Lecture 9

Dimensionality Reduction

ISLR 12, ESL 14, tSNE

Jilles Vreeken
Aleksandar Bojchevski

UNIVERSITÄT DES SAARLANDES

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Supervised vs. Unsupervised Learning

We focused mostly on supervised learning methods such as regression and classification

- The goal was to predict an outcome Y, from a set of features $X_1, X_2, \dots, X_p$

In unsupervised learning we are only given the features $X_1, X_2, \dots, X_p$ and we are interested in findings some interesting about the data or hidden (latent) structure

- Informative ways to visualize the data
- Discover subgroups (clusters) among the variables or observations
- Project the data from a high- to a low-dimensional space
- Anomaly detection

Other learning paradigms (out of scope for the lecture):

- reinforcement learning
- self-supervised learning (e.g. reducing unsupervised to supervised learning)

Unsupervised Learning

Unsupervised learning is more exploratory and thus more challenging

- we have no clear target question – no output guides our predictions
- therefore, it is difficult to assess the quality of our results
- compared to supervised where we could just look at e.g. the test error

Big advantage: Much easier to obtain large amounts of unlabeled data

Examples

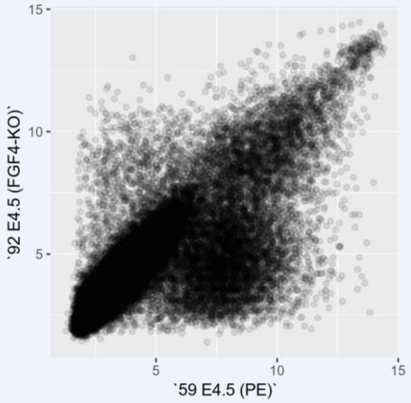
- grouping genomic signatures of cancer samples by subtype
- characterizing shoppers browsing and purchasing habits
- movies grouped by the ratings assigned by movie viewers

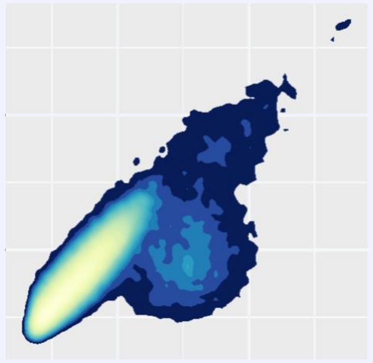
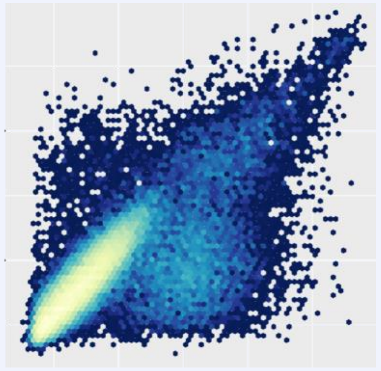
Visual Data Exploration (Tutorial)

1D data: compute summary statistics: mean, mode, median, quartiles, box-whiskers plot

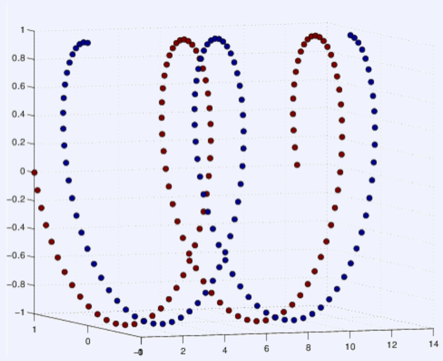
1D data distribution: histograms, dots and beeswarm plot, kernel density estimation, violin plot

2D data: scatter plots, density plots, hexagon plots

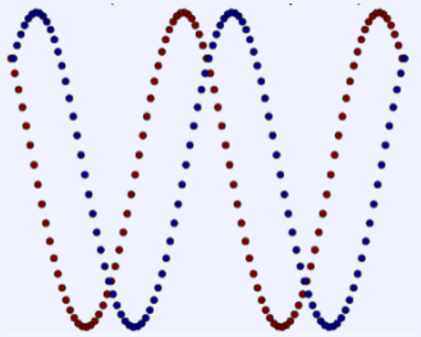


Visualizing more than two dimensions

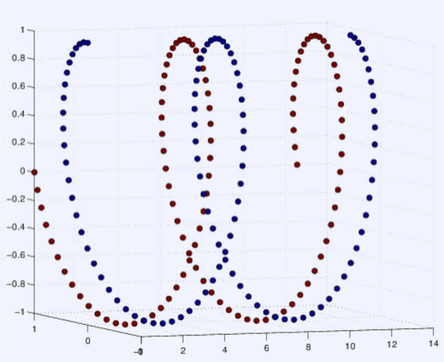


original 3D data

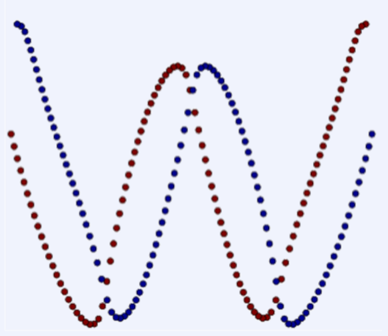


2D reduction with PCA

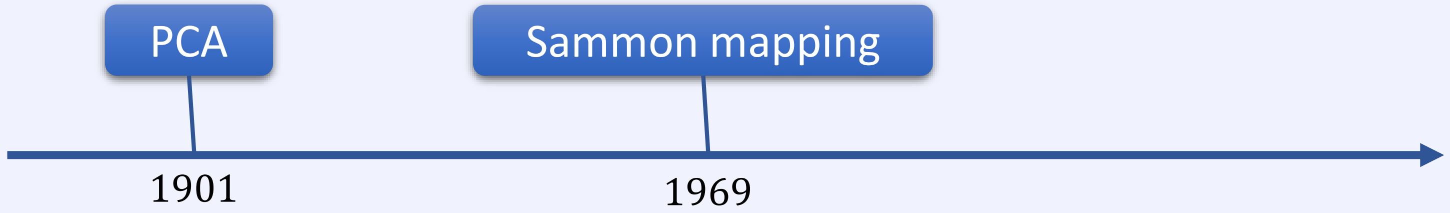
Visualizing more than two dimensions



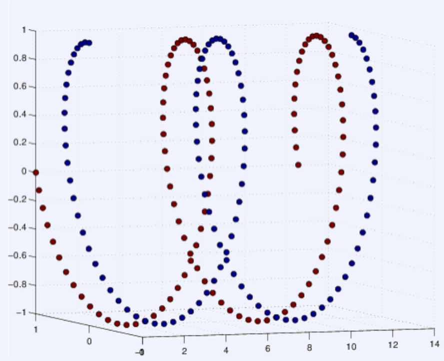
original 3D data



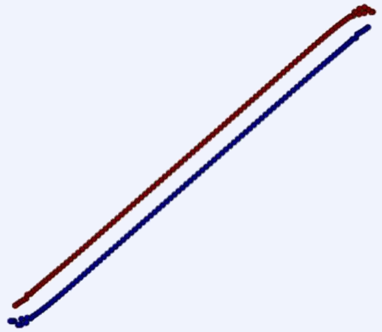
2D reduction with Sammon mapping



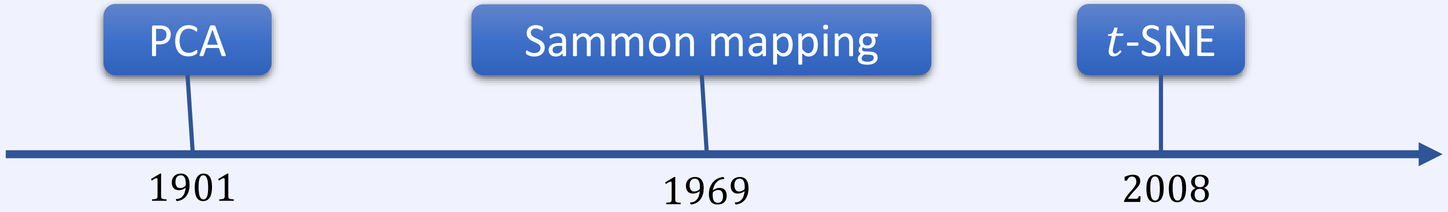
Visualizing more than two dimensions



original 3D data



2D reduction with t-SNE



Dimensionality Reduction: Further Motivation

High-dimensional data is challenging:

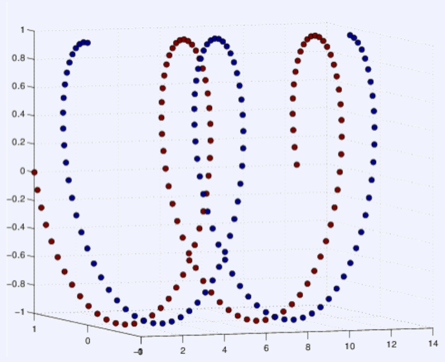
- hard to visualize high-dimensional data
- highly correlated dimension could cause trouble for some algorithms
- curse of dimensionality: we need exponential amounts of data to characterize the density as the dimensionality goes up
- computation is expensive because of high complexity of distance functions

Often the data lies on a low-dimensional manifold, embedded in a high-dimensional space

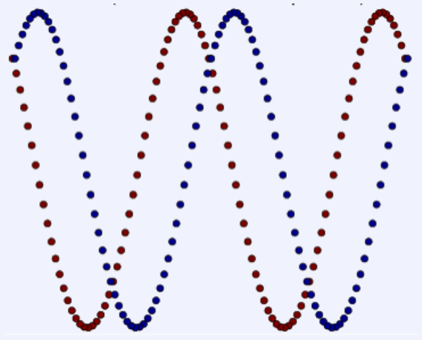
Goal: Reduce the dimensionality while avoiding information loss and preserving the structure

- computational or memory savings
- uncover the intrinsic dimensionality of the data

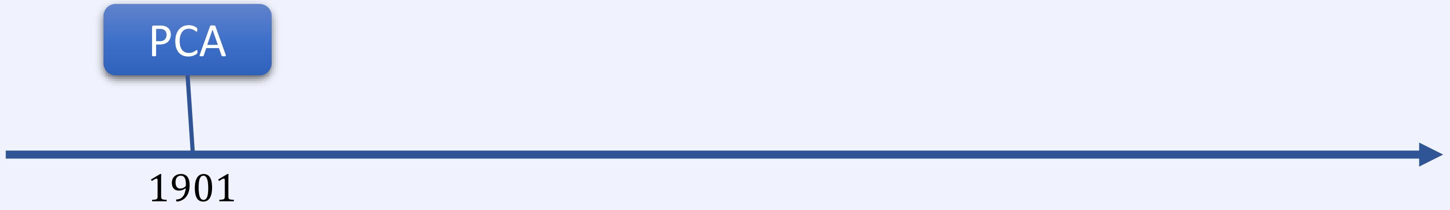
PCA



original 3D data



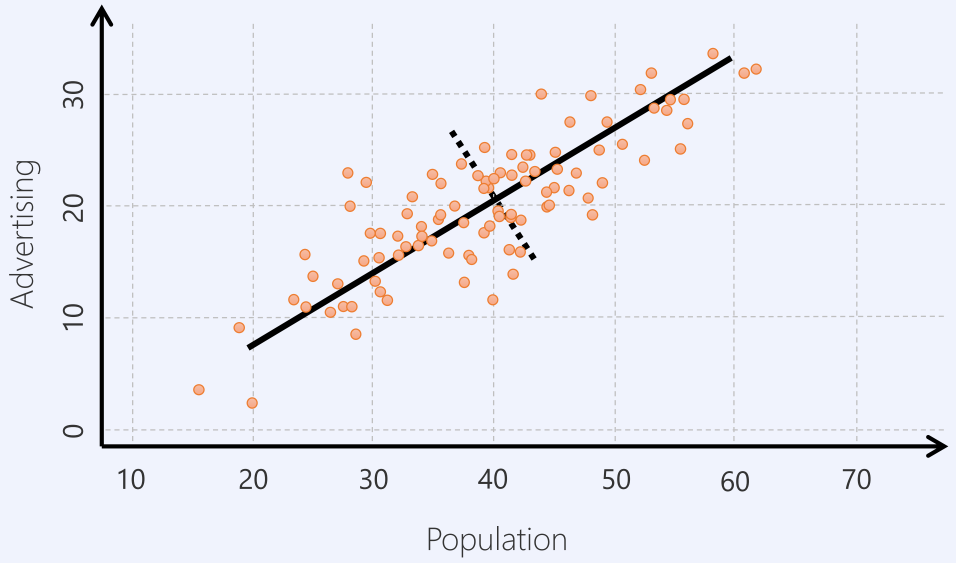
2D reduction with PCA



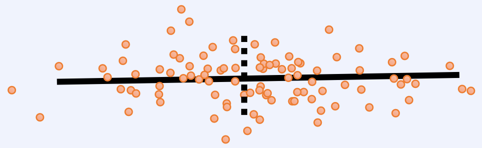
Principal Component Analysis

Example: population and ad spending for 100 different cities shown as circles

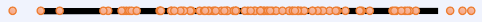
- Data are roughly linear along one direction with a small variance along a second direction
- Solid line indicates the first principal component (PC) direction, and dotted line the second PC
- Most of the variation is along the first PC



- The PCs define a new coordinate system



- Project points onto the first PC



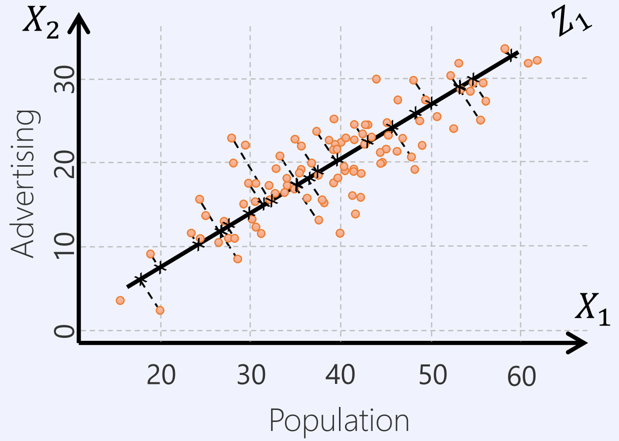
Principal Component Analysis

The first PC is the direction in space along which **variance** of data is **greatest**

- if projected onto this direction the resulting one-dimensional dataset has the largest possible variance
- The $\boldsymbol{j^{th}}$ PC is the direction orthogonal to all previous PCs, on which the remaining variance is largest

At the same time the first PC **minimizes** the sum of squared distances (dashed lines)

- the line that is closest to all the observations



Principal Component Analysis

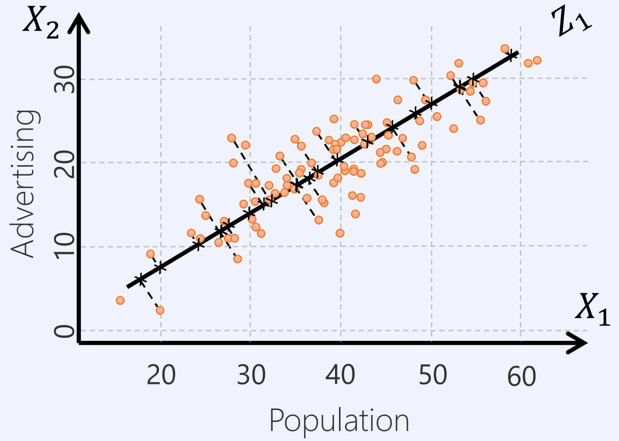
Formally we define the first PC Z_1 as a **linear combination** of mean-centered X_j

$$Z_1 = \sum_{j=1}^{p} \phi_{j1}(X_j - \bar{X}_j) \text{ for constants } \phi_{11}, \phi_{21}, \dots, \phi_{p1} \text{ and means } \bar{X}_j$$

- we require $\phi_{11} + \phi_{21} + \cdots + \phi_{p1} = 1$ to prevent arbitrary scaling
- find ϕ_{j1} such that variance is maximized / distance is minimized
- Z_1 is a n-dimensional vector
- its components z_{i1} are called the **PC scores**

- Solve the following problem subject to the scaling constraint

$$\max_{\phi_{11},\phi_{21},\dots,\phi_{p1}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2}_{\text{variance}} = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}(X_j - \bar{X}_j)\right)^2$$



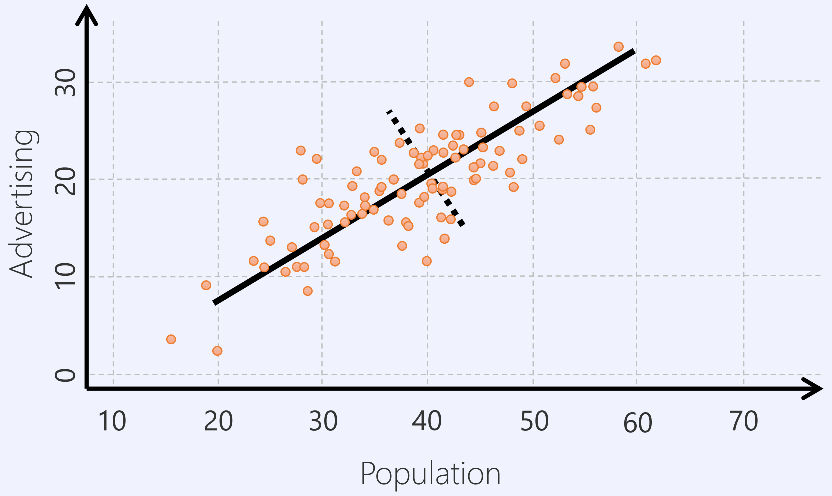
Principal Component Analysis

Example:

- first PC $Z_1 = 0.839(\mathbf{pop} - \overline{\mathbf{pop}}) + 0.544(\mathbf{ad} - \overline{\mathbf{ad}})$
- $\phi_{11} = 0.839, \phi_{21} = 0.544$ component loadings

- out of every linear combination of **pop** and **ad** with $\phi_{11}^2 + \phi_{21}^2 = 1$ the first PC has the highest variance i.e. $Var(\phi_{11}(\mathbf{pop} - \overline{\mathbf{pop}}) + \phi_{21}(\mathbf{ad} - \overline{\mathbf{ad}}))$ is maximum
- at the same time first PC is the closest line to the data

- second PC $Z_2 = 0.544(\mathbf{pop} - \overline{\mathbf{pop}}) - 0.839(\mathbf{ad} - \overline{\mathbf{ad}})$



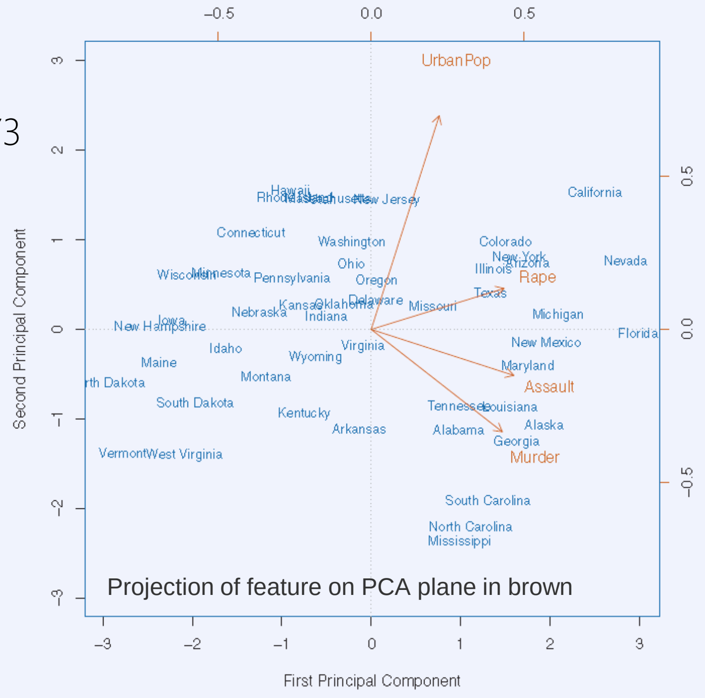
Principal Component Analysis

PCA of the **USArrests** dataset

- statistics in arrests per 100,000 residents in the US in 1973
- 50 observations, 1 per state, 4 inputs
  - **Murder** numeric murder arrests
  - **Assault**: numeric assault arrests
  - **UrbanPop**: percent urban population
  - **Rape**: numeric rape arrests

	PC1	PC2
Murder	**0.5358995**	**-0.4181809**
Assault	**0.5831836**	**-0.1879856**
UrbanPop	**0.2781909**	**0.8728062**
Rape	**0.5434321**	**0.1673186**

PCA "loading vector"
direction of the principal component



(ISLR 10.2)

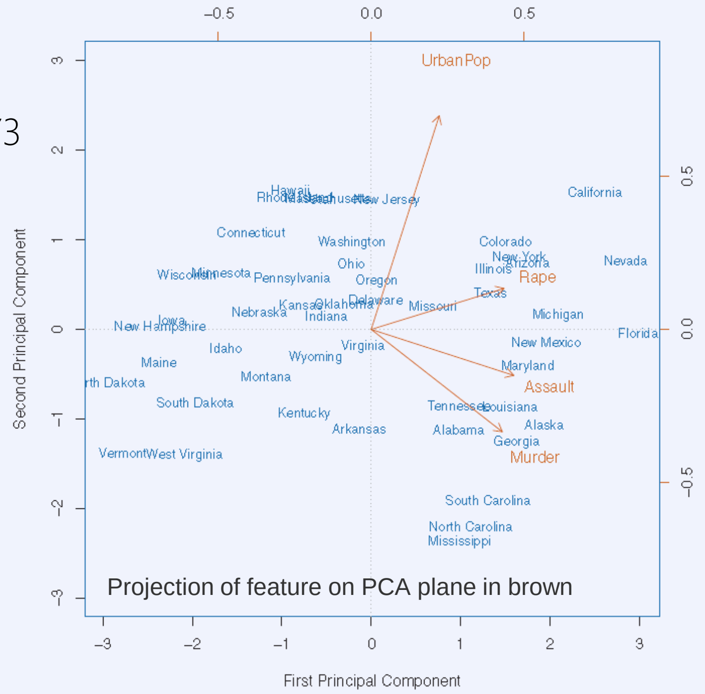
Principal Component Analysis

PCA of the **USArrests** dataset

- statistics in arrests per 100,000 residents in the US in 1973
- 50 observations, 1 per state, 4 inputs
  - **Murder** numeric murder arrests
  - **Assault**: numeric assault arrests
  - **UrbanPop**: percent urban population
  - **Rape**: numeric rape arrests

	PC1	PC2
Murder	**0.5358995**	**-0.4181809**
Assault	**0.5831836**	**-0.1879856**
UrbanPop	**0.2781909**	**0.8728062**
Rape	**0.5434321**	**0.1673186**

projection of feature vectors on principal component surface



Principal Component Analysis

Interpretation

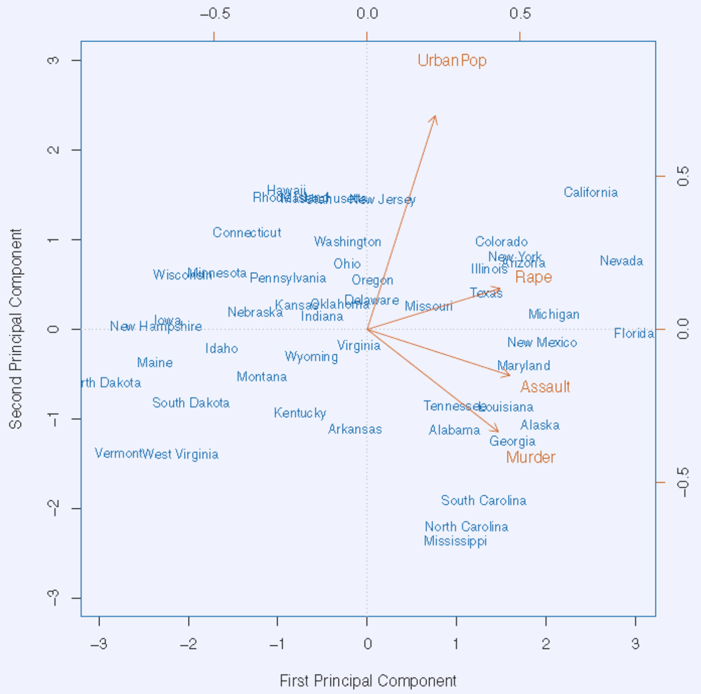
- crime variables are highly correlated:
  - projection vectors point in about same direction
- less correlation with **`UrbanPop`**

PC1 reflects crime rate

- high in California, Nevada, Florida
- low in W.-Virginia, the Dakotas etc.

PC2 reflects urbanization

- high in California
- low in the Carolinas and Mississippi



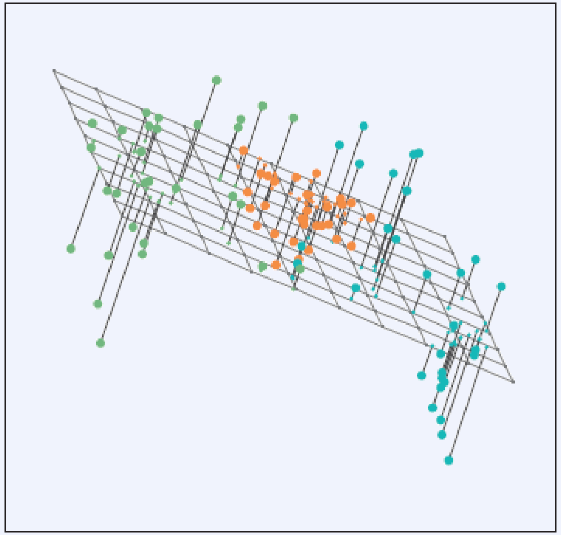
Principal Component Analysis

Interpretation 1

- PCs are directions of highest variance of the data
- PC score of an input is its projection onto the PC loading vector

Interpretation 2

- first PC minimizes the total sum of square distances
- second PC is the first PC of the residual, i.e. the direction in which the variance of the residual is maximized / distance is minimized
- the PC hyperplane is the affine subspace such that the total sum of square distances from the subspace is minimal


3D simulated dataset with the first two PCs

Interpretation 3

- PCA finds a a linear transformation into a new coordinate system where the data is linearly uncorrelated

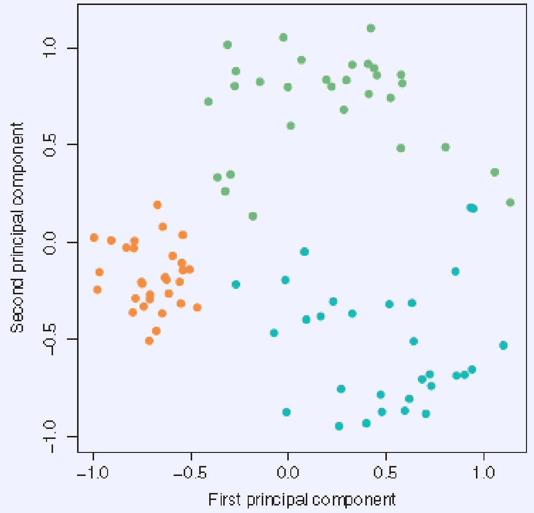
Principal Component Analysis

Interpretation 1

- PCs are directions of highest variance of the data
- PC score of an input is its projection onto the PC loading vector

Interpretation 2

- first PC minimizes the total sum of square distances
- second PC is the first PC of the residual, i.e. the direction in which the variance of the residual is maximized / distance is minimized
- the PC hyperplane is the affine subspace such that the total sum of square distances from the subspace is minimal



3D simulated dataset with the first two PCs

Interpretation 3

- PCA finds a a linear transformation into a new coordinate system where the data is linearly uncorrelated

How to choose the number of PCs

If the goal is to use PCA for visualization the we can only select 2 or 3

If the goal is to preprocess the data before another method (e.g. before running regression)

- select #PCs such that a target proportion of the total variance is explained (PVE)
  - total variance is $\sum_{j=1}^{p} \mathrm{Var}(X_j)$
  - variance explained by the m-th principal component $\mathbf{Var}(Z_m)$
- if we select k componets we explain $\frac{\sum_{i=1}^{k} \mathrm{Var}(Z_m)}{\sum_{j=1}^{p} \mathrm{Var}(X_j)}$
  - select k such that the above fraction equals e.g. 90%
  - look for an elbow in the PVE plot

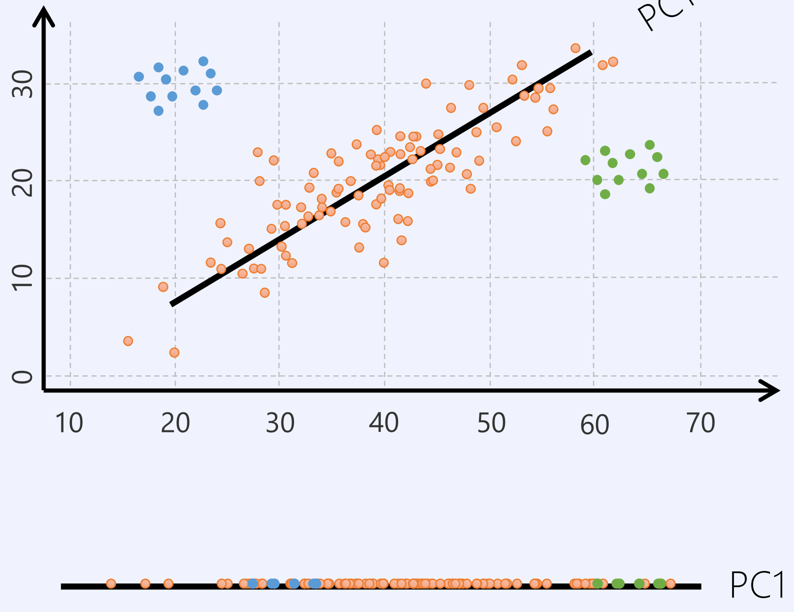
We can also just use cross-validation on the final dowstream error

Principal Component Analysis

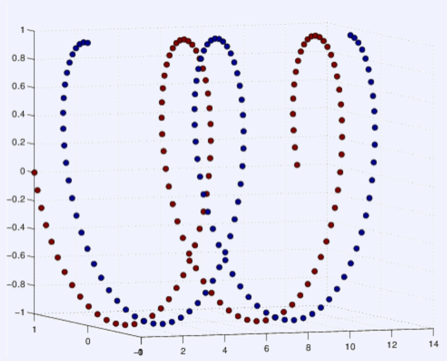
PCA finds the global (linear) structure in the data

- can lead to local inconsistencies
- far away points can become nearest neighbors
- depending on the application this is a problem

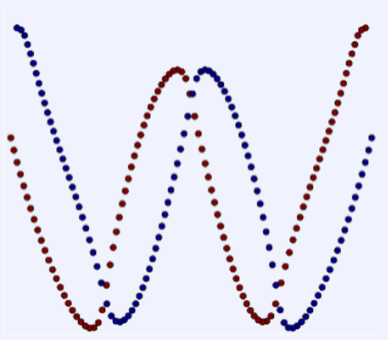
Idea: Preserve local structure (distances) instead



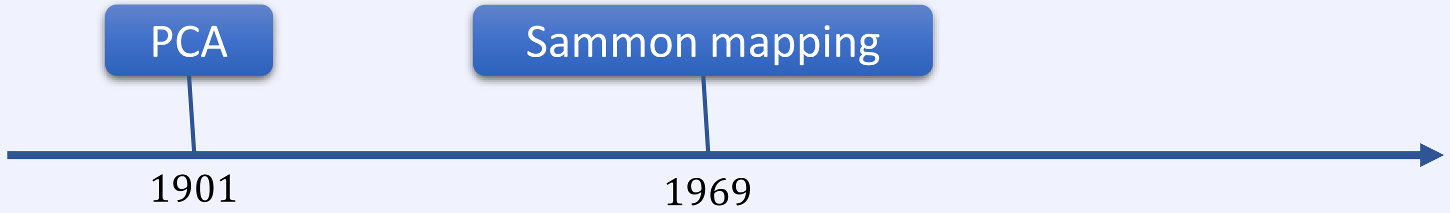
Sammon Mapping (MDS)



original 3D data



2D reduction with Sammon mapping



MDS: Multidimensional Scaling

Project high-dimensional distances onto low-dimensional space $\mathbb{R}^k$

- let data points be $x_1, \ldots, x_N \in \mathbb{R}^p$
- project onto $z_1, \ldots, z_N \in \mathbb{R}^k$
- minimize a stress function S

Kruskal-Shepard (least-squares): $S_M(z_1, \ldots, z_N) = \sum_{i \neq i'} (d_{ii'} - \|z_i - z_{i'}\|)^2$

Sammon mapping: $S_{S_m}(z_1, \ldots, z_N) = \sum_{i \neq i'} \frac{(d_{ii'} - \|z_i - z_{i'}\|)^2}{d_{ii'}}$

- emphasizes preserving smaller distances

Multidimensional Scaling & PCA

Minimization by gradient descent

- classic scaling for similarities $s_{ii'}$
- often we use the centered inner product $s_{ii'} = \langle x_i - \bar{x}, x_{i'} - \bar{x} \rangle$
- we then minimize

$$S_C(z_1, \dots, z_N) = \sum_{i,i'} (s_{ii'} - \langle z_i - \bar{z}, z_{i'} - \bar{z} \rangle)^2$$

by choosing $z_1, \dots, z_N \in \mathbb{R}^k$

- this has a solution in terms of eigenvectors
- if the similarities are centered inner products then in fact this is exactly principal components

Multidimensional Scaling

MDS only needs the similarities or dissimilarities, not the point coordinates

Non-metric version of Shepard-Kruskal scaling only uses ranks

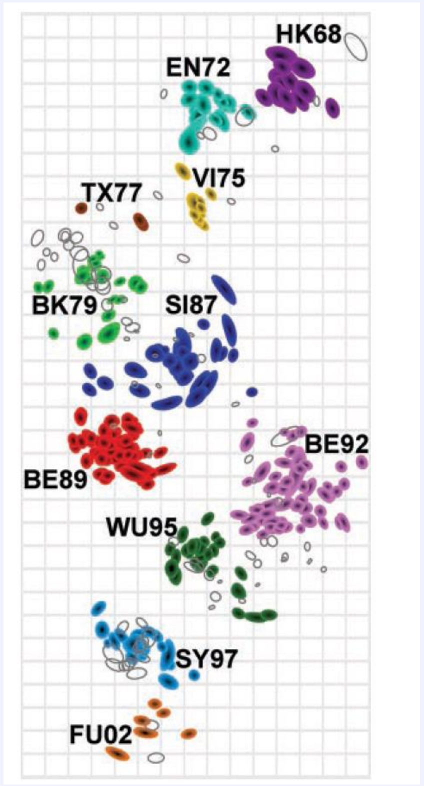
$$S_{NM}(z_1, \dots, z_N) = \frac{\sum_{i \neq i'} [\|z_i - z_{i'}\| - \theta(d_{ii'})]^2}{\sum_{i \neq i'} \|z_i - z_{i'}\|^2}$$

- θ is an arbitrary increasing function
- with θ fixed we minimize over z_i by gradient descent
- with z_i fixed the best monotonic θ is found by “isotonic regression” (version of quadratic programming)

Multidimensional Scaling

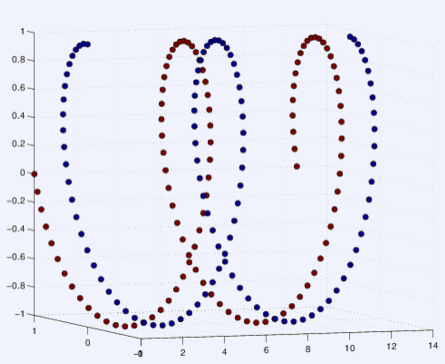
Example: Antigenic shift of influenza virus

- original space has 79 dimensions
- multiple runs of gradient descent with random starting solutions
- level of increase of stress function with decreasing $\boldsymbol{k}$
  can point to “dimensionality” of the data
- here results do not change significantly
  if one projects to 2, 3, 4, or 5 dimensions

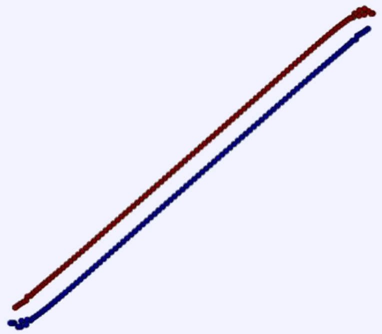


(Smith, Lapedes et al. Science 2004) 25

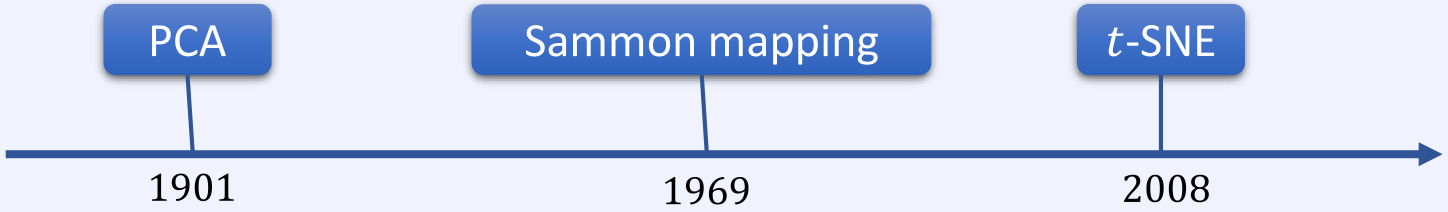
t-SNE



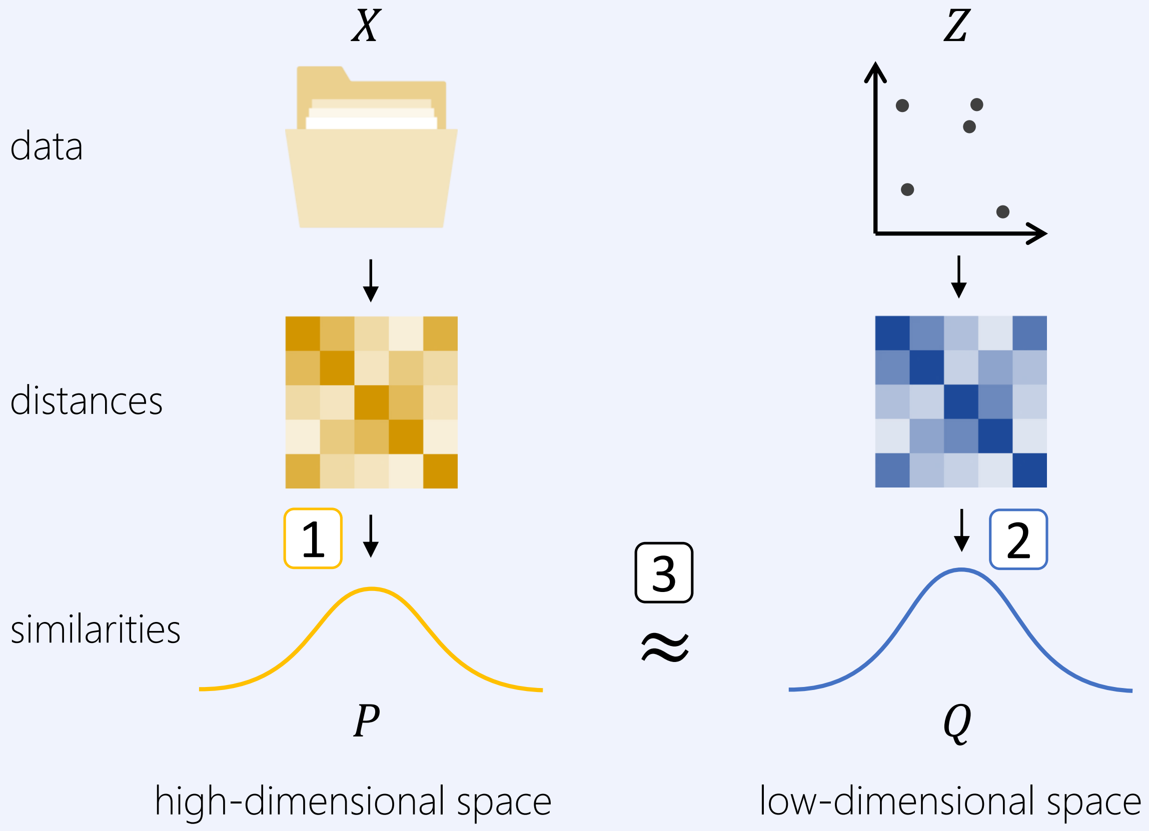
original 3D data



2D reduction with t-SNE

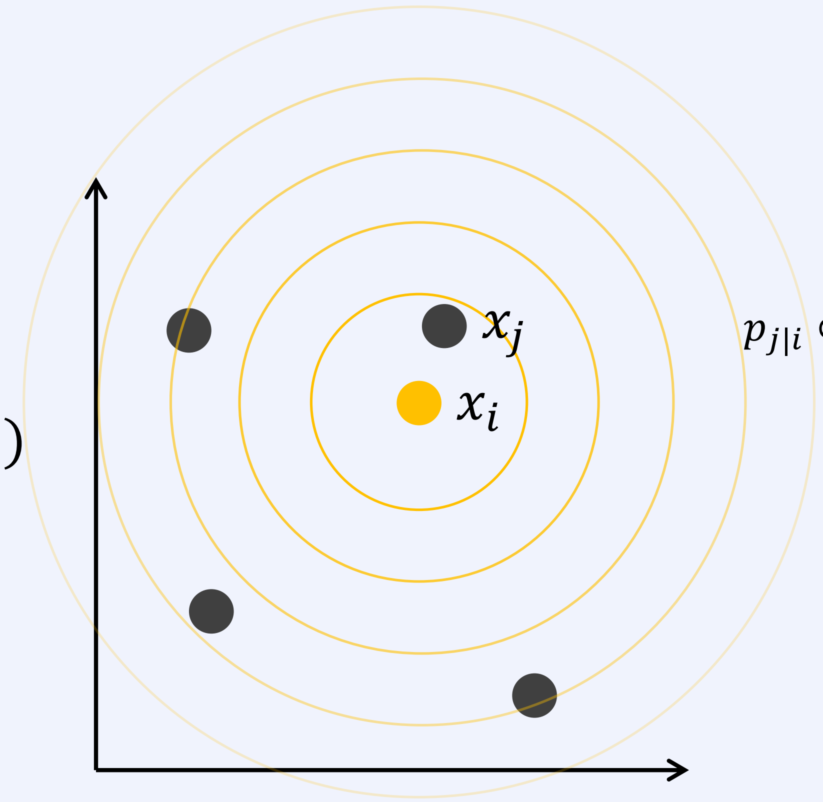


Stochastic Neighbor Embedding



1 High-Dimensional Similarities

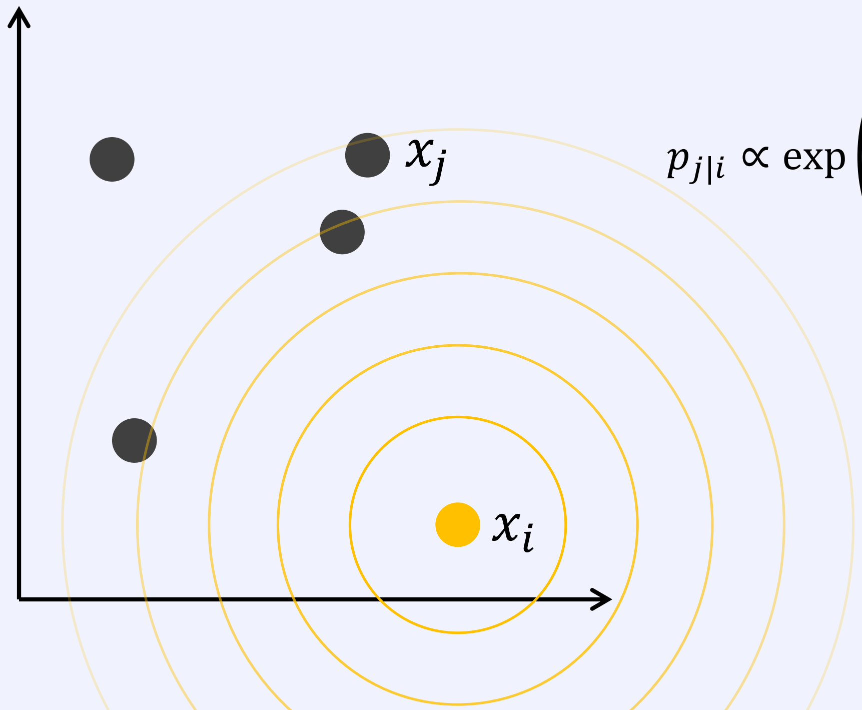
$$P_i \propto N(x_i, \sigma_i)$$



$$p_{j|i} \propto \exp\left(-\frac{\left\|x_j - x_i\right\|^2}{2\sigma_i^2}\right)$$

1 High-Dimensional Similarities

$$P_i \propto N(x_i, \sigma_i)$$



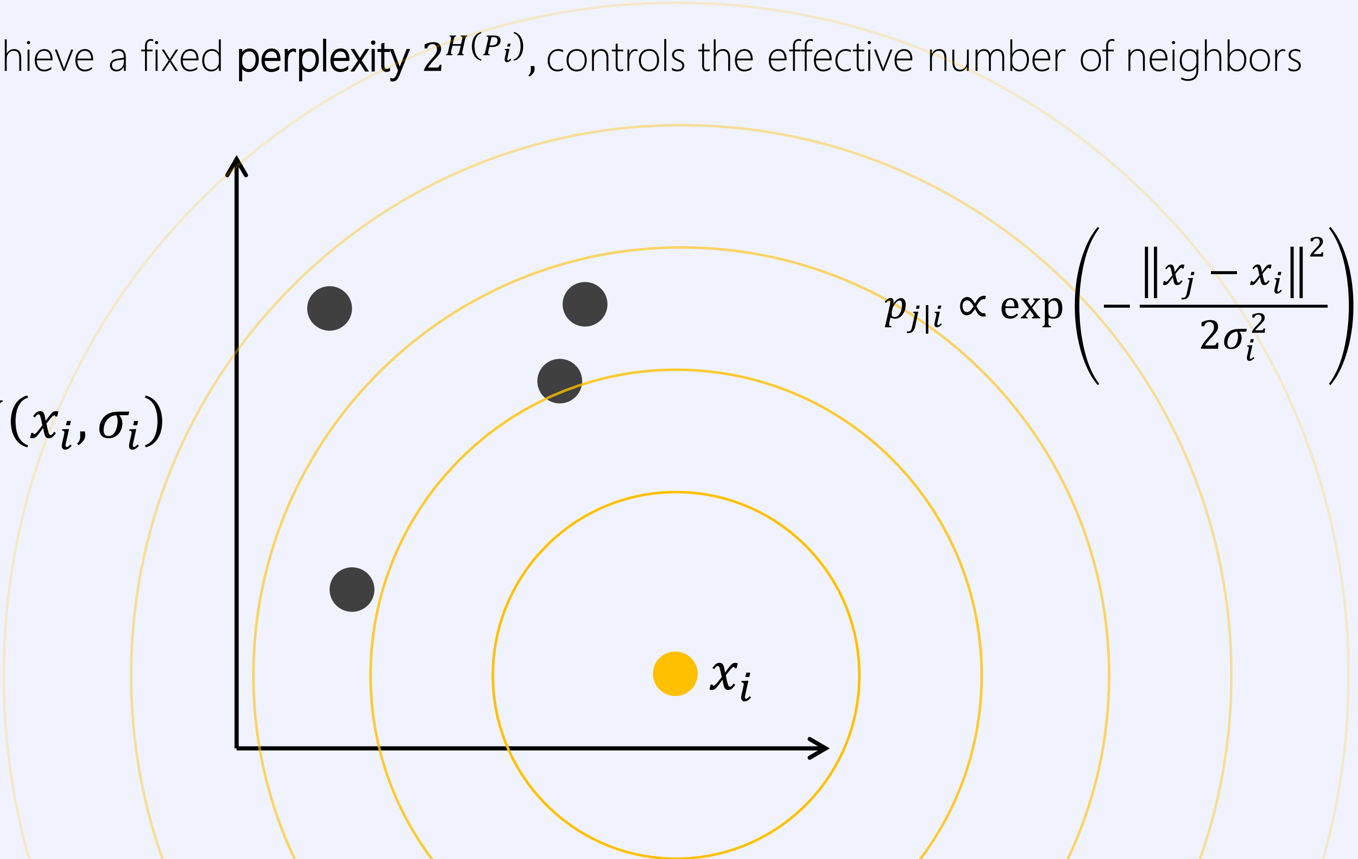
$$p_{j|i} \propto \exp\left(-\frac{\left\|x_j - x_i\right\|^2}{2\sigma_i^2}\right)$$

1 High-Dimensional Similarities

Choose σ_i to achieve a fixed **perplexity** $2^{H(P_i)}$, controls the effective number of neighbors

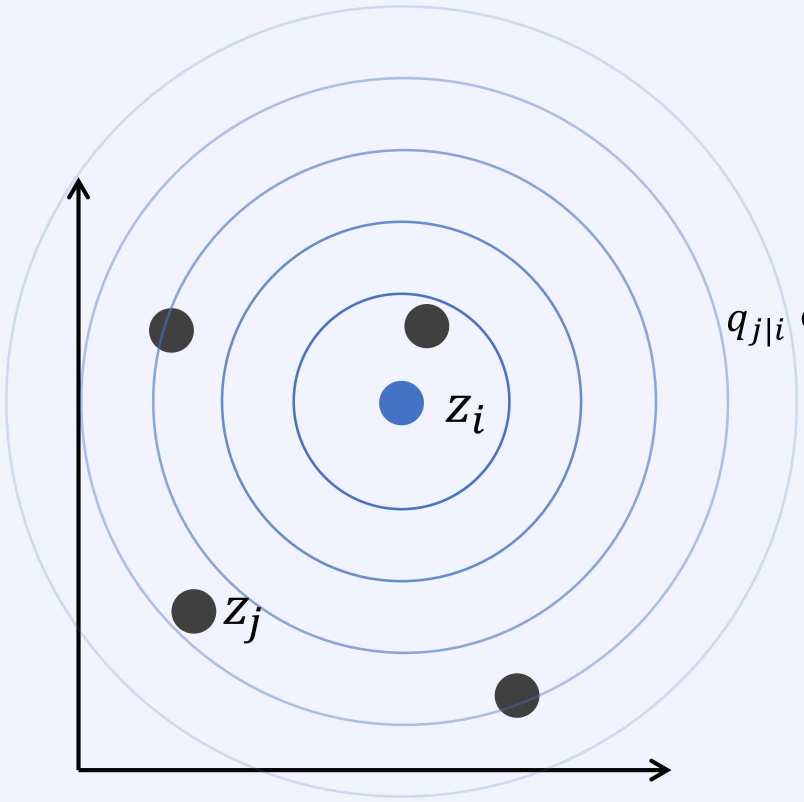
$$P_i \propto N(x_i, \sigma_i)$$

$$p_{j|i} \propto \exp\left(-\frac{\left\|x_j - x_i\right\|^2}{2\sigma_i^2}\right)$$



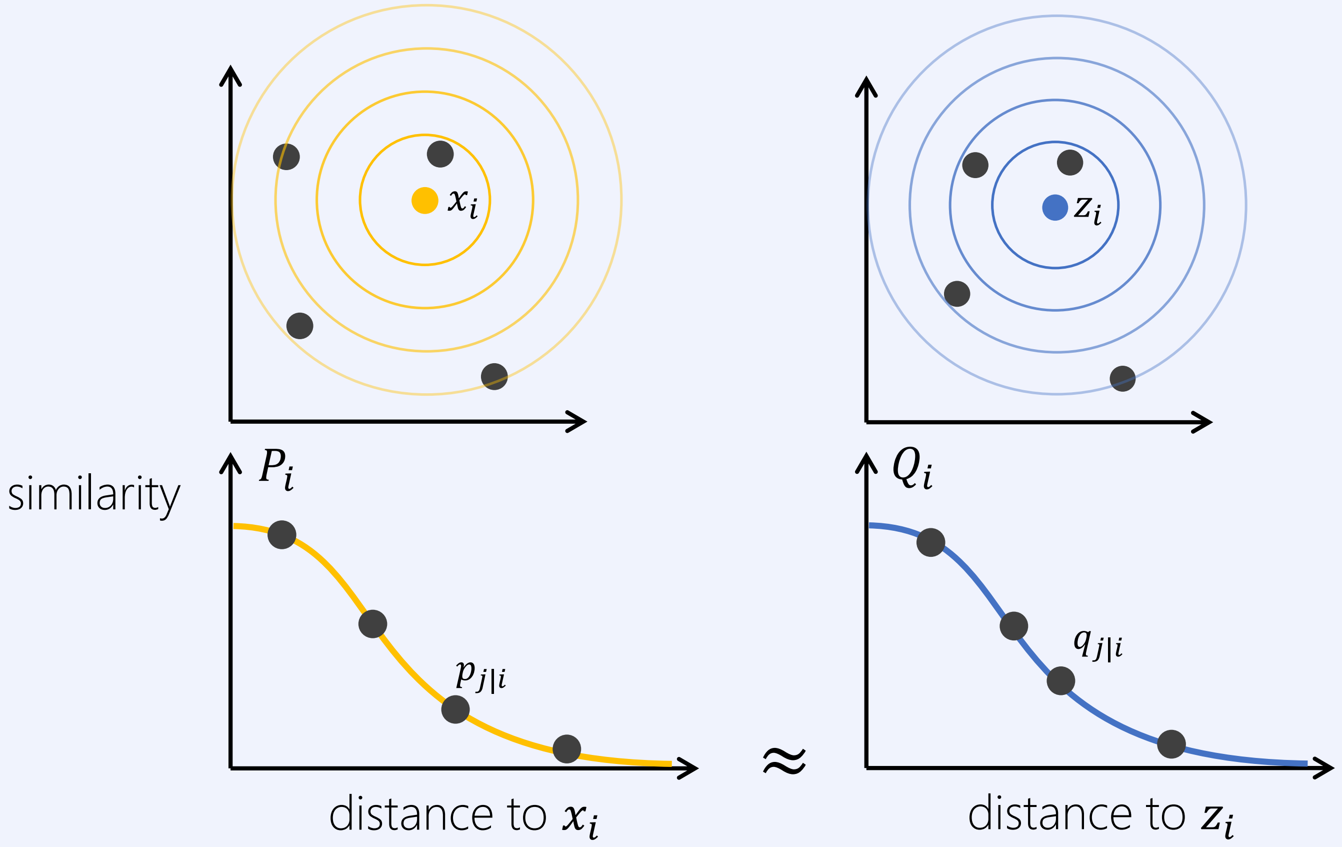
2 Low-Dimensional Similarities

$$Q_{j|i} \propto N\left(z_i, \frac{1}{\sqrt{2}}\right)$$



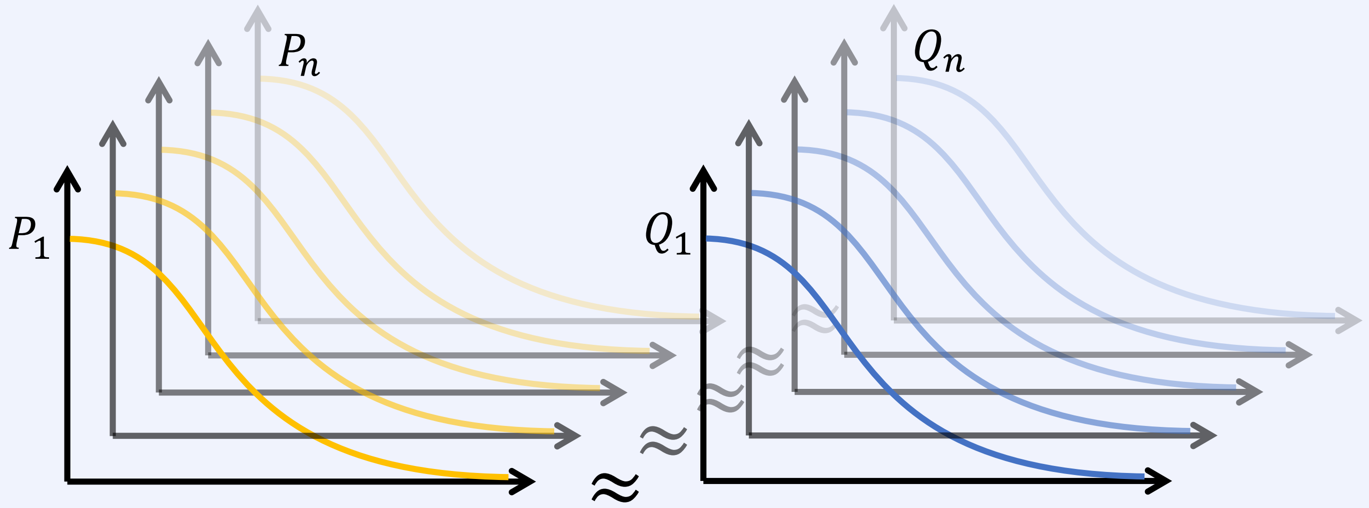
$$q_{j|i} \propto \exp\left(-\frac{\left\|z_j - z_i\right\|^2}{2}\right)$$

3 Comparing Distributions


x_i
z_i
similarity
P_i
Q_i
$p_{j|i}$
$q_{j|i}$
$\approx$
distance to $\boldsymbol{x_i}$
distance to $\boldsymbol{z_i}$

3 Comparing Distributions

Find an embedding $\boldsymbol{Z}$ minimizing the difference between all P_i, Q_i distributions



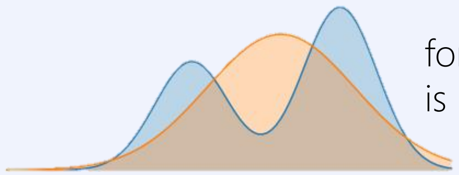
3 Comparing Distributions with KL divergence

The KL divergence is a principled way to measure the “distance” between distributions

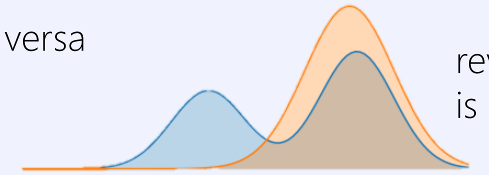
$$C_i = KL(P_i \parallel Q_i) = H(P_i, Q_i) - H(P_i) = \sum_{i \neq j} p_{j|i} \cdot \log \frac{p_{j|i}}{q_{j|i}}$$

expected #bits to encode P_i using Q_i expected #bits to encode P_i

example: given P optimize Q



forward $KL(P_i||Q_i)$
is **mean**-seeking



reverse $KL(Q_i||P_i)$
is **mode**-seeking

Properties of KL

- $KL(P_i \parallel Q_i) \geq 0$ for any P_i and Q_i
- $KL(P_i \parallel Q_i) = 0$ iff $P_i = Q_i$
- is asymmetric $KL(P_i \parallel Q_i) = KL(Q_i \parallel P_i)$
- large penalty when small $q_{j|i}$ for a large $p_{j|i}$ but not vice versa

Stochastic Neighbor Embedding (SNE) Summary

Go from distances in high-dimensional space to conditional probabilities

- $p_{j|i}$ is the probability that data point $\boldsymbol{x}_i$ "wants" data point $\boldsymbol{x}_j$ as its neighbour
- $q_{j|i}$ is the probability that transformed point $\boldsymbol{z}_i$ "wants" point $\boldsymbol{z}_j$ to be its neighbour
- variances $\boldsymbol{\sigma}_i$ are picked such that each point has "approximately the same number of neighbors"

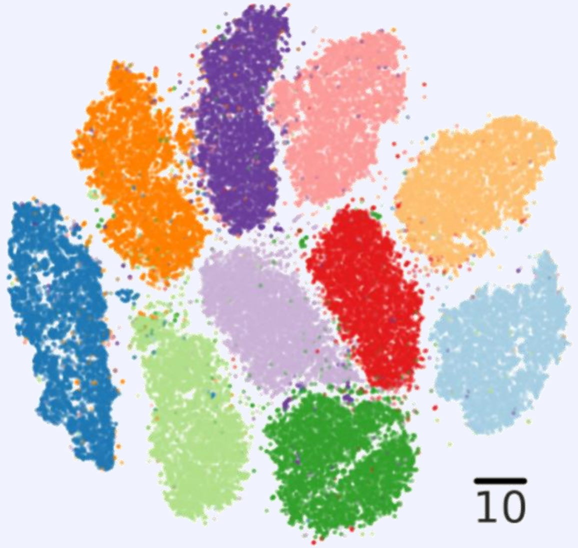
Find $\boldsymbol{z}_i$'s such that neighborhood probabilities are similar to those in original space

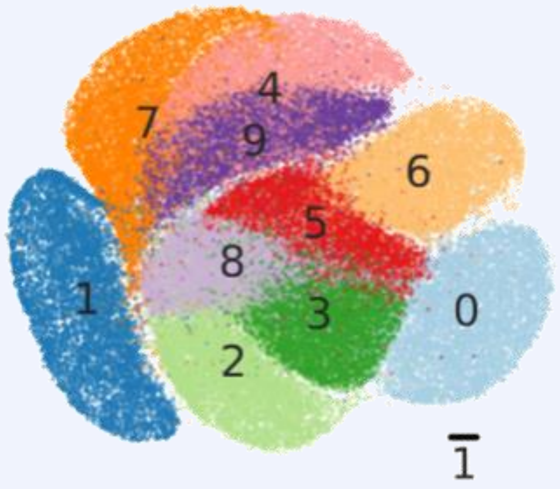
Use KL divergence to measure the "distance" between neighborhood probabilities

Use gradient descent to find z_i, $\frac{\partial C_i}{\partial z_i} = 2\sum_j \left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)(z_i - z_j)$

- can be interpreted as force based layout

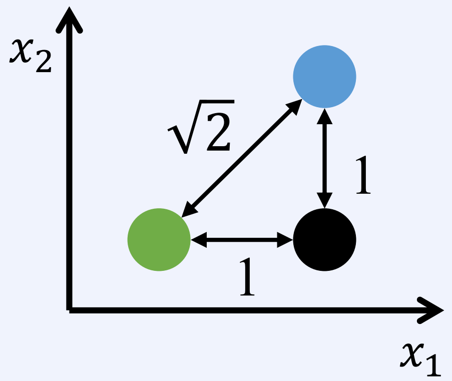
The Crowding Problem

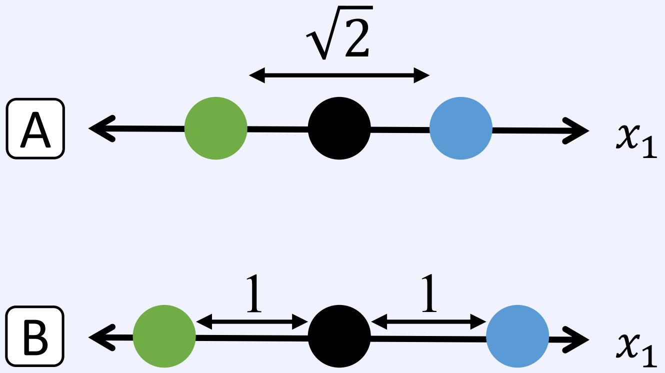




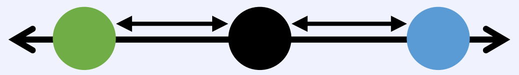
(Kobak et al. "Heavy-tailed kernels reveal a finer cluster structure in t-SNE visualisations", 2019) 36

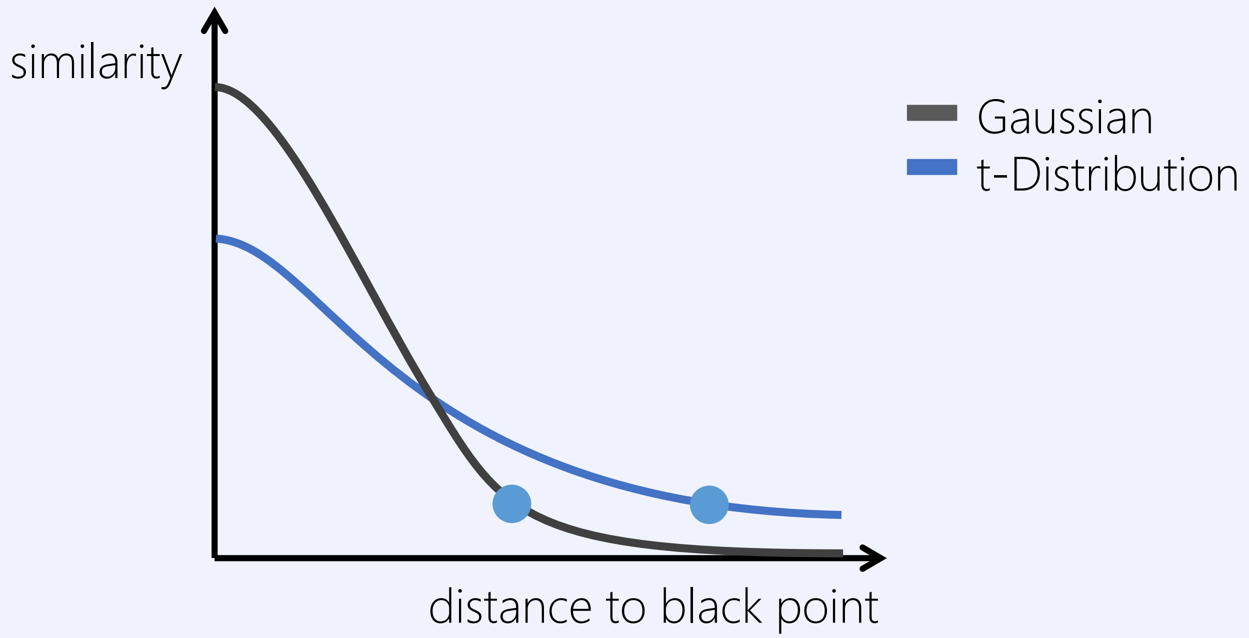
The Crowding Problem



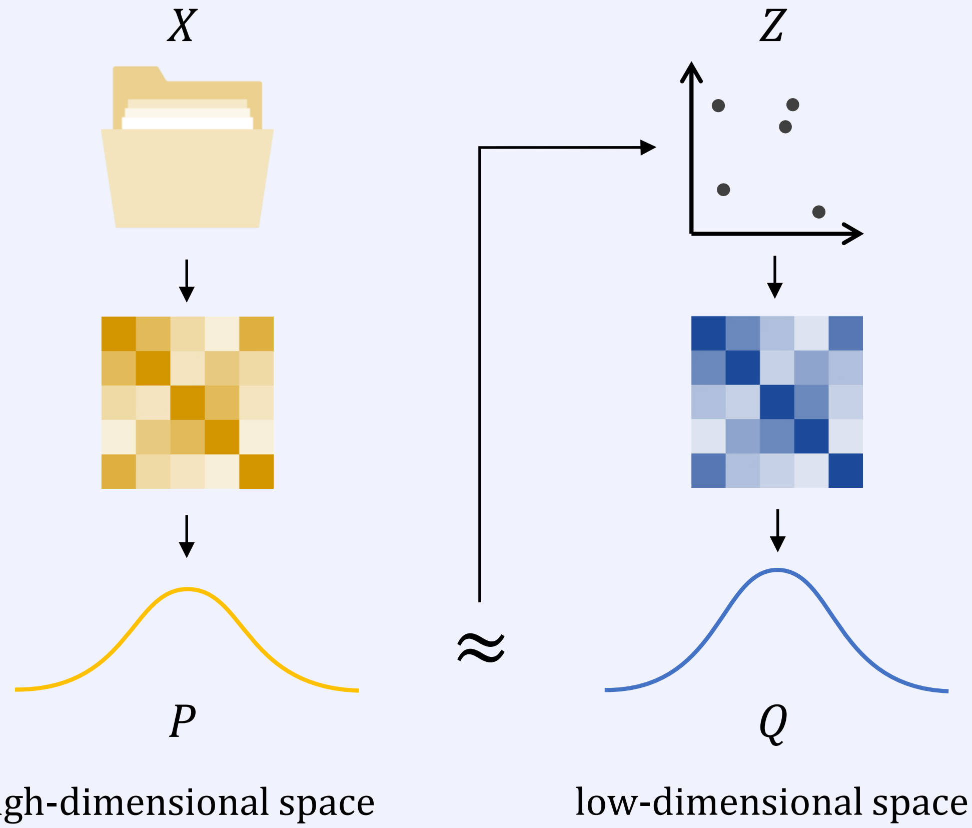


Solving the Crowding Problem



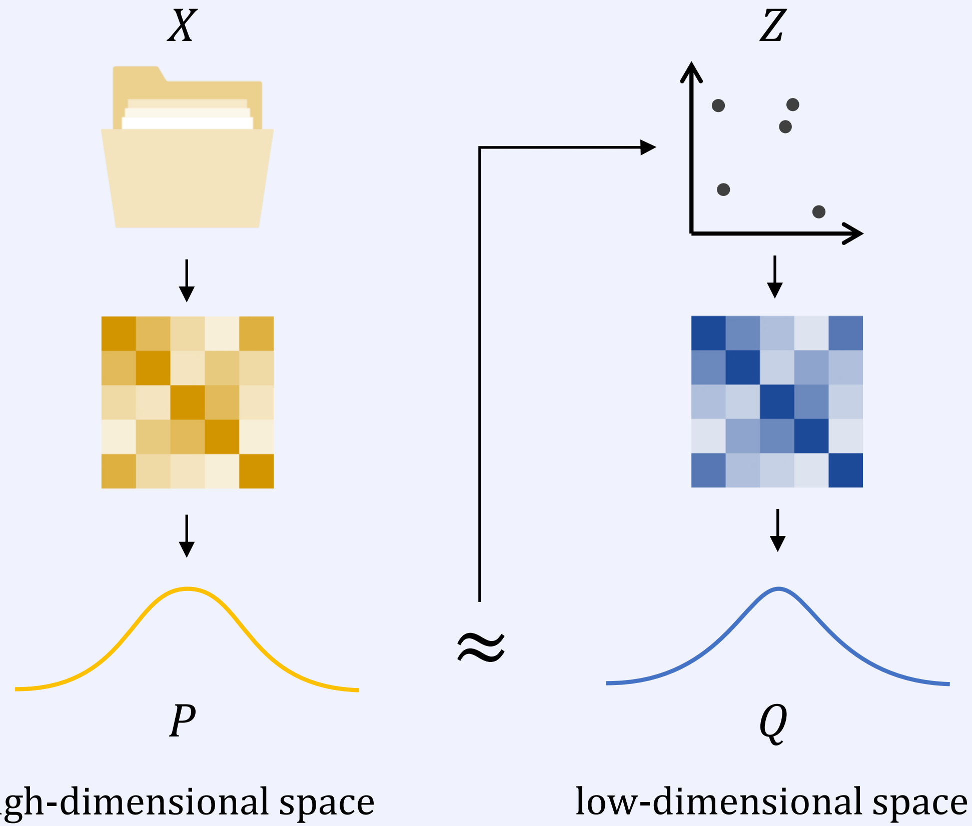


Stochastic Neighbor Embedding



(Hinton & Roweis, 2002)

t-distributed Stochastic Neighbor Embedding



(van der Maarten & Hinton, 2008)

From SNE to t-SNE

Use a **symmetric** distance function and **joint** instead of conditional probabilities

- One main P and Q by symmetrizing and normalizing $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$ and set $p_{ii} = 0$
- $C = KL(P \| Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$
- makes the optimization problem easier to solve

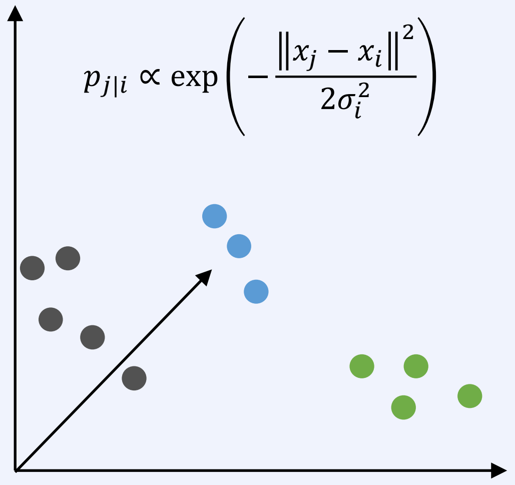
Use t-distributions for the (lower dimensional) map space $q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2)^{-1}}$

- heavier (compared to Gaussian) tail of t-distribution compensates for less space in lower dimensions
- volume of a ball scales with r^d so discrepancy in available space gets more pronounced as r grows.
- helps to avoid "crowding" effect and more faithfully reflect longer range structure

(van der Maarten & Hinton, 2008)

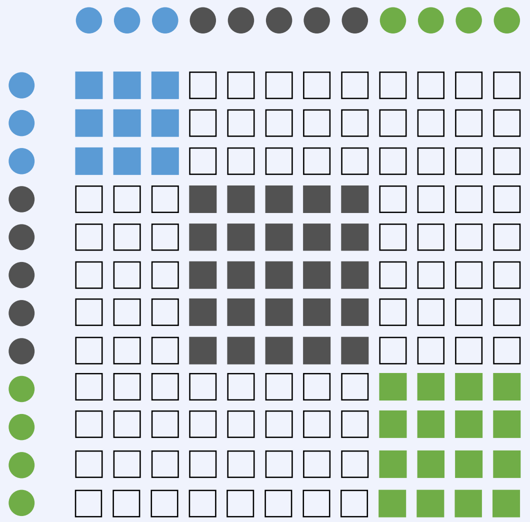
Another interpretation of SNE and t-SNE

Preserve **neighborhood** graph: low. dim neighbors as similar as possible to original neighbors



$$p_{j|i} \propto \exp\left(-\frac{\|x_j - x_i\|^2}{2\sigma_i^2}\right)$$

high dimensional data



neighbor / similarity graph

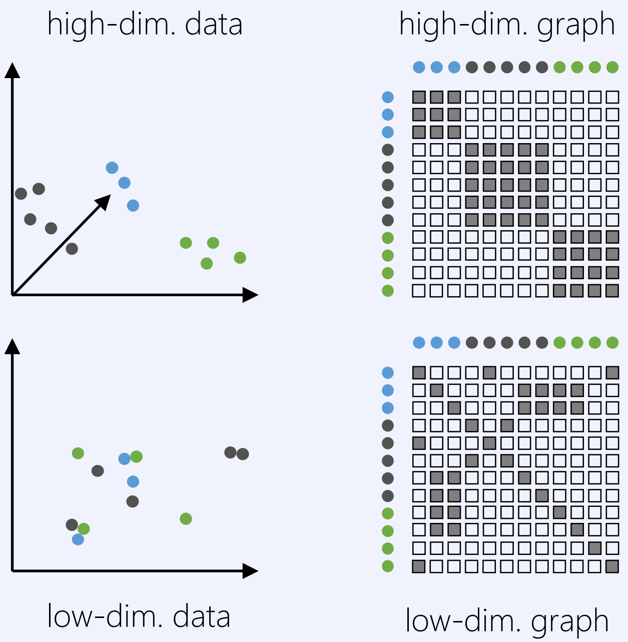
Another interpretation of SNE and t-SNE

Preserve **neighborhood** graph: low. dim neighbors as similar as possible to original neighbors

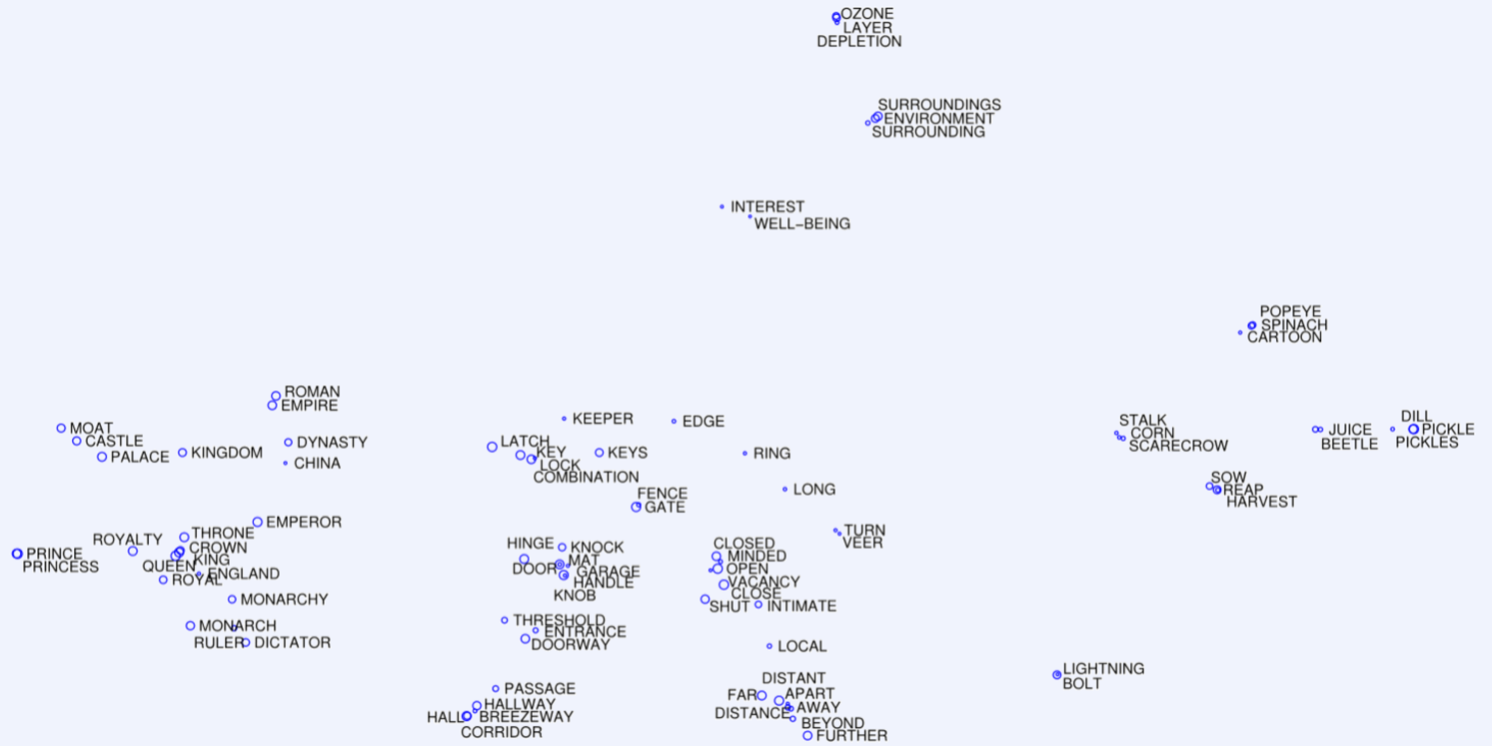
- construct neighborhood graph in high-dim. space
- initialize points in low-dim. Space
- construct neighborhood graph in low-dim space
- optimize coordinates so the two graphs look the same

Computing all pairwise distances can be very slow

Idea: (Approximately) compute only k nearest neighbors

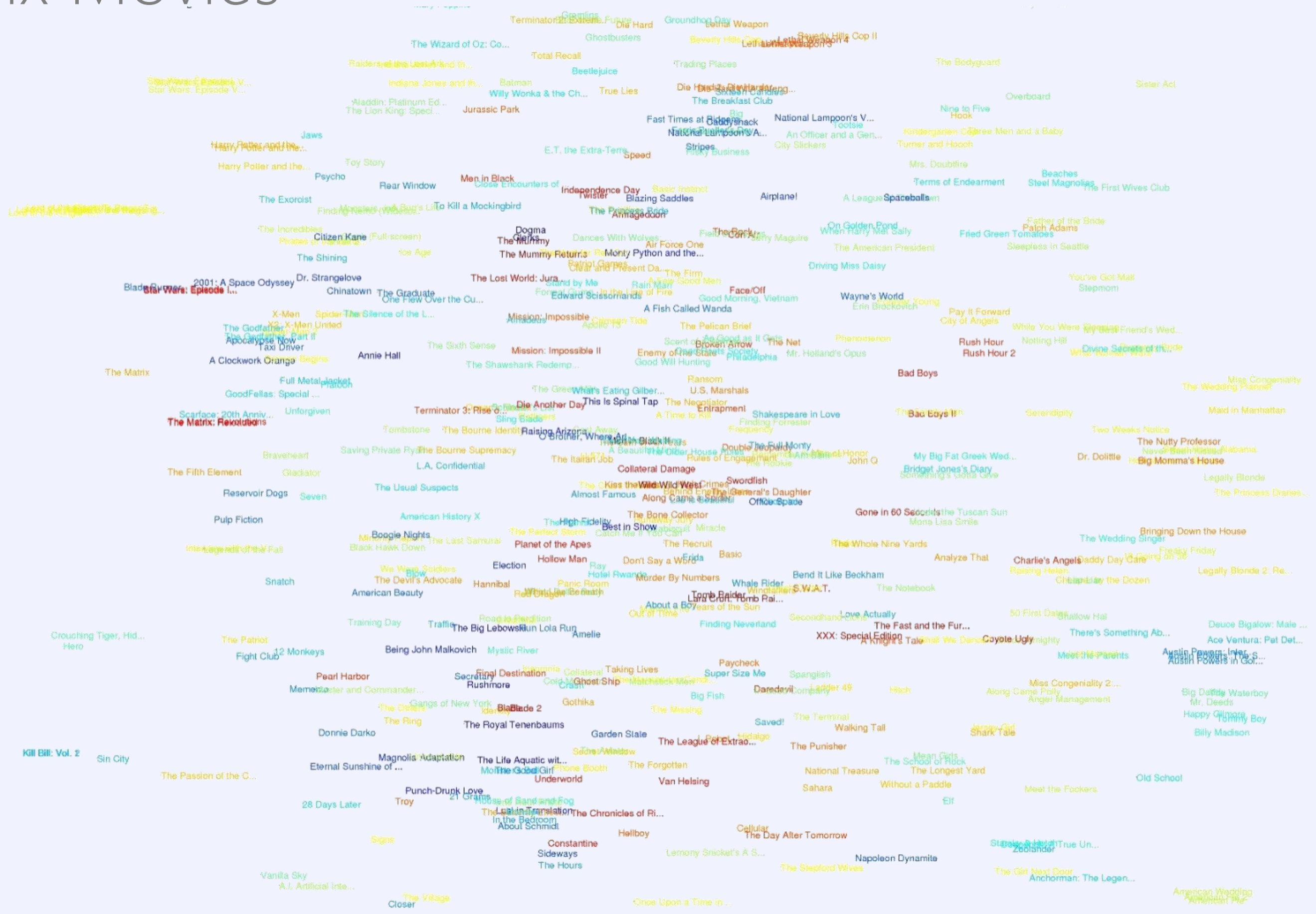


Word Association Data

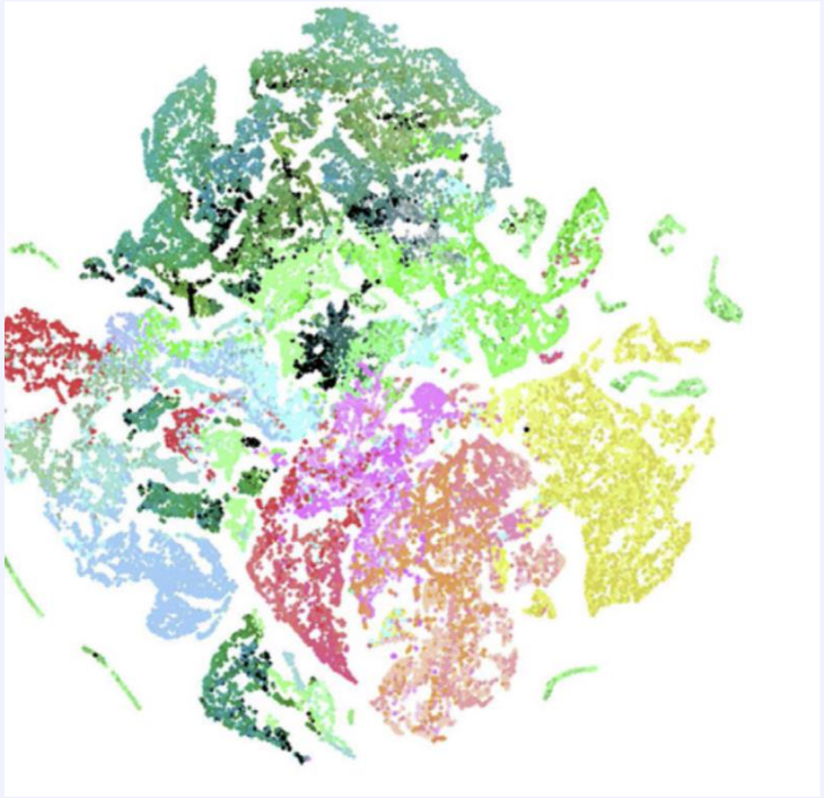


https://lvdmaaten.github.io/tsne/

Netflix Movies



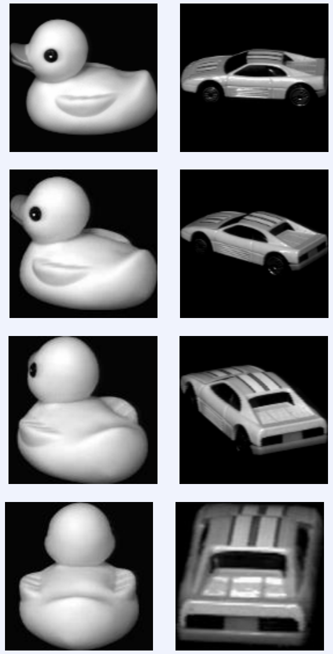
Mouse Brain Cells

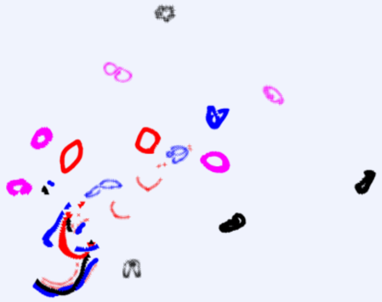


(Mahfouz et al. 2014) 46

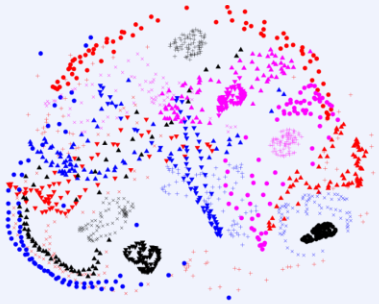
COIL-20 Object Data Set

Examples from COIL-20

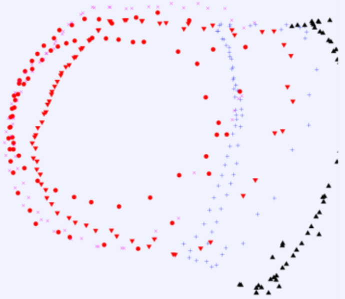




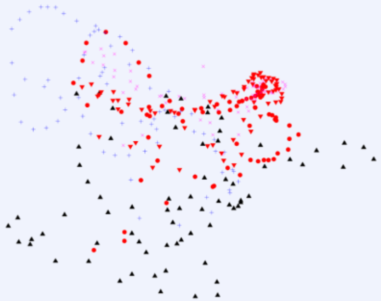
t-SNE



Sammon mapping



Isomap



LLE

Advantages and Disadvantages of t-SNE

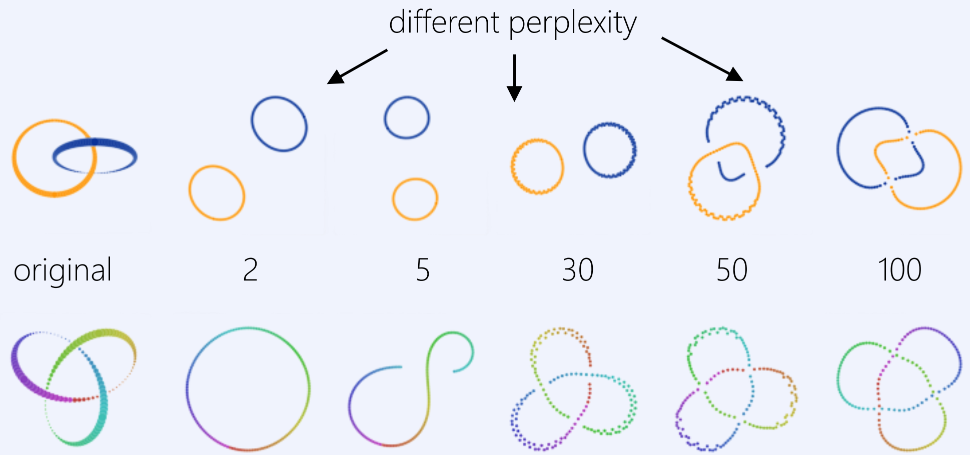
- current standard for visualizing high-dimensional data
- helps understand "black-box" algorithms like DNN
- reduced "crowding problem" with heavy tailed distribution

- t-SNE plots can sometimes be mysterious or misleading
  - Be careful with cluster sizes and cluster distances
- can be sensitive to hyperparameters
- random noise does not always look random
- no easy way to compute the embedding of new data
- not great for more than 3 dimensions

Popular alternative: **U**niform **M**anifold Approximation and **P**rojection (UMAP)

Interactive t-SNE

[Interactive widgets to better understand t-SNE.](#)



Summary

High-dimensional data is challenging in many ways

Goal of dimensionality reduction is to reduce the dimensions while preserving some structure

PCA is linear transformation that preserves the global structure
- finds the hyperplane that maximizes variance of the data / minimizes distance to projection

MDS directly preserves distances

t-SNE preserves similarity between datapoints defined by e.g. a Gaussian kernel