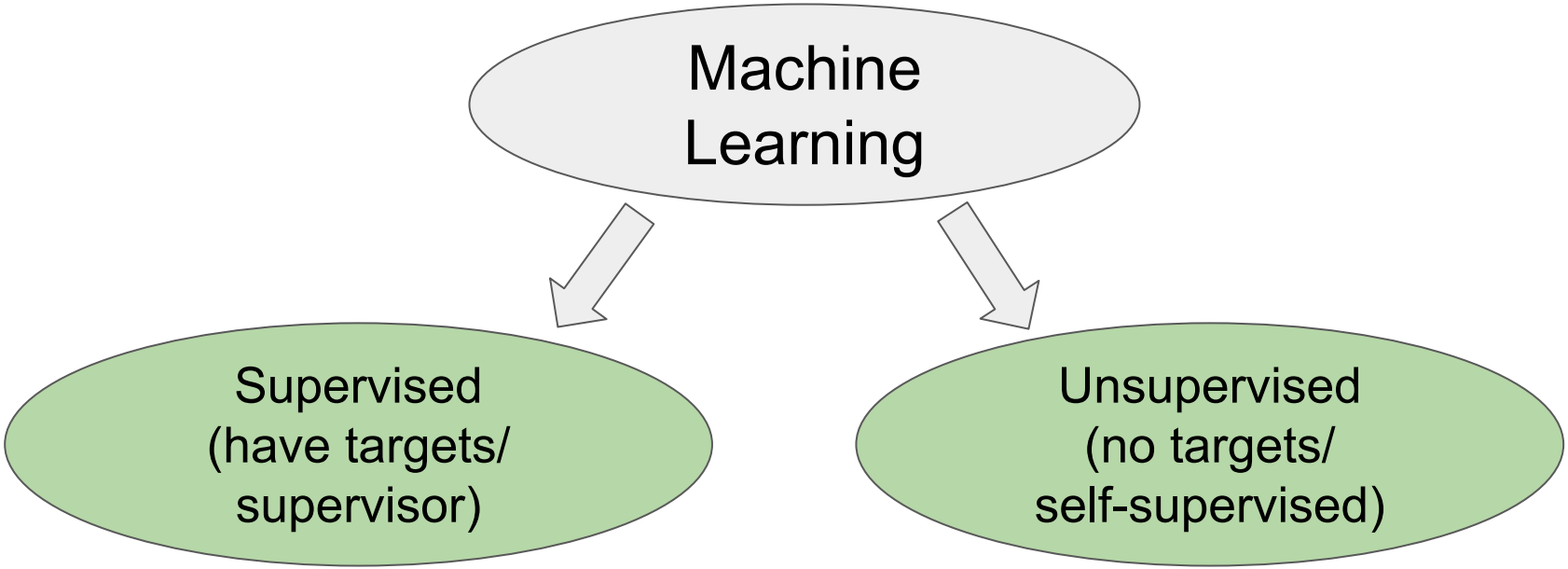
Lecture 14: Unsupervised learning

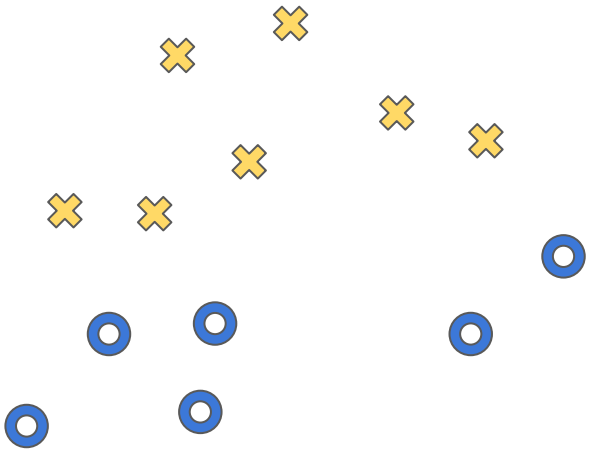
Ivan Provilkov

Plan

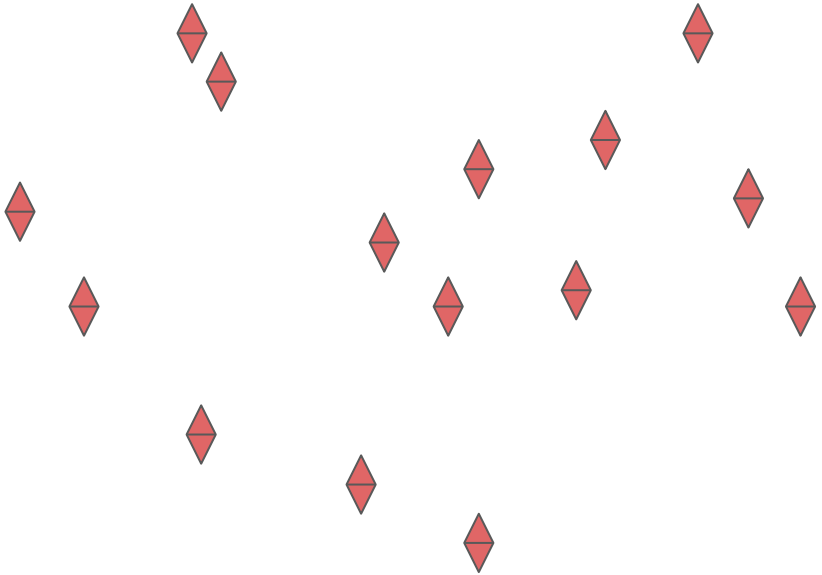
- Dimensionality reduction
  - Linear model
  - PCA
  - t-SNE
- Clustering
  - K-Means
  - DBSCAN


Machine Learning
Supervised (have targets/ supervisor)
Unsupervised (no targets/ self-supervised)

Supervised learning



Unsupervised learning



Dimensionality reduction

Dimensionality reduction

The data lies approximately on a manifold of much lower dimension than the input space

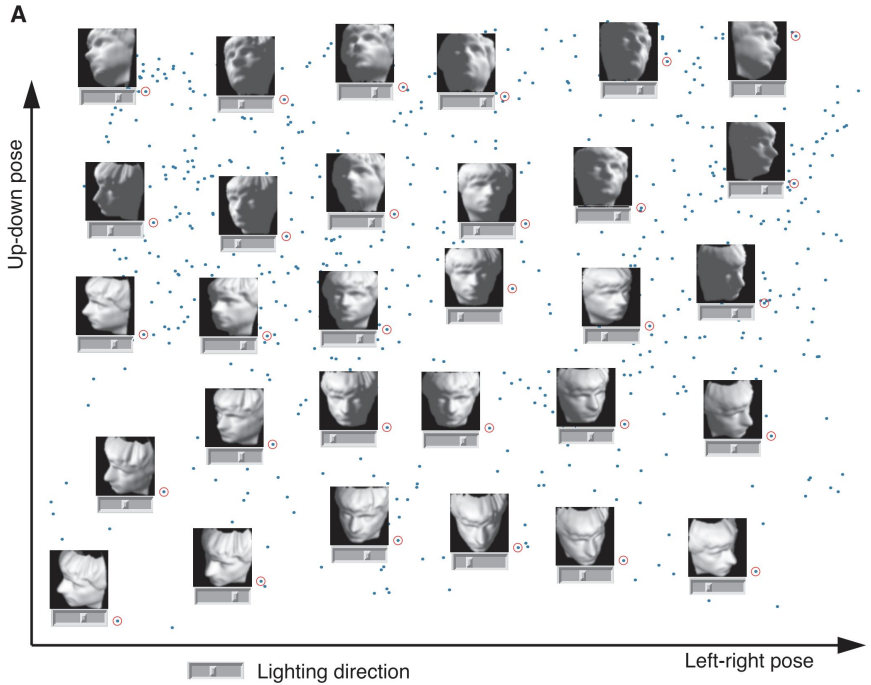


Image source: Tenenbaum, de Silva, Langford A Global Geometric Framework for Nonlinear Dimensionality Reduction

Why do we need dimensionality reduction

Why do we need dimensionality reduction

- Multicollinearity

Why do we need dimensionality reduction

- Multicollinearity
- Dimensionality curse

Why do we need dimensionality reduction

- Multicollinearity
- Dimensionality curse
- Interpretability

Decrease the dimensionality using linear methods

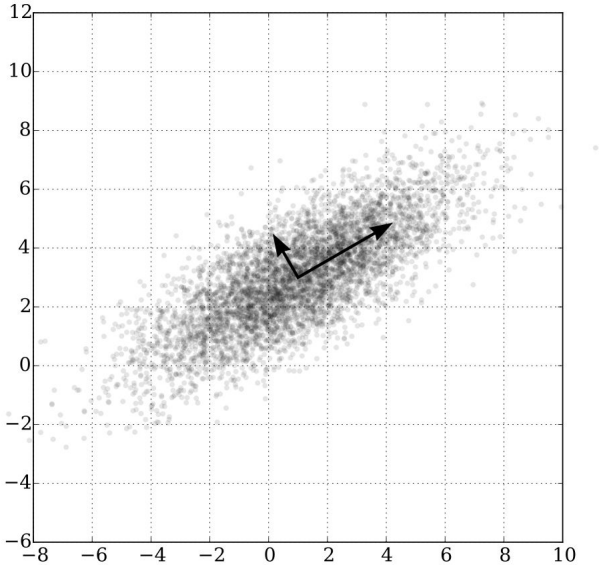
Decrease the dimensionality using linear methods

$$f(x,\alpha) = \sum_{j=1}^{n} \alpha_j f_j(x)$$

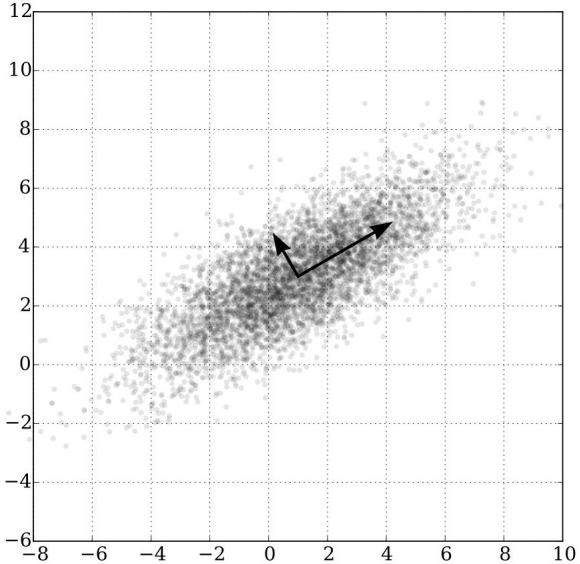
$$\underset{\ell\times n}{F} = \begin{pmatrix} f_1(x_1) & \dots & f_n(x_1) \\ \dots & \dots & \dots \\ f_1(x_\ell) & \dots & f_n(x_\ell) \end{pmatrix}, \quad \underset{\ell\times 1}{y} = \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix}, \quad \underset{n\times 1}{\alpha} = \begin{pmatrix} \alpha_1 \\ \dots \\ \alpha_n \end{pmatrix}$$

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} \big(f(x_i,\alpha) - y_i\big)^2 = \|F\alpha - y\|^2 \to \min_{\alpha}$$

PCA



PCA

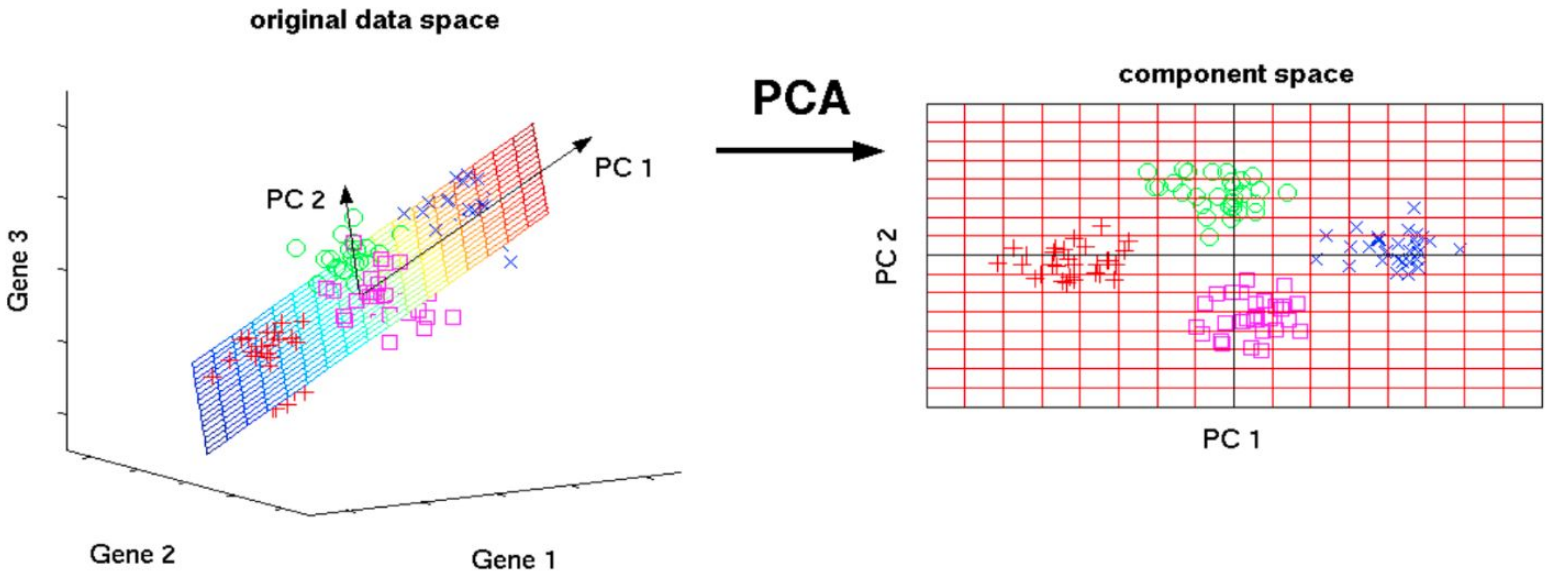


$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \{\|\mathbf{X}\mathbf{w}\|^2\} = \arg\max_{\|\mathbf{w}\|=1} \left\{\mathbf{w}^T\mathbf{X}^\mathbf{T}\mathbf{X}\mathbf{w}\right\}$$

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X}\mathbf{w}_{(s)}\mathbf{w}_{(s)}^\mathrm{T}$$

$$\mathbf{w}_{(k)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \|\hat{\mathbf{X}}_k\mathbf{w}\|^2 \right\} = \arg\max \left\{ \frac{\mathbf{w}^T\hat{\mathbf{X}}_k^T\hat{\mathbf{X}}_k\mathbf{w}}{\mathbf{w}^T\mathbf{w}} \right\}$$

PCA

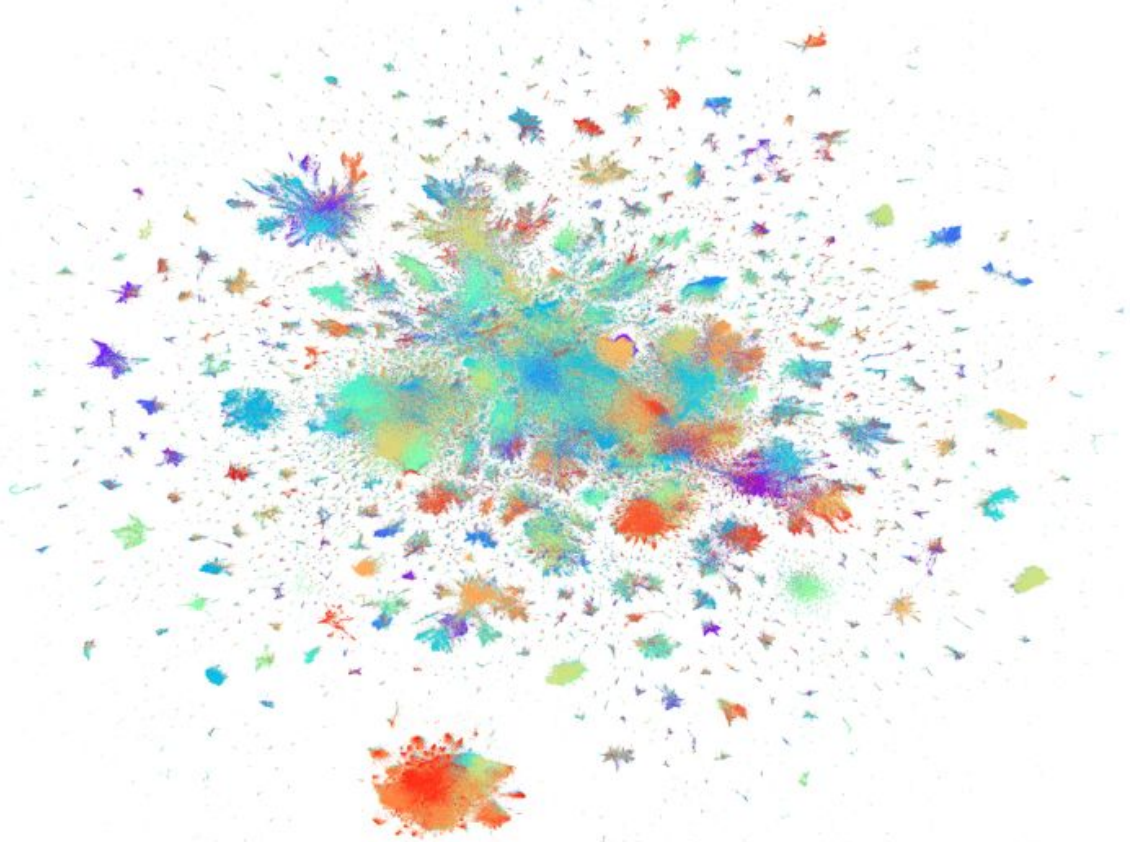

original data space
PC 1
PC 2
Gene 3
Gene 2
Gene 1
PCA
component space
PC 2
PC 1

PCA

- Works bad with nonlinear clusters
- Works bad as 2d projection for high-dimensional data

t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE makes every slide 42% better (c)



Source: [Habrahabr post on t-SNE]

t-SNE

- Nonlinear dimensionality reduction
- Good for visualization

t-SNE

Idea: Convert pairwise distances to probabilities, preserve probabilities through the spaces

t-SNE

Idea: Convert pairwise distances to probabilities, preserve probabilities through the spaces

$$p_{j|i} = \frac{\exp(-\frac{||x_i - x_j||^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{||x_i - x_k||^2}{2\sigma_i^2})}$$

asymmetric probability of object i chooses j as its neighbour

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

t-SNE

In target (2-dimensional) space, Student-t distribution:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

Idea: construct embedding s.t. this distributions are close.
What are close distributions?

Kullback–Leibler divergence

$$D_{KL}(P \parallel Q) = \sum_{i,j} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Seems similar to cross-entropy, doesn’t it?

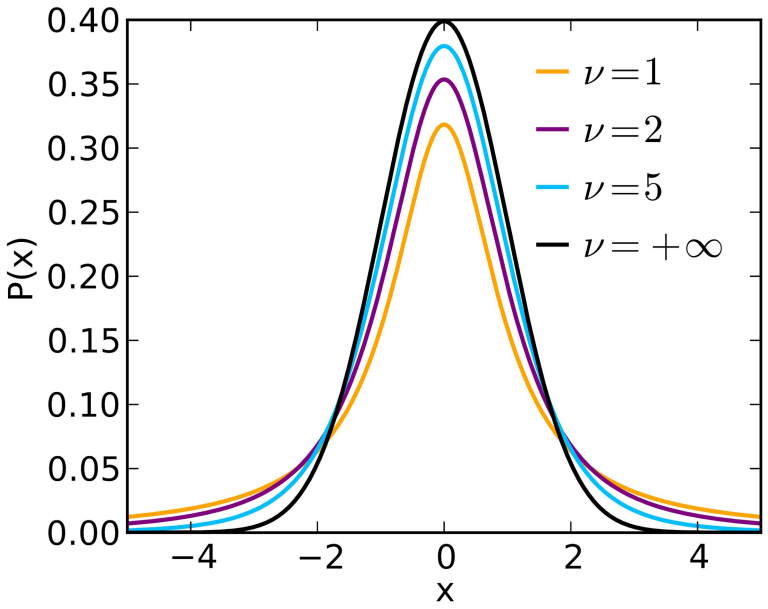
t-distributed Stochastic Neighbor Embedding

How to improve it:

1. make distribution symmetric
2. make it decrease faster than Gaussian
   (use Student's t-distribution)

$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2 / 2\sigma^2\right)}$$

$$q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)}$$

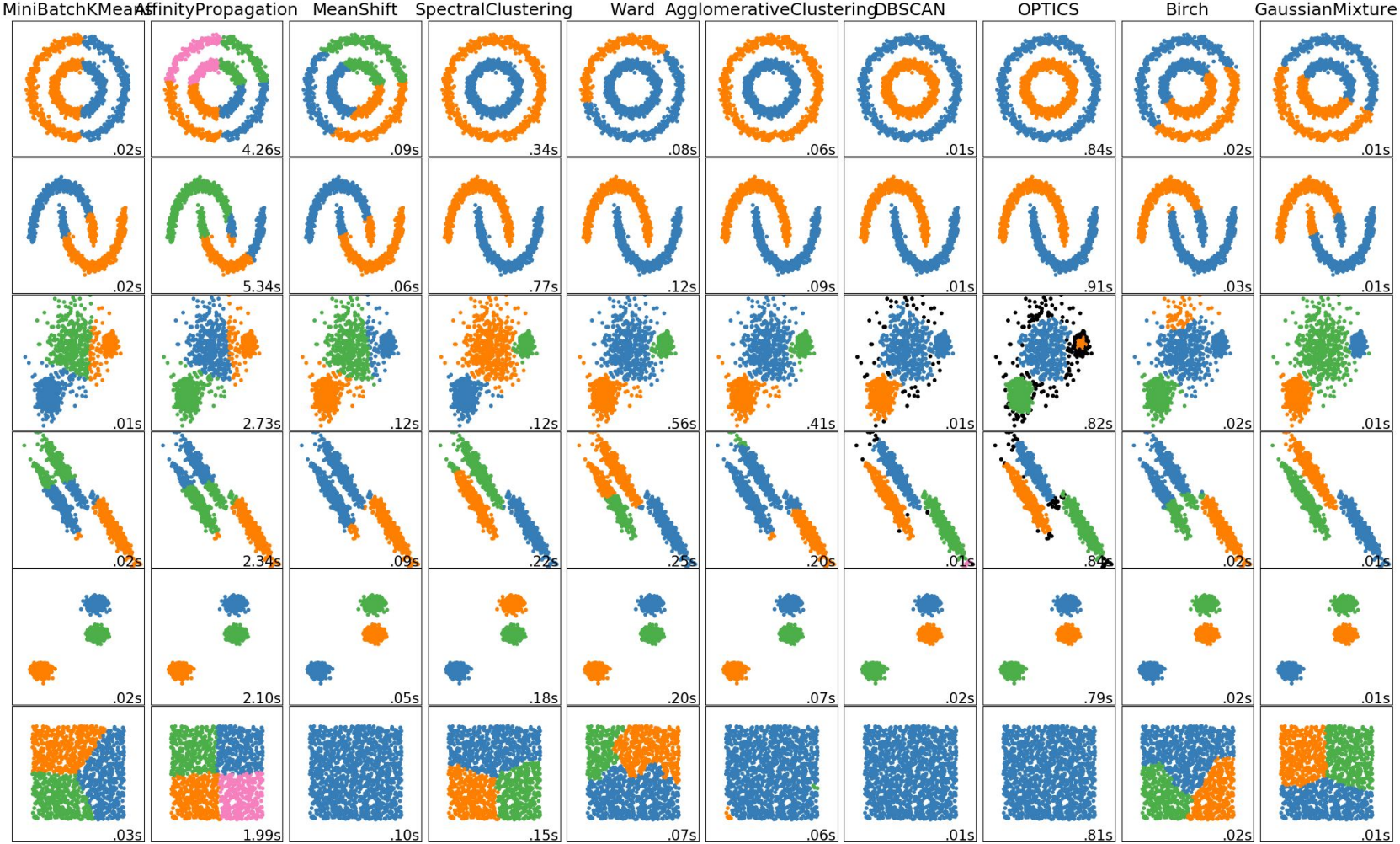


Link to original paper

Clustering

Clustering

- Want to find clusters in the data
- Many different metrics
- Number of clusters usually is not known


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
OPTICS
Birch
GaussianMixture
.02s
4.26s
.09s
.34s
.08s
.06s
.01s
.84s
.02s
.01s
.02s
5.34s
.06s
.77s
.12s
.09s
.01s
.91s
.03s
.01s
.01s
2.73s
.12s
.12s
.56s
.41s
.01s
.82s
.02s
.01s
.02s
2.34s
.09s
.22s
.25s
.20s
.01s
.84s
.02s
.01s
.02s
2.10s
.05s
.18s
.20s
.07s
.02s
.79s
.02s
.01s
.03s
1.99s
.10s
.15s
.07s
.06s
.01s
.81s
.02s
.01s

Clustering

There are many methods:

- Graph based
- Hierarchical
- Statistical
- Others (ex. neural)

Clustering

Quality measures examples:

- Mean in-cluster distance

$$F_0 = \frac{\sum\limits_{i<j} [y_i = y_j]\, \rho(x_i, x_j)}{\sum\limits_{i<j} [y_i = y_j]} \to \min$$

- Mean inter-cluster distance

$$F_1 = \frac{\sum\limits_{i<j} [y_i \neq y_j]\, \rho(x_i, x_j)}{\sum\limits_{i<j} [y_i \neq y_j]}$$

K-Means

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* (≤ *n*) sets $\mathbf{S} = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

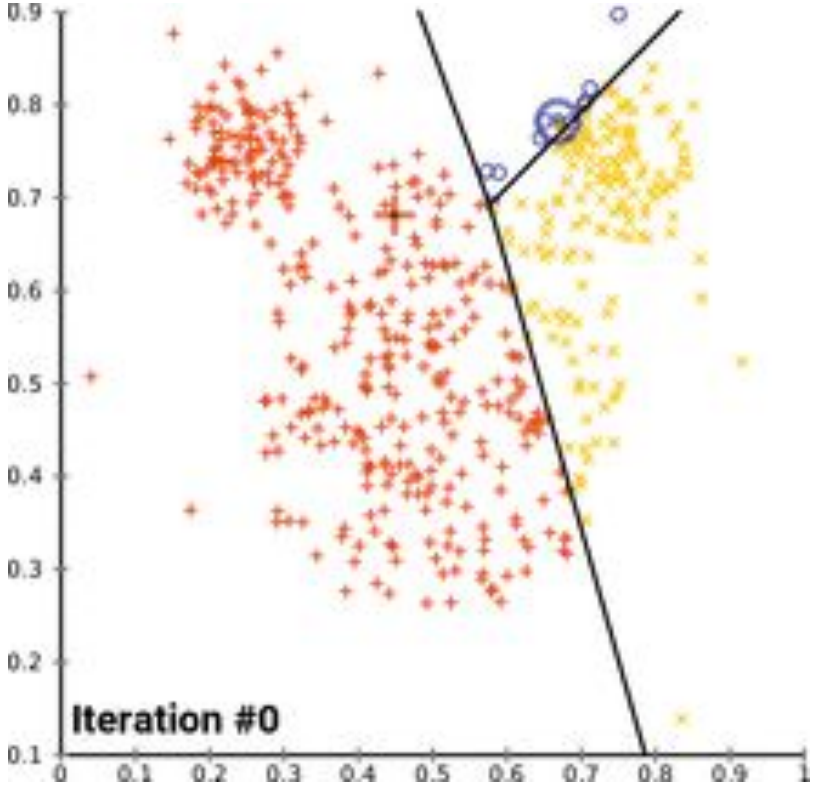
K-Means

Training:

- Repeat:
  - Assign each object to the nearest cluster (by mean of cluster members)
  - Recalculate means of clusters

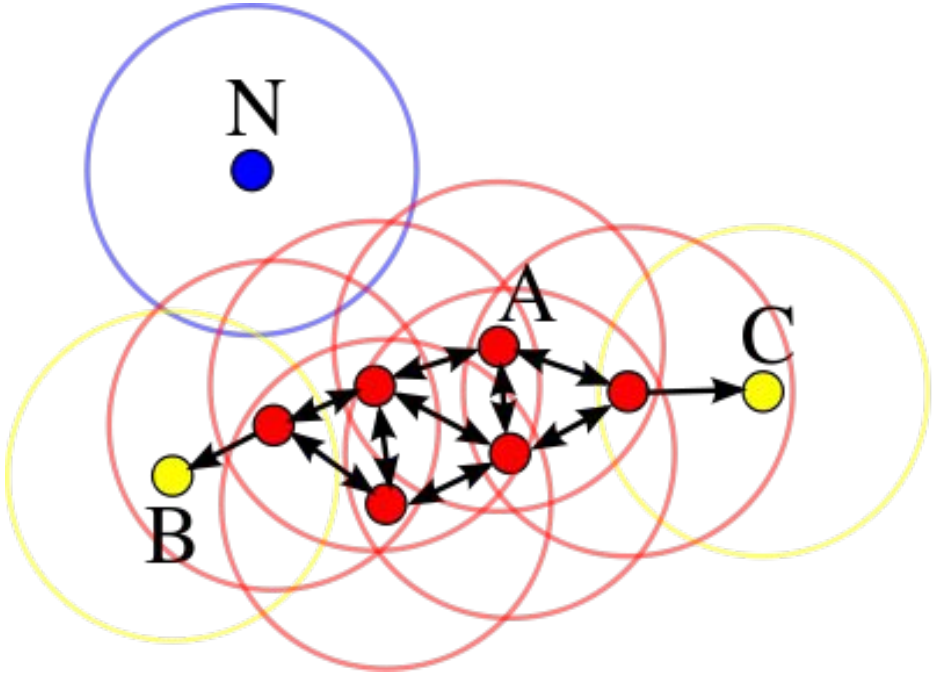
$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

K-Means



DBSCAN

Density-based spatial clustering of applications with noise



DBSCAN

Density-based spatial clustering of applications with noise

- A point p is a *core point* if at least minPts points are within distance ε of it (including p).
- A point q is *directly reachable* from p if point q is within distance ε from core point p. Points are only said to be directly reachable from core points.
- A point q is *reachable* from p if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i. Note that this implies that the initial point and all points on the path must be core points, with the possible exception of q.
- All points not reachable from any other point are *outliers* or *noise points*.

DBSCAN

Density-based spatial clustering of applications with noise

1. Find the points in the ε (eps) neighborhood of every point, and identify the core points with more than minPts neighbors.
2. Find the connected components of *core* points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor, otherwise assign it to noise.