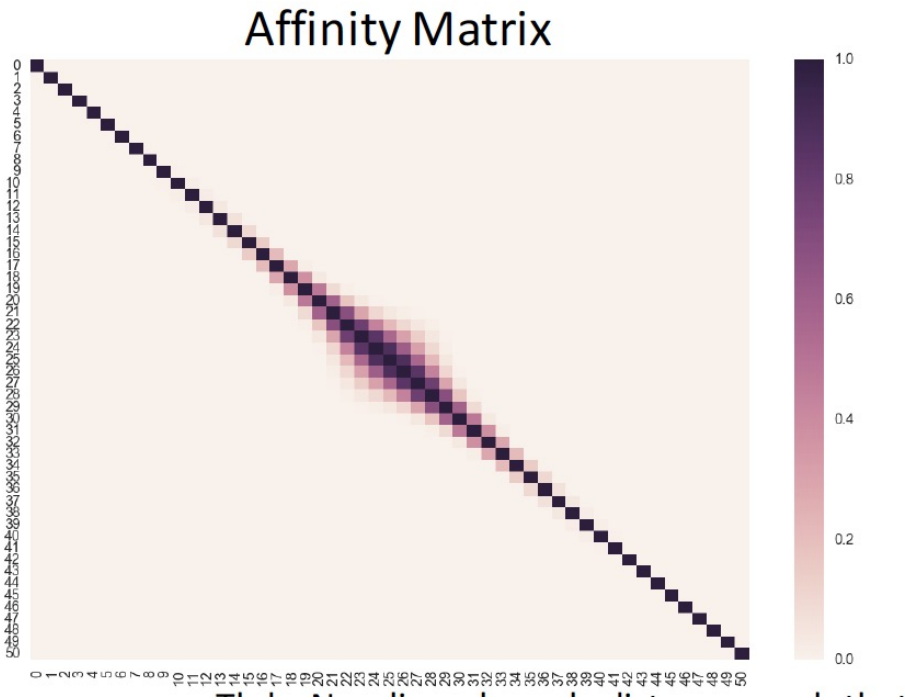
Example


30
25
20
15
10
5
0
−5
−30
−20
−10
0
10
20
30

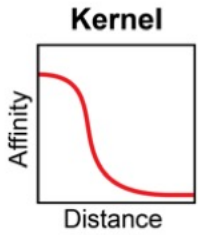

Distance Matrix
A
B
C
25
20
15
10
5
0

Affinity is the inverse proportional to distance

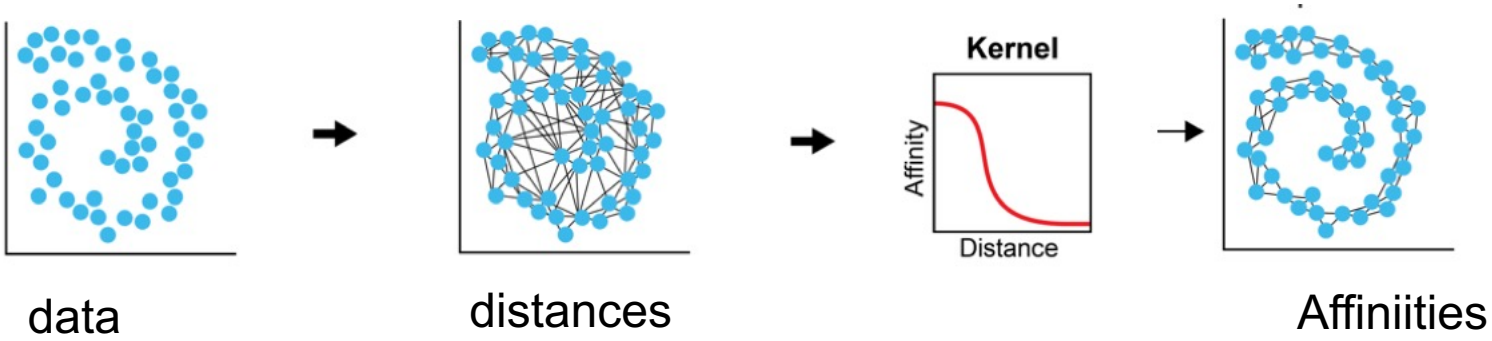




$$Affinity_{i,j} = s_{i,j} = exp\left(-\frac{dist(x_i, x_j)^2}{2\sigma^2}\right)$$

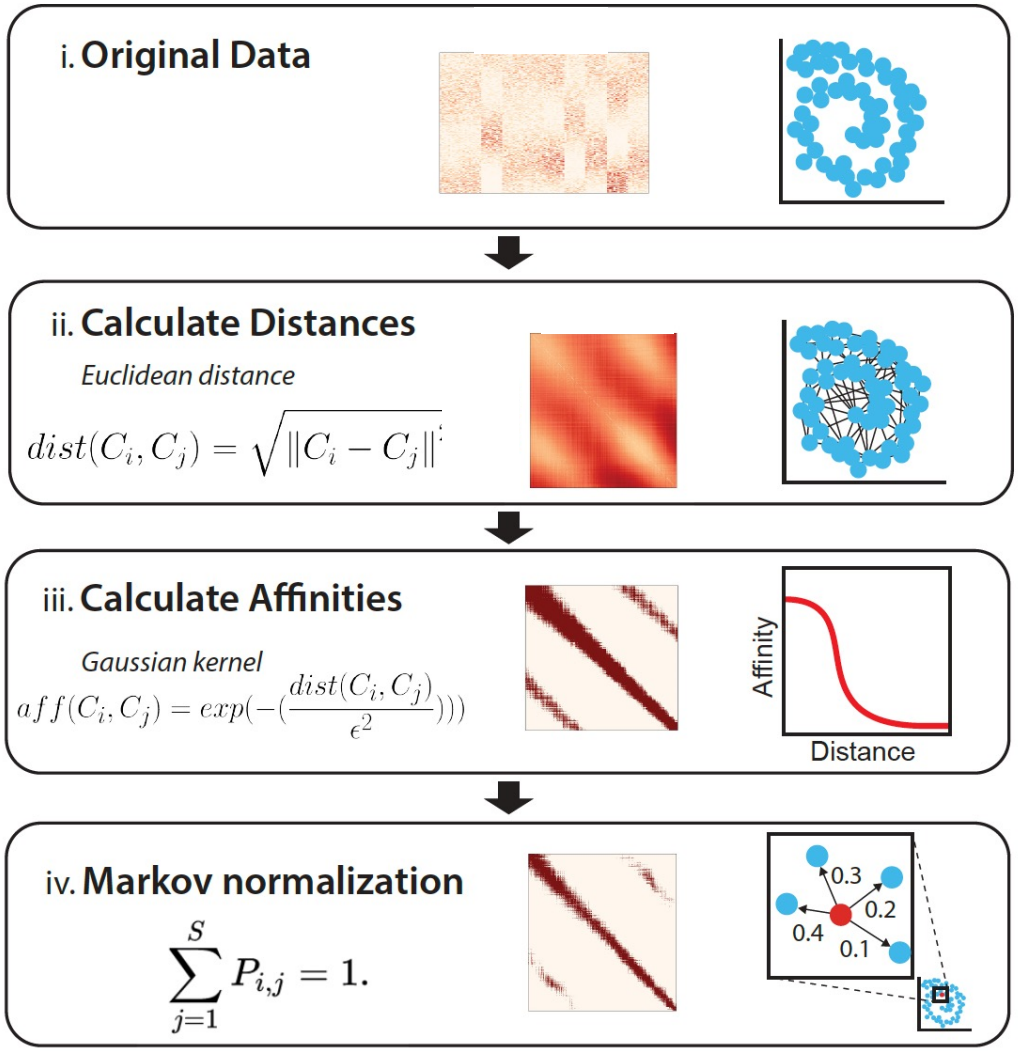


Kernel PCA



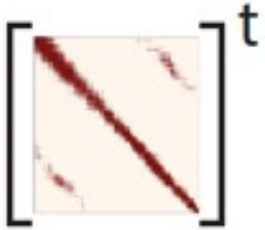
- Use eigenvectors of an **affinity matrix** instead of covariance matrix
- The family of methods called kernel PCA:
- Specific variants include:
  - Laplacian Eigenmaps
  - Diffusion Maps

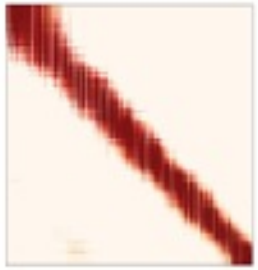
Non-Linear Dimensionality Reduction: Diffusion Map Steps



Continued

v. **Exponentiate markov matrix**



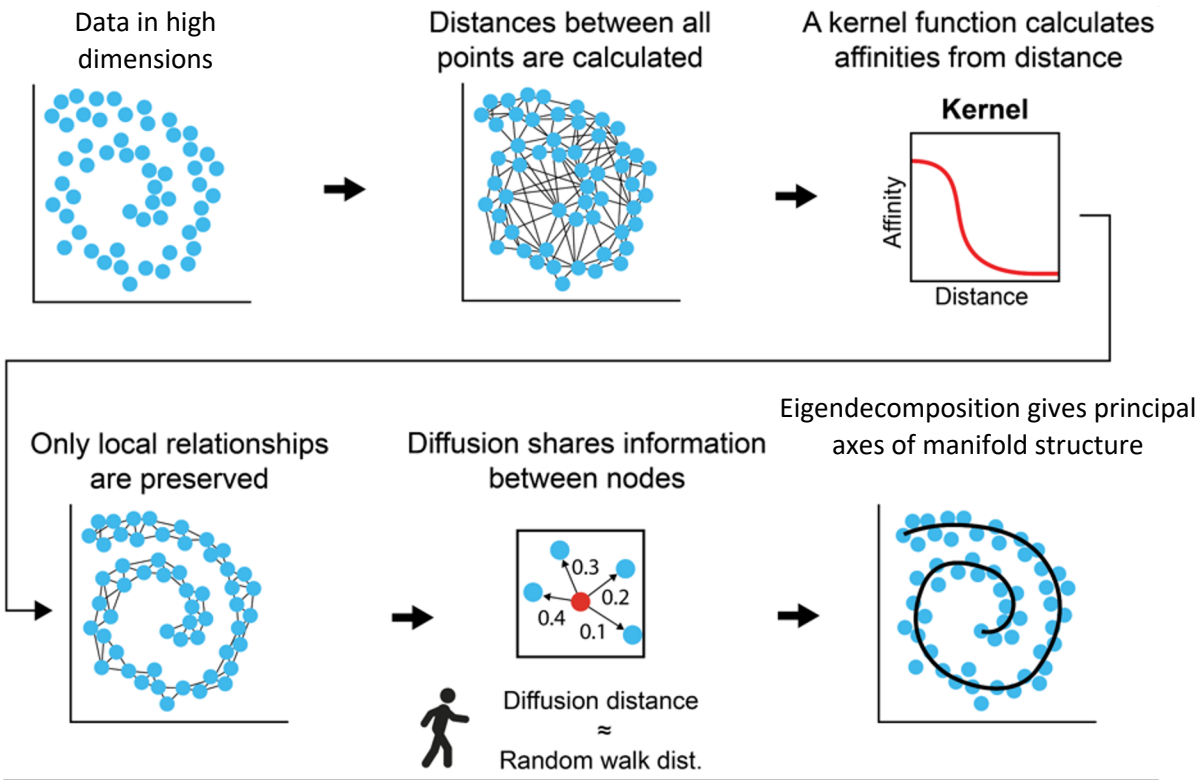


Eigendecompose

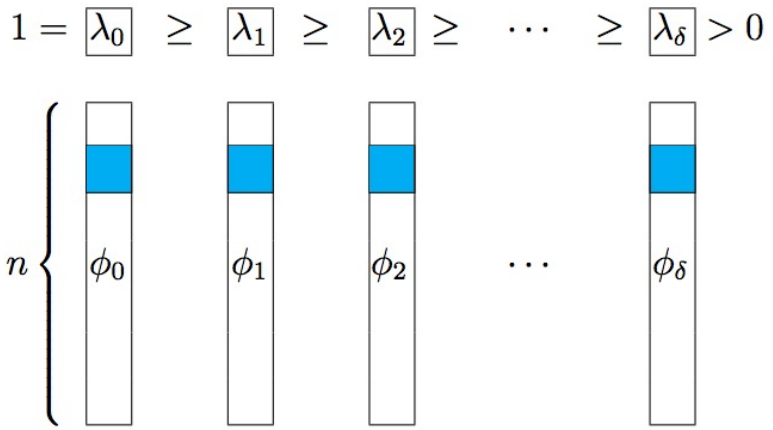
DM = Eigenvectors of Markov Matrix

- The **Diffusion MAP** is the set of eigenvectors of the Markov matrix
- Unlike PCA which finds the linear paths of greatest variation
- Eigenvalues similar to PCA explain the extent of variation explained by the diffusion component
- But first diffusion vector has a special meaning

Diffusion Maps



Diffusion Maps

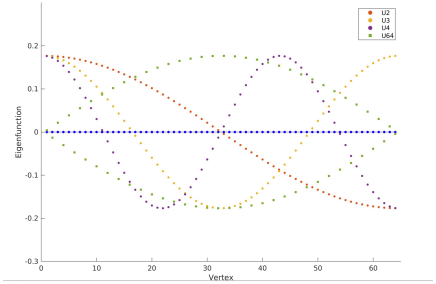


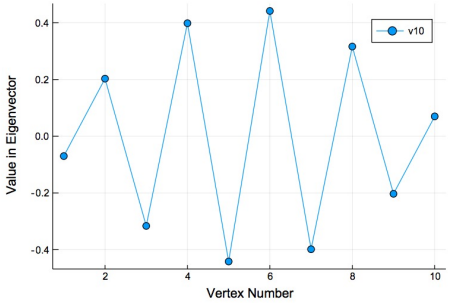
Coifman, Lafon 2006

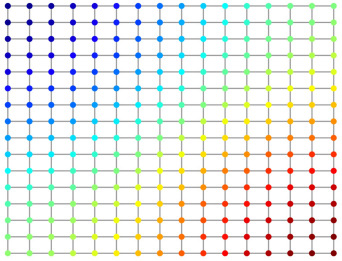
$$x \mapsto \Phi(x) \triangleq [\lambda_0\phi_0(x)\,,\ \lambda_1\phi_1(x)\,,\ \lambda_2\phi_2(x)\,,\ \ldots\,,\ \lambda_\delta\phi_\delta(x)]^T$$

[Coifman, Lafon, ACHA 2006]

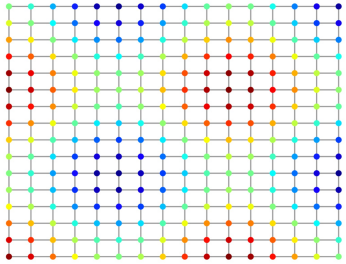
Eigenvectors are frequency harmonics



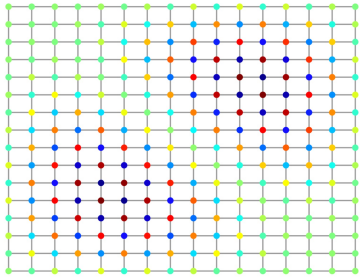




2nd Eigenvector

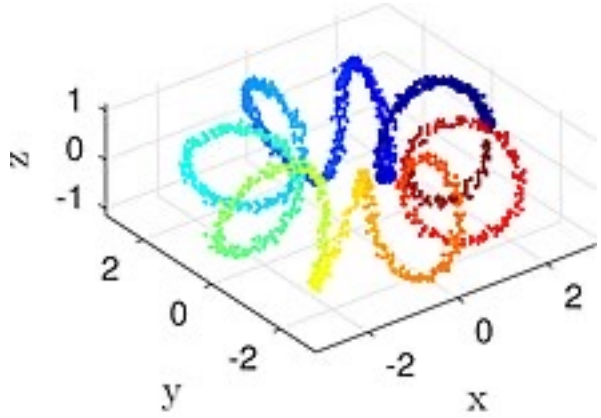


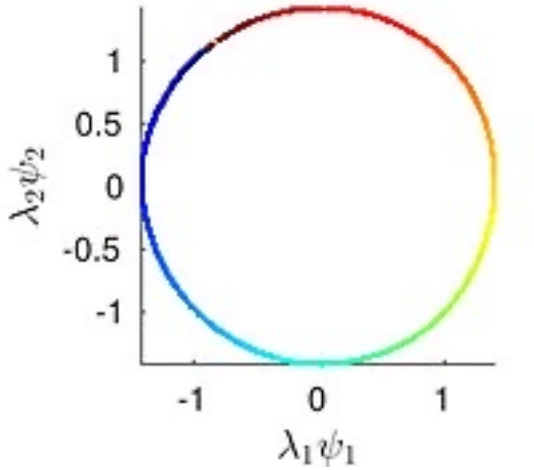
10th Eigenvector



2nd to last eigenvector

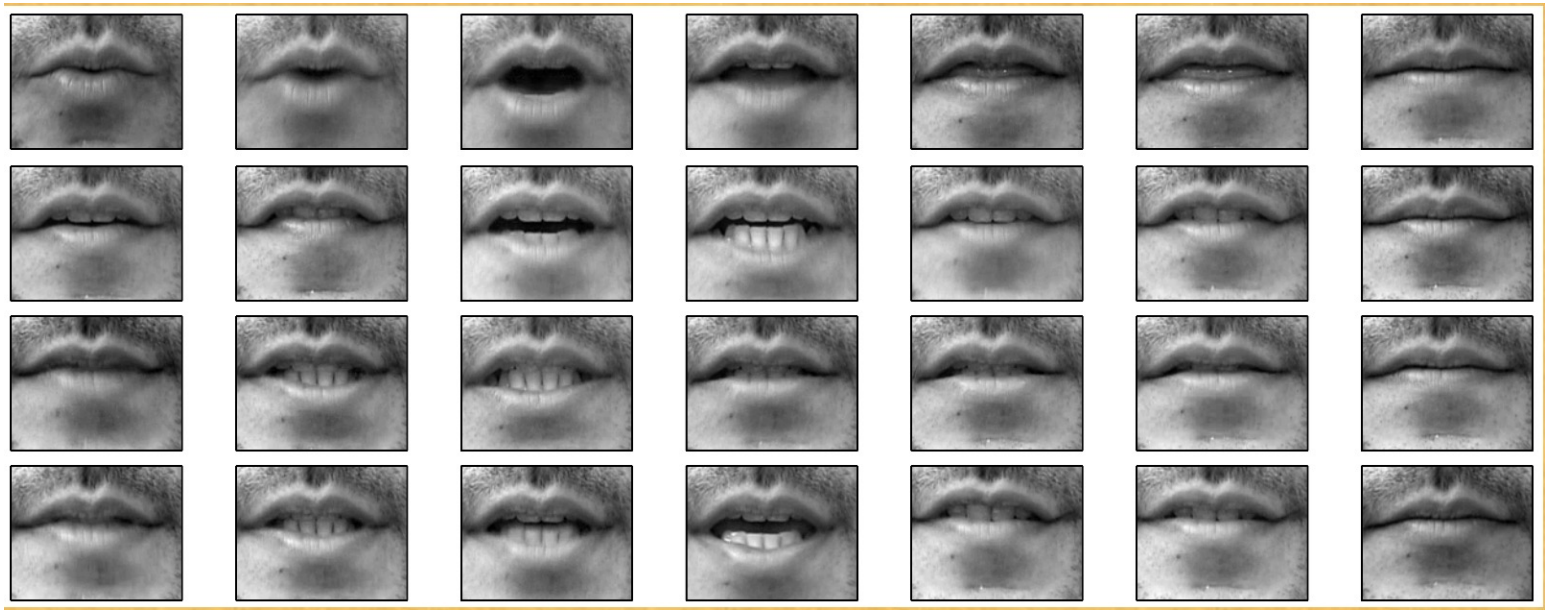
Example Embedding





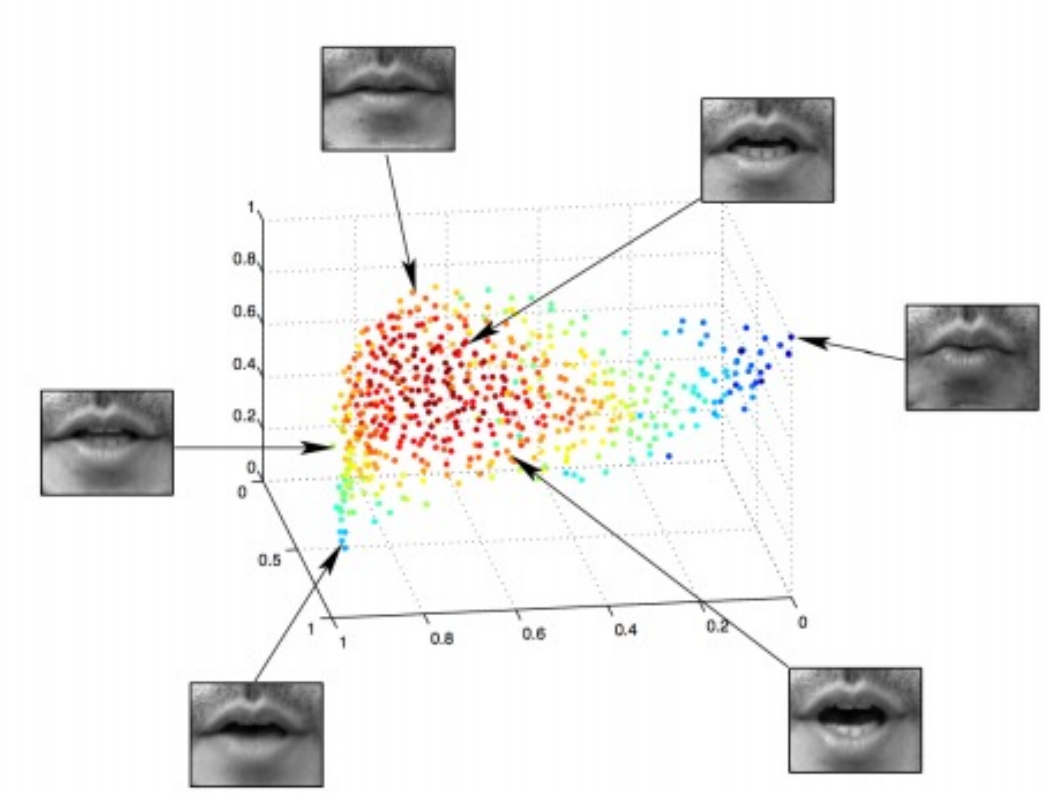
Lips Dataset: Spoken Digits

ONE



[3] S. Lafon, Y. Keller, and R. R. Coifman. Data Fusion and Multi-Cue Data Matching by Diffusion Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pages 1784-1797, 2006.

Lips Manifold



Opening of mouth

portion of teeth visible