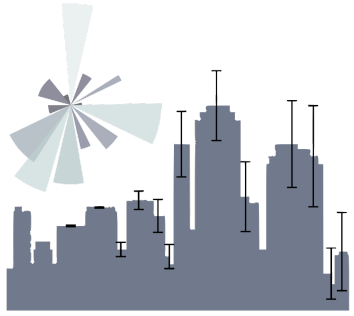
principles of Urban Science 10



unsupervised learning

dr.federica bianco | *fbb.space* | *fedhere* | *fedhere*

this slide deck:

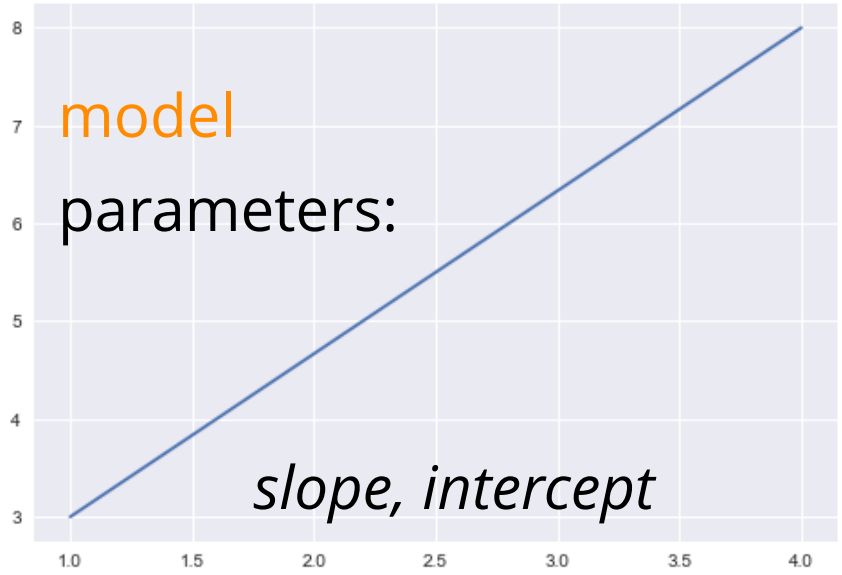
slides.com/federicabianco/pus2020_10

- **Machine Learning basic concepts**
  - interpretability
  - parameters vs hyperparameters
  - supervised/unsupervised
- **Clustering : how it works**
  - **Partitioning**
    - **Hard clustering**
    - **Soft Clustering**
  - **Hirarchical**
    - **agglomerative**
    - **devisive**
  - **also:**
    - **Density based**
    - **Grid based**
    - **Model based**

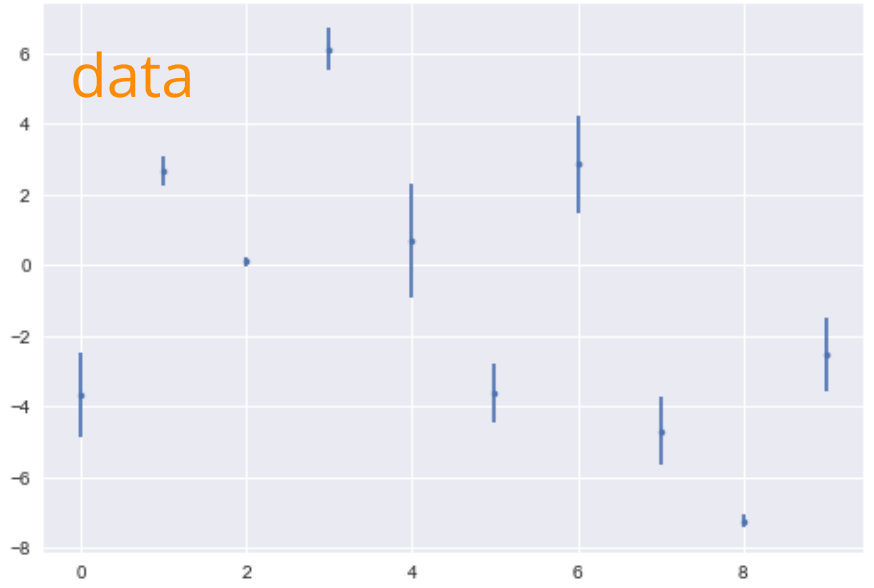
what is machine learning

what is machine learning?

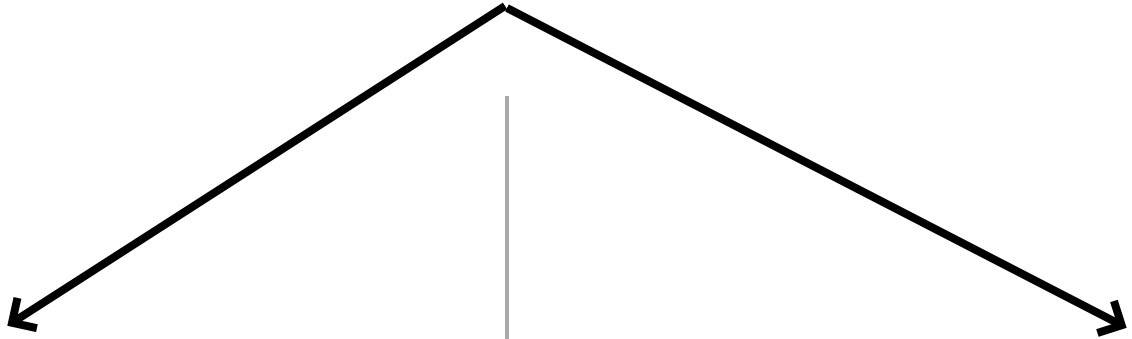
[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed. Arthur Samuel, 1959



ML: any model with parameters learnt from the data



what is machine learning?



supervised learning

classification

prediction

feature selection

unsupervised learning

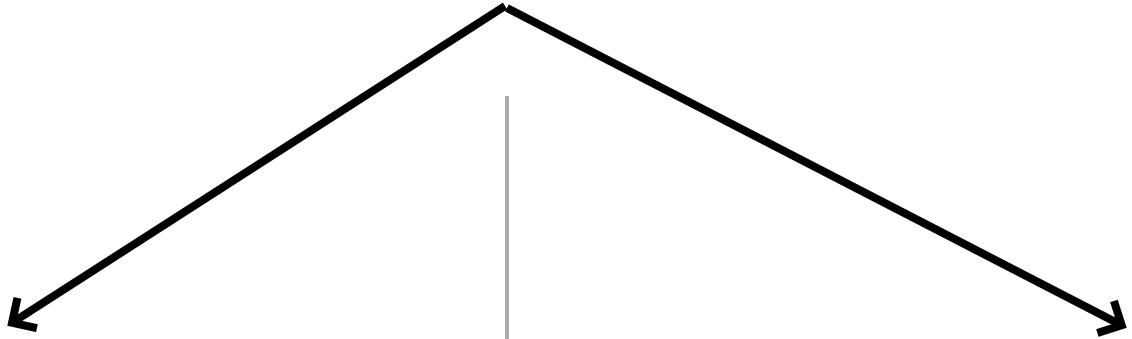
understanding structure

organizing/compressing data

anomaly detection

dimensionality reduction

what is machine learning?



supervised learning

k-Nearest Neighbors

Regression

Support Vector Machines

Classification/Regression Trees

Neural networks

classification

prediction

feature selection

unsupervised learning

understanding structure

organizing/compressing data

anomaly detection

dimensionality reduction

clustering

PCA

Apriori

general ML points

used to:

understand structure of feature space

classify based on examples,

regression (classification with infinitely small classes)

general ML points

should be interpretable: why?

ethical implications

predictive policing,

selection of conference participants.

general ML points

should be interpretable: why?

ethical implications

predictive policing,

selection of conference participants.

causal connection

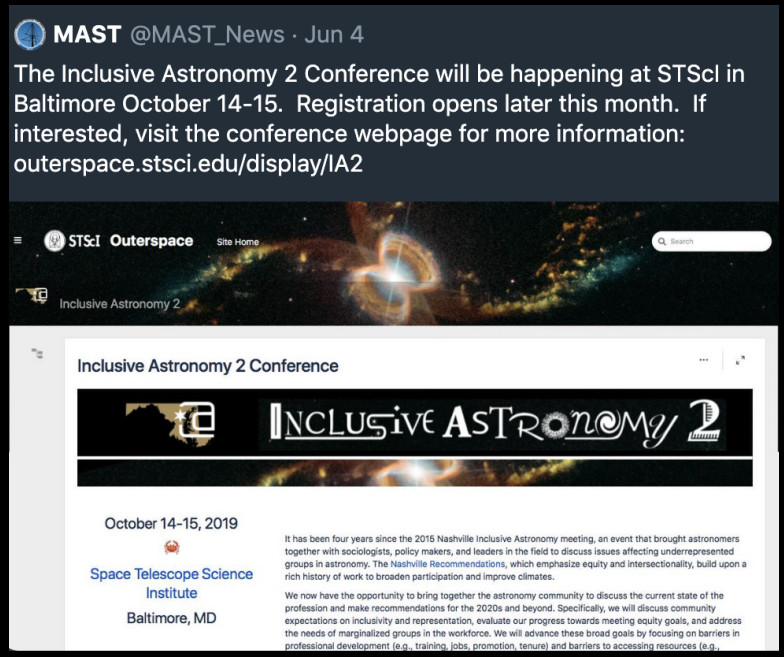
why the model made a choice?

which feature mattered?

general ML points

should be interpretable:

ethical implications



general ML points

ML model have *parameters* and *hyperparameters*

parameters: the model optimizes based on the data

hyperparameters: chosen by the model author, could be based on domain knowledge, other data, guessed (?!).

e.g. the shape of the polynomial

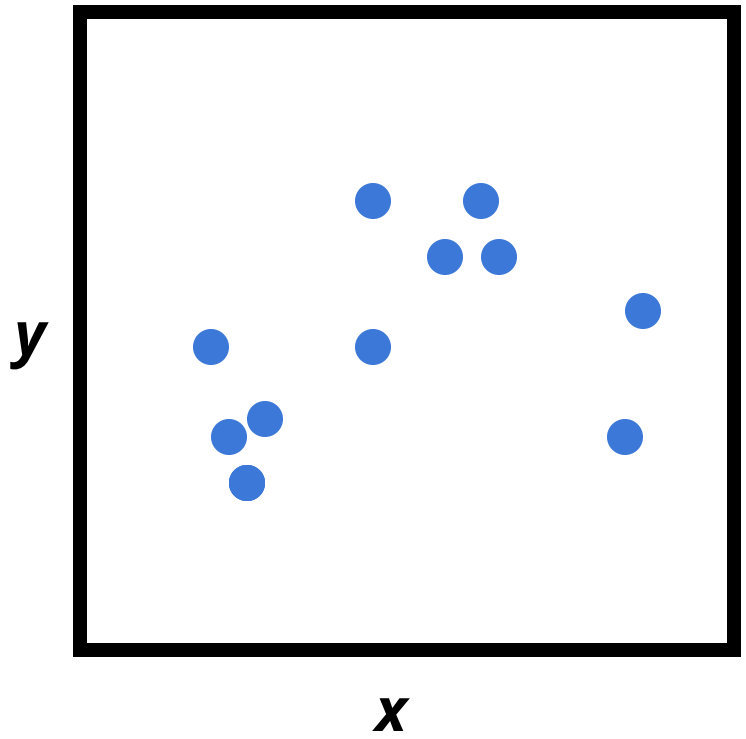
1 clustering

clustering is an unsupervised learning method

GOAL: partitioning data in *maximally homogeneous, maximally distinguished* subsets.

observed **features:**

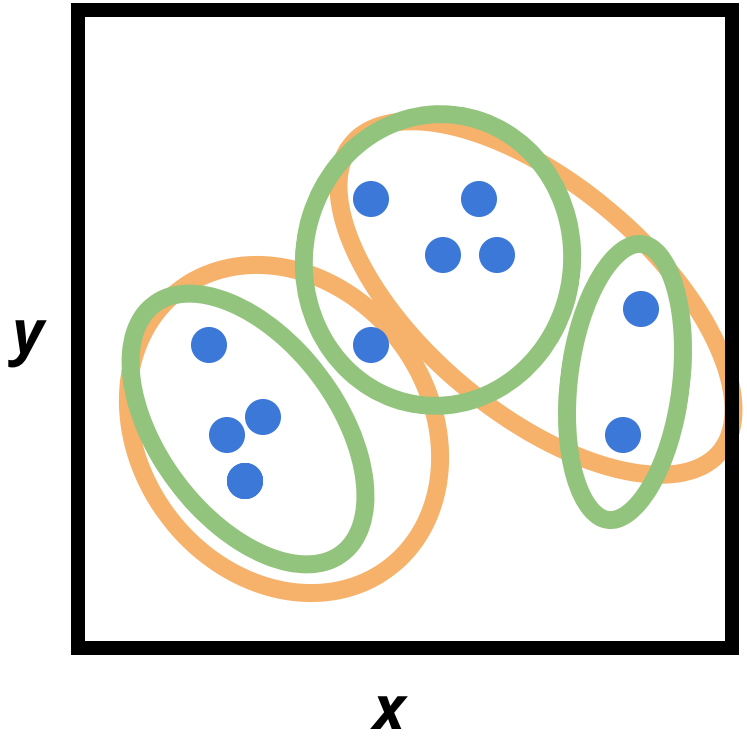
$(\vec{x}, \vec{y})$



clustering is an unsupervised learning method

all features are observed for all objects in the sample

$(\vec{x}, \vec{y})$



how should I group the observations in this feature space?

e.g.: how many groups should I make?

optimal clustering

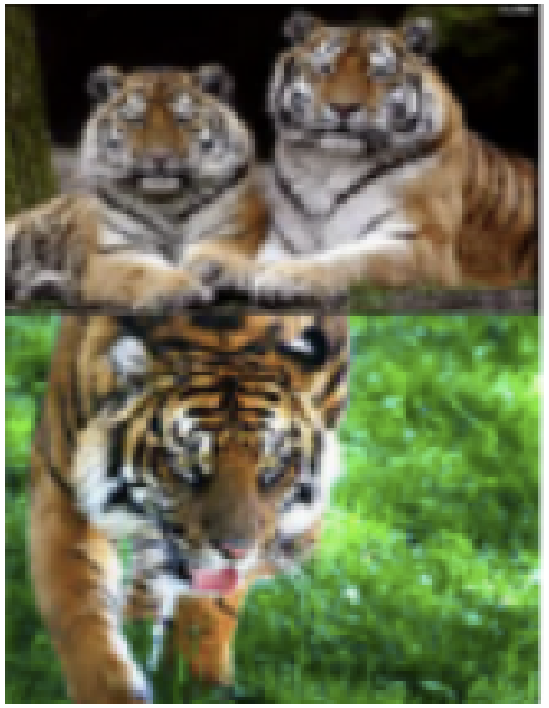
http://www-bcf.usc.edu/~soltanol/Applications.html

internal criterion:

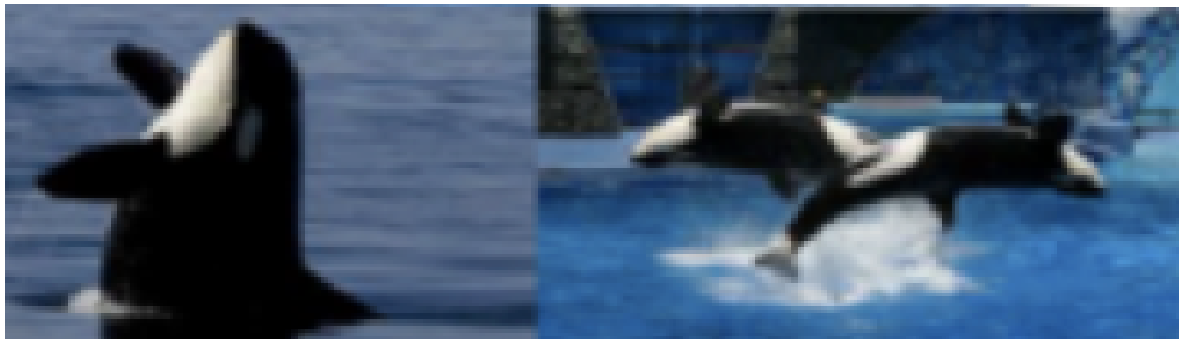
members of the cluster should be similar to each other (intra cluster compactness)

external criterion:

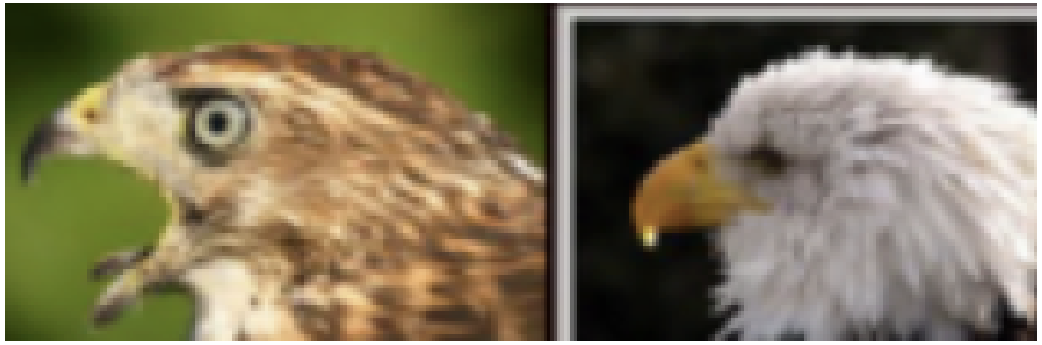
objects outside the cluster should be dissimilar from the objects inside the cluster



tigers



wales



raptors

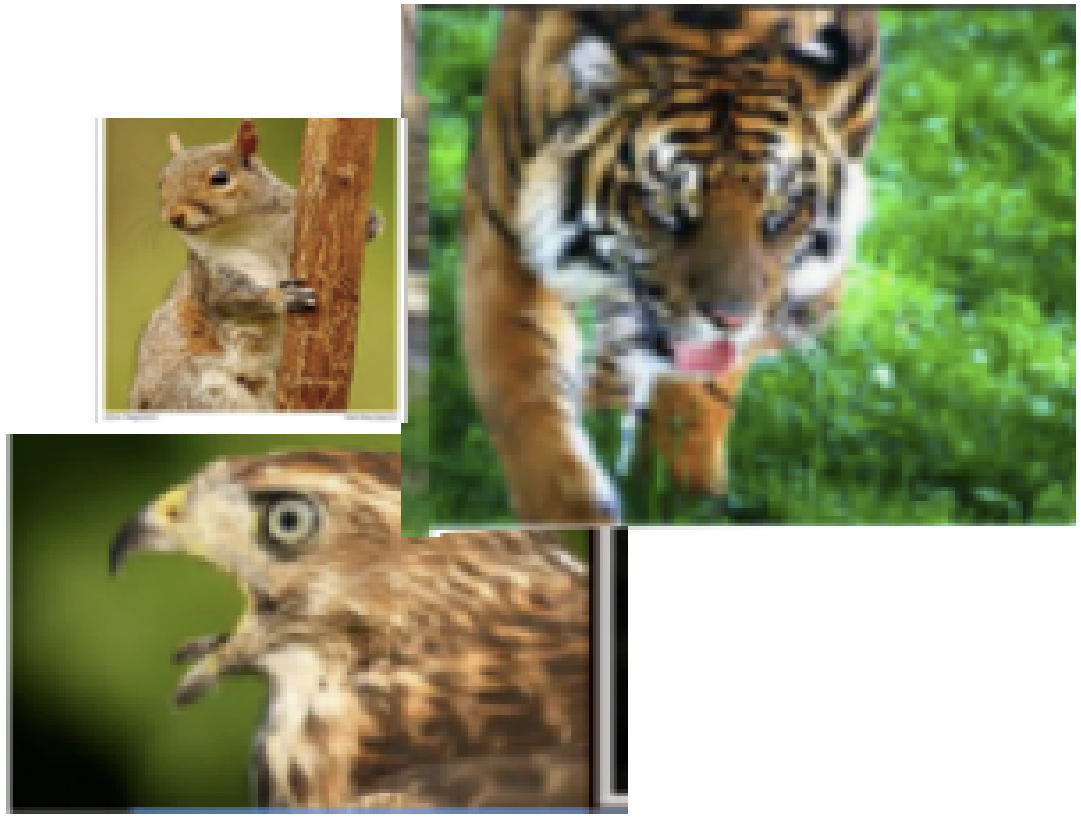
zoologist's clusters

internal criterion:

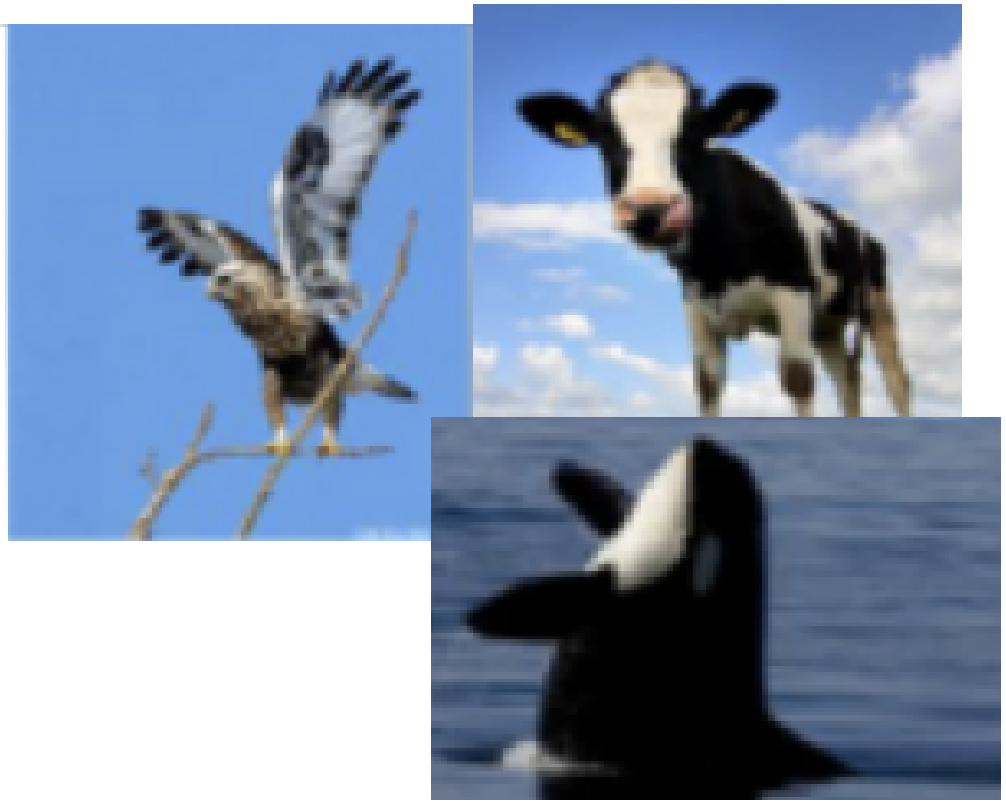
members of the cluster should be similar to each other (intra cluster compactness)

external criterion:

objects outside the cluster should be dissimilar from the objects inside the cluster



orange/green



black/white/blue

photographer's clusters

internal criterion:

members of the cluster should be similar to each other (intra cluster compactness)

external criterion:

objects outside the cluster should be dissimilar from the objects inside the cluster

the optimal clustering depends on:

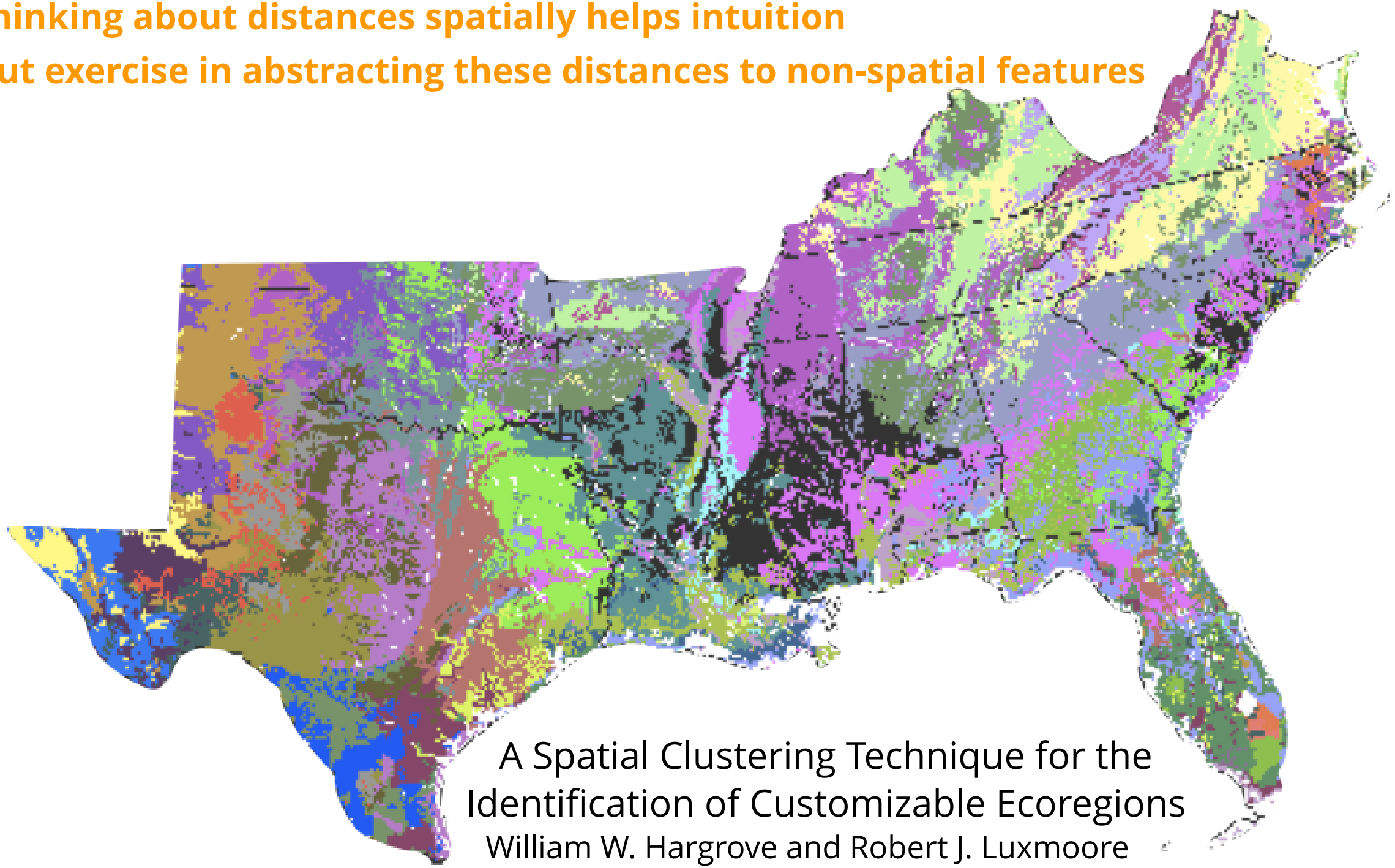
- how you define similarity/distance
- the purpose of the clustering

the ideal clustering algorithm should have:

- Scalability (naive algorithms are Np hard)
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shapes
- Minimal requirement for domain knowledge
- Deals with noise and outliers
- Insensitive to order
- Allows incorporation of constraints
- Interpretable

thinking about distances spatially helps intuition

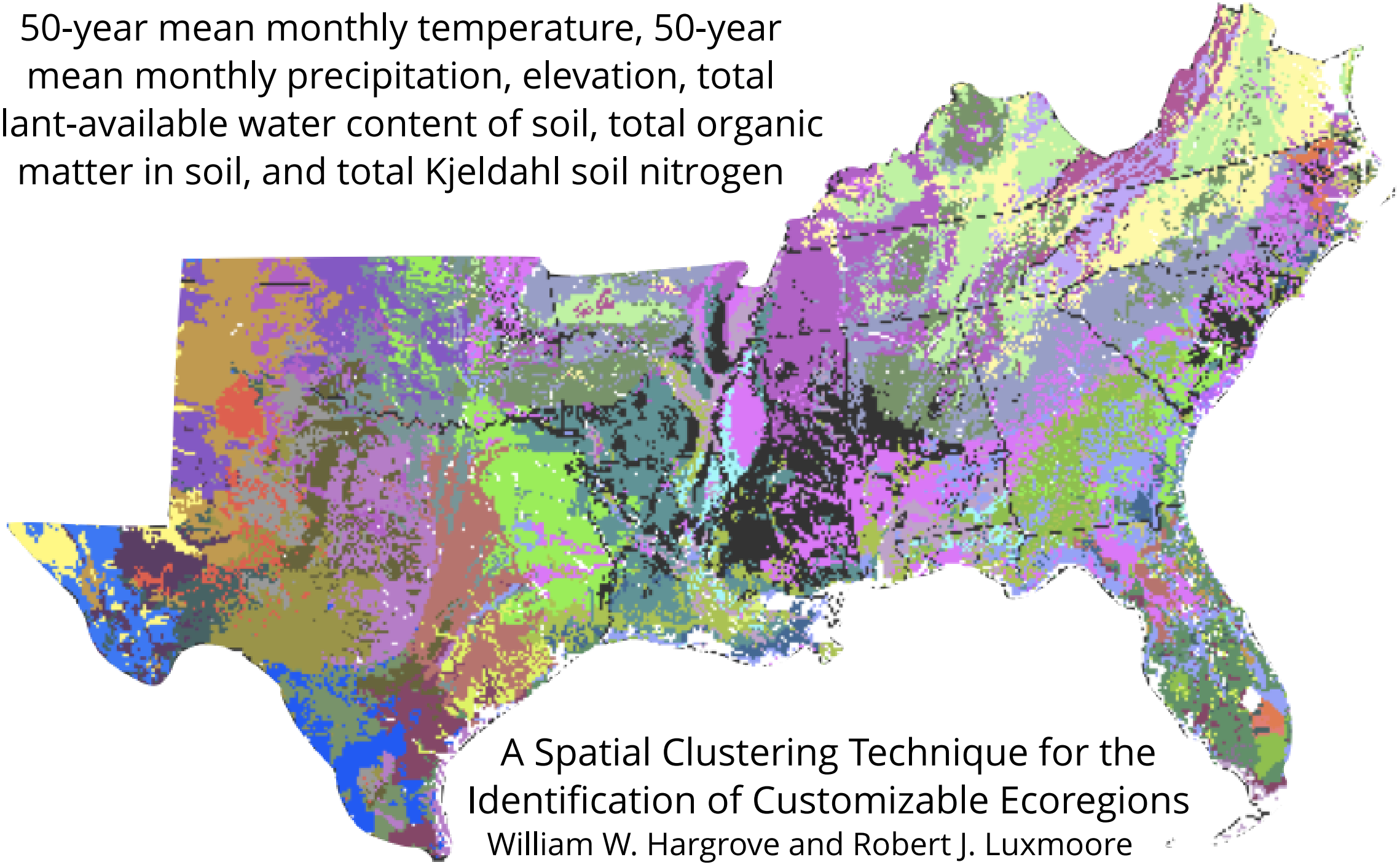
but exercise in abstracting these distances to non-spatial features



A Spatial Clustering Technique for the Identification of Customizable Ecoregions

William W. Hargrove and Robert J. Luxmoore

50-year mean monthly temperature, 50-year mean monthly precipitation, elevation, total plant-available water content of soil, total organic matter in soil, and total Kjeldahl soil nitrogen



A Spatial Clustering Technique for the Identification of Customizable Ecoregions

William W. Hargrove and Robert J. Luxmoore

distance metrics

2

distance metrics
continuous variables

Minkowski family of distances

$$D(i,j) = \sqrt[p]{|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \ldots + |x_{iN} - x_{jN}|^p}$$

distance metrics continuous variables

Minkowski family of distances

$$D(i,j) = \sqrt[p]{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^p}$$

N features (dimensions)

distance metrics

continuous variables

Minkowski family of distances

$$D(i,j) = \sqrt[p]{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^p}$$

N features (dimensions)

properties:

$$D(i,j) > 0$$

$$D(i,i) = 0$$

$$D(i,j) = D(j,i)$$

$$D(i,j) <= D(i,k) + D(k,j)$$

distance metrics

continuous variables

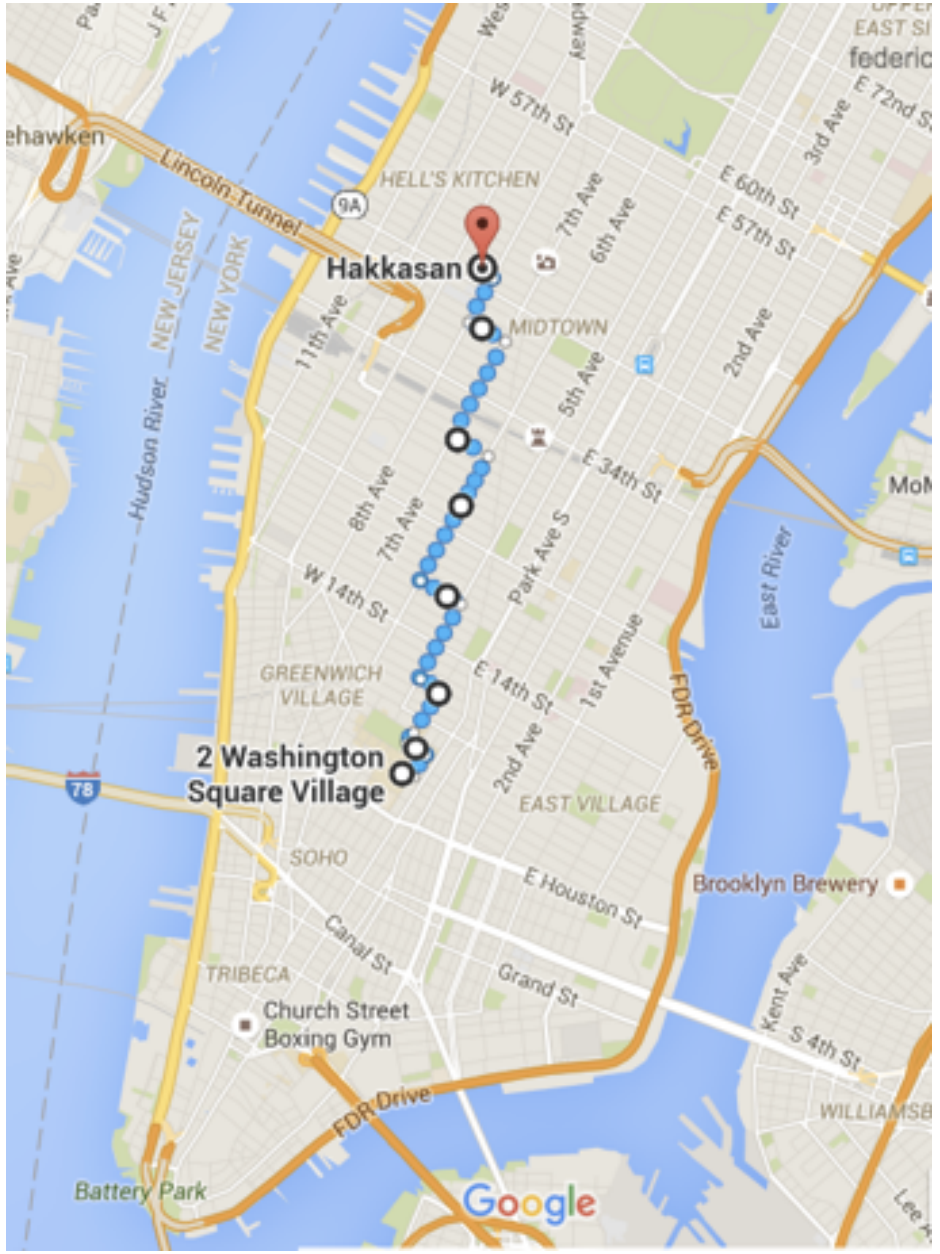
Minkowski family of distances

$$D(i,j) = \sqrt[p]{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^p}$$

Manhattan: p=1

$$D_{Man}(i,j) = \sum_{k=1}^{N} |x_{ik} - x_{jk}|$$

features: x, y



distance metrics continuous variables

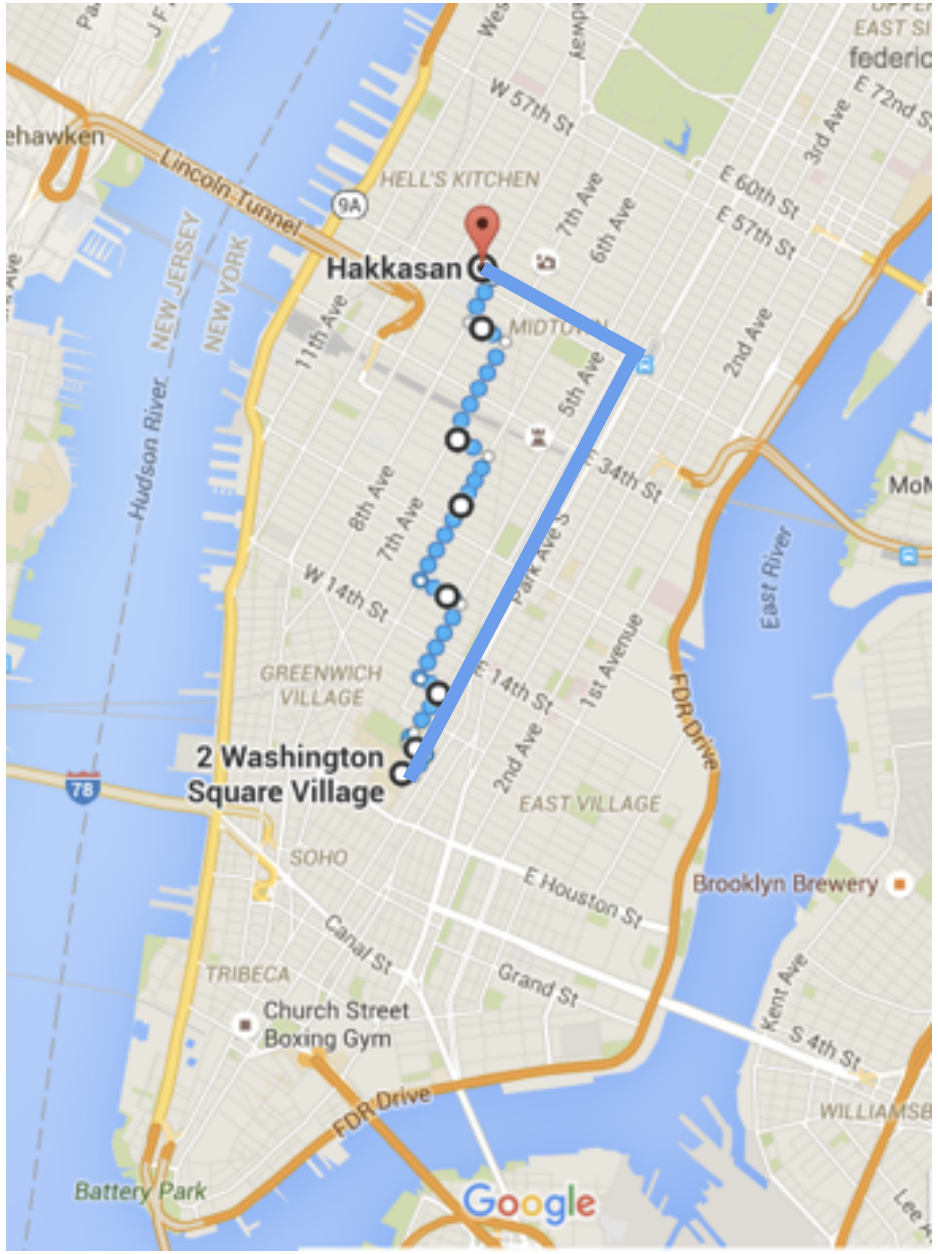
Minkowski family of distances

$$D(i,j) = \sqrt[p]{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^p}$$

Manhattan: p=1

$$D_{Man}(i,j) = \sum_{k=1}^{N} |x_{ik} - x_{jk}|$$

features: x, y



distance metrics

continuous variables

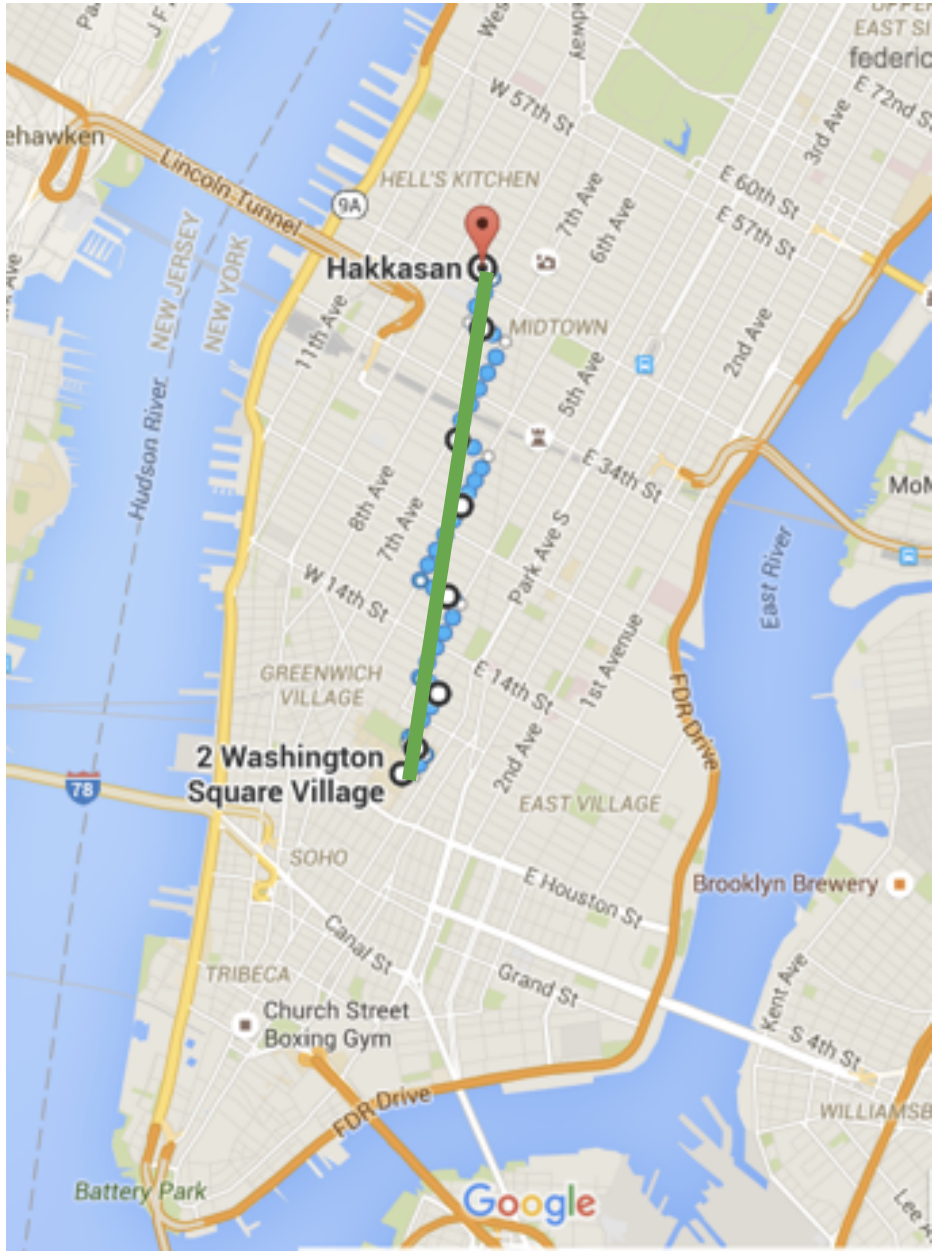
Minkowski family of distances

$$D(i,j) = \sqrt[p]{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^p}$$

Euclidean: p=2

$$D_{Euc}(i,j) = \sqrt{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^2}$$

features: x, y



distance metrics

continuous variables

features

latitude and longitude

$\phi_i, \lambda_i, \phi_j, \lambda_j$

Minkowski family of distances

$$D(i,j) = \sqrt[p]{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^p}$$

Great Circle distance

$$D(i,j) = R \arccos(\sin\phi_i \cdot \sin\phi_j + \cos\phi_i \cdot \cos\phi_j \cdot \cos\Delta\lambda)$$

distance metrics

categorical variables: *binary*

observation *i*

observation *j*

	1	**0**	**sum**
1	M11	M10	M11+M10
0	M01	M00	M01+M00
sum	M11+M01	M10+M00	M11+M00+ M01+ M10

$M_{i=0,j=0}$: number of features in neither

$M_{i=1,j=1}$: number of features in both

$M_{i=1,j=0}$: number of features in *i* but not *j*

$M_{i=0,j=1}$: number of features in *j* but not *i*

Uses presence/absence of features in data

distance metrics

categorical variables: *binary*

observation *i*

observation *j*

	1	**0**	**sum**
1	M11	M10	M11+M10
0	M01	M00	M01+M00
sum	M11+M01	M10+M00	M11+M00+ M01+ M10

Simple Matching Coefficient or Rand similarity

$$SMC(i,j) = \frac{M_{i=0,j=0}+M_{i=1,j=1}}{M_{i=0,j=0}+M_{i=1,j=0}+M_{i=0,j=1}+M_{i=1,j=1}}$$

Simple Matching Distance

$$SMD(i,j) = 1 - SMC(i,j)$$

$M_{i=0,j=0}$: number of features in neither

$M_{i=1,j=1}$: number of features in both

$M_{i=1,j=0}$: number of features in *i* but not *j*

$M_{i=0,j=1}$: number of features in *j* but not *i*

Uses presence/absence of features in data

distance metrics

categorical variables: *binary*

observation *i*

observation *j*

	1	**0**	**sum**
1	M11	M10	M11+M10
0	M01	M00	M01+M00
sum	M11+M01	M10+M00	M11+M00+ M01+ M10

Simple Matching Coefficient or Rand similarity

$$SMC(i,j) = \frac{M_{i=0,j=0}+M_{i=1,j=1}}{M_{i=0,j=0}+M_{i=1,j=0}+M_{i=0,j=1}+M_{i=1,j=1}}$$

Simple Matching Distance

$$SMD(i,j) = 1 - SMC(i,j)$$

$M_{i=0,j=0}$: number of features in neither

$M_{i=1,j=1}$: number of features in both

$M_{i=1,j=0}$: number of features in *i* but not *j*

$M_{i=0,j=1}$: number of features in *j* but not *i*

Uses presence/absence of features in data

distance metrics

categorical variables: *binary*

observation *i*

observation *j*

	1	**0**	**sum**
1	M11	M10	M11+M10
0	M01	M00	M01+M00
sum	M11+M01	M10+M00	M11+M00+ M01+ M10

Simple Matching Coefficient or Rand similarity

$$SMC(i,j) = \frac{M_{i=0,j=0}+M_{i=1,j=1}}{M_{i=0,j=0}+M_{i=1,j=0}+M_{i=0,j=1}+M_{i=1,j=1}}$$

Simple Matching Distance

$$SMD(i,j) = 1 - SMC(i,j)$$

$M_{i=0,j=0}$: number of features in neither

$M_{i=1,j=1}$: number of features in both

$M_{i=1,j=0}$: number of features in *i* but not *j*

$M_{i=0,j=1}$: number of features in *j* but not *i*

Uses presence/absence of features in data

distance metrics

categorical variables: *binary*

observation *i*

observation *j*

	1	**0**	**sum**
1	M11	M10	M11+M10
0	M01	M00	M01+M00
sum	M11+M01	M10+M00	M11+M00+ M01+ M10

Jaccard similarity

$$J(i,j) = \frac{M_{i=1,j=1}}{M_{i=0,j=1}+M_{i=1,j=0}+M_{i=1,j=1}}$$

Jaccard distance

$$D(i,j) = 1 - J(i,j)$$

distance metrics

categorical variables: *binary*

observation *i*

observation *j*

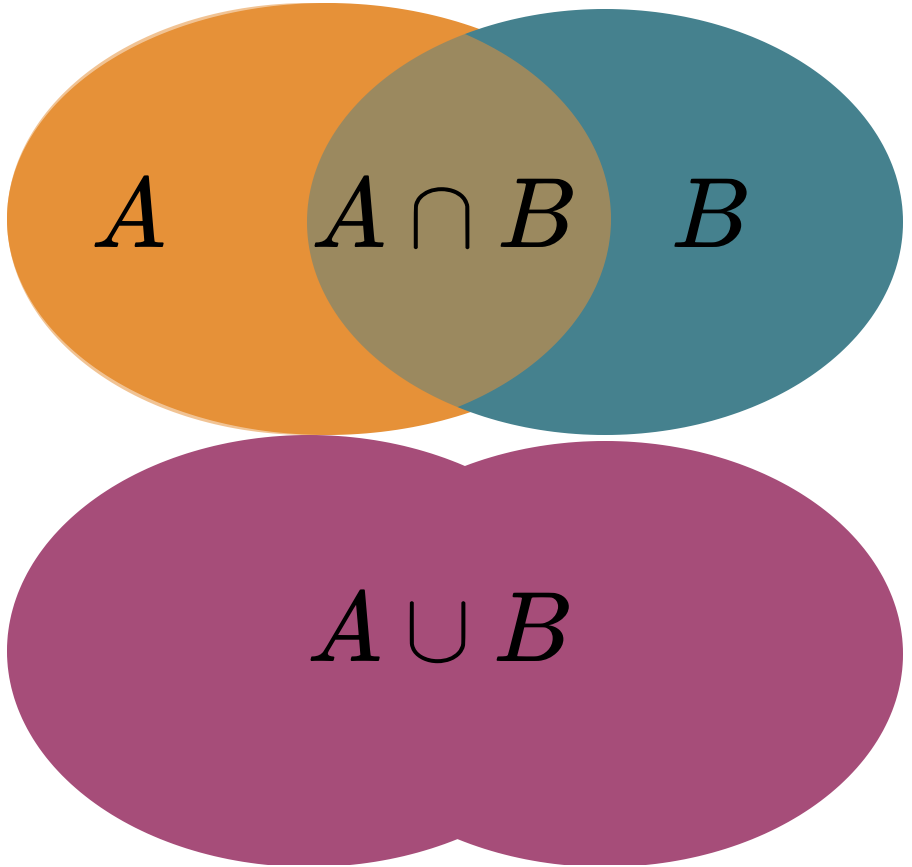
	1	**0**	**sum**
1	M11	M10	M11+M10
0	M01	M00	M01+M00
sum	M11+M01	M10+M00	M11+M00+ M01+ M10

Jaccard similarity

$$J(i,j) = \frac{A \cap B}{A \cup B}$$

Jaccard distance

$$D(i,j) = 1 - J(i,j)$$



distance metrics

categorical variables: *binary*

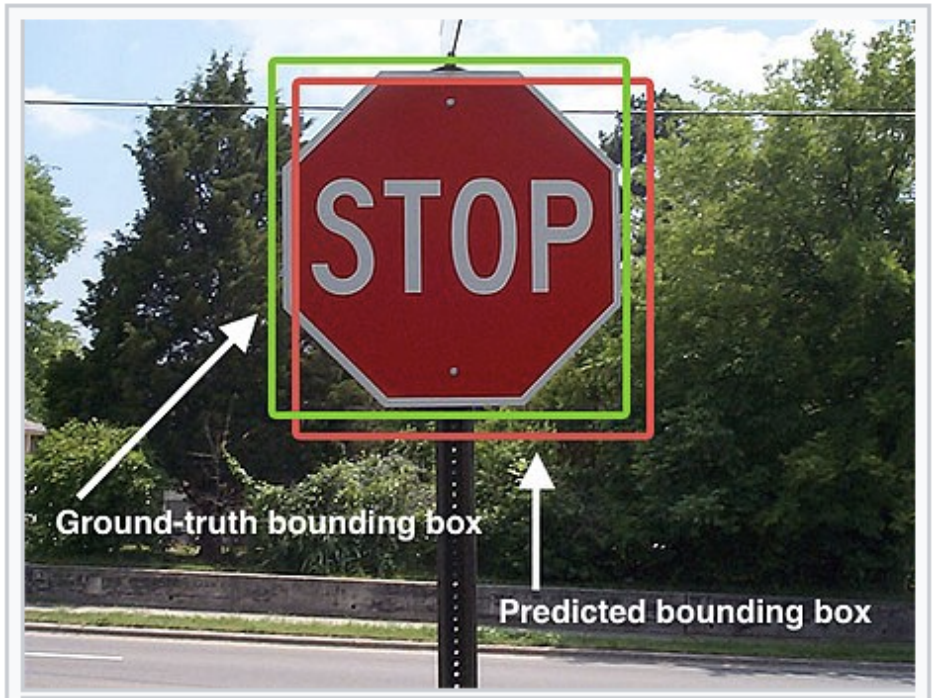
Jaccard similarity

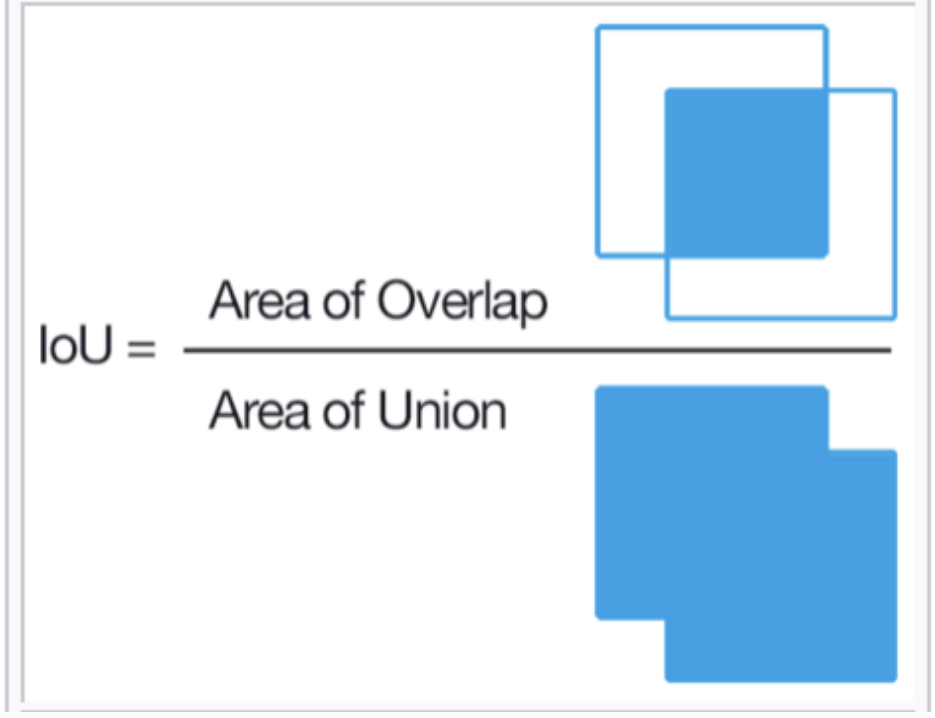
$$J(i,j) = \frac{A \cap B}{A \cup B}$$

Application to Deep Learning for image recognition

Convolutional Neural Nets

https://en.wikipedia.org/wiki/Jaccard_index





distance metrics

https://docs.scipy.org/doc/scipy/reference/spatial.distance.html

SciPy.org

SciPy.org | Docs | SciPy v1.3.1 Reference Guide

Spatial algorithms and data structures (scipy.spatial)

index | modules | next | previous

Distance computations (scipy.spatial.distance)

Function Reference

Distance matrix computation from a collection of raw observation vectors stored in a rectangular array.

pdist(X[, metric])	Pairwise distances between observations in n-dimensional space.

Table of Contents

- Distance computations (scipy.spatial.distar
  - Function Reference

Previous topic

scipy.spatial.procruste

Next topic

Distance functions between two boolean vectors (representing sets) `u` and `v`. As in the case of numerical vectors, `pdist` is more efficient for computing the distances between all pairs.

dice(u, v[, w])	Compute the Dice dissimilarity between two boolean 1-D arrays.
hamming(u, v[, w])	Compute the Hamming distance between two 1-D arrays.
jaccard(u, v[, w])	Compute the Jaccard-Needham dissimilarity between two boolean 1-D arrays.

Distance functions between two numeric vectors `u` and `v`. Computing distances over a large collection of vectors is inefficient for these functions. Use `pdist` for this purpose.

braycurtis(u, v[, w])	Compute the Bray-Curtis distance between two 1-D arrays.
canberra(u, v[, w])	Compute the Canberra distance between two 1-D arrays.
chebyshev(u, v[, w])	Compute the Chebyshev distance.
cityblock(u, v[, w])	Compute the City Block (Manhattan) distance.

whitening

3

Data can have covariance (and it almost always does!)

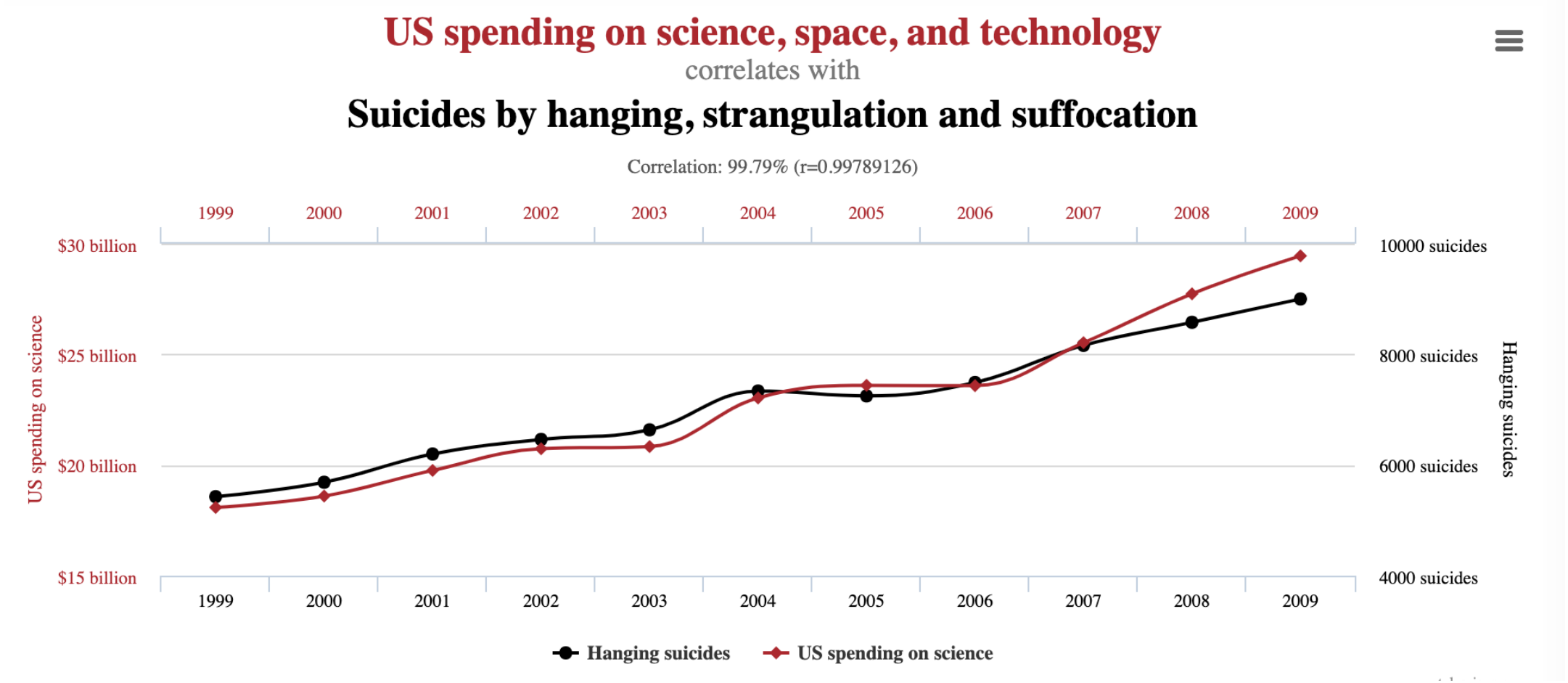
PLUTO Manhattan data (42,000 x 15)

axis 0 -> observations

	Borough	Block	Lot	CD	CT2010	CB2010	SchoolDist	Council	ZipCode	FireComp	...	TaxMap	EDesigNum	APPBBL	APPDate	PLUTOMapID	Ver
0	MN	1545	52	108	138	4000	02	5	10028	E022	...	10515	None	0.000000e+00	None	1	
1	MN	723	7501	104	93	6000	02	3	10001	E003	...	10302	None	1.007230e+09	11/30/2006	1	
2	MN	1680	48	111	170	5000	04	8	10029	E091	...	10605	None	0.000000e+00	None	1	
3	MN	1385	32	108	130	2003	02	4	10021	E039	...	10508	None	0.000000e+00	None	1	
4	MN	1197	27	107	169	5000	03	6	10024	E074	...	10408	None	0.000000e+00	None	1	

axis 1 -> features

Data can have covariance (and it almost always does!)

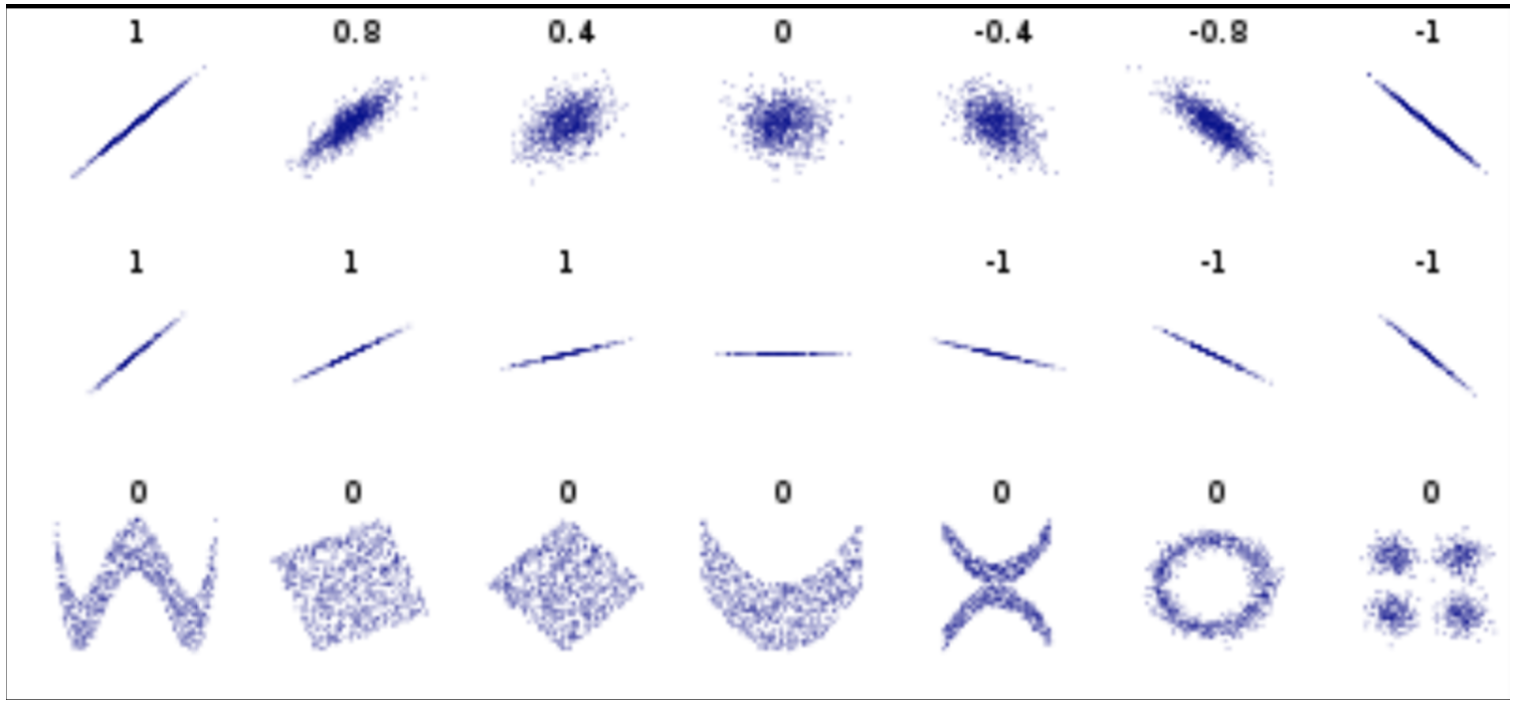


https://www.tylervigen.com/spurious-correlations

Data can have covariance (and it almost always does!)

Pearson's correlation (linear correlation)

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$



correlation = correlation / variance

Data can have covariance (and it almost always does!)

PLUTO Manhattan data (42,000 x 15) correlation matrix

axis 0 -> observations

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

axis 1 -> features

Data can have covariance (and it almost always does!)

PLUTO Manhattan data (42,000 x 15) correlation matrix

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & 0 & \cdots & 0 \\ 0 & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

A covariance matrix is diagonal if the data has no correlation

Full On Whitening

: remove covariance by diagonalizing the transforming the data with a matrix that diagonalizes the covariance matrix

find the matrix W that diagonalized Σ

```
from zca import ZCA
import numpy as np
X = np.random.random((10000, 15)) # data
array
trf = ZCA().fit(X)
X_whitened = trf.transform(X)
X_reconstructed =
trf.inverse_transform(X_whitened)
assert(np.allclose(X, X_reconstructed))
```

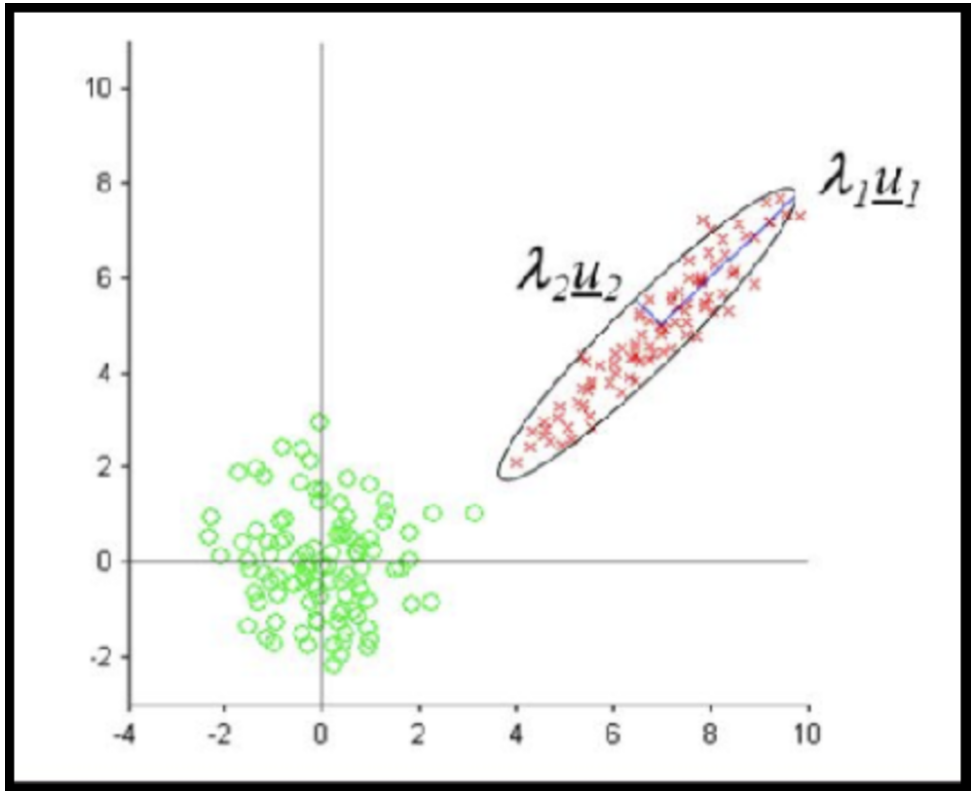
this is at best hard, in some cases impossible even numerically on large datasets

Generic preprocessing

Data can have covariance (and it almost always does!)

ORIGINAL DATA

STANDARDIZED DATA



Data that is not correlated appear as a sphere in the Ndimensional feature space

Generic preprocessing

for each feature: divide by standard deviation and subtract mean

```
X = preprocessing.scale(X, axis=0)
```

Last executed 2018-12-12 09:35:39 in 46ms

```
X.mean(axis=0)
```

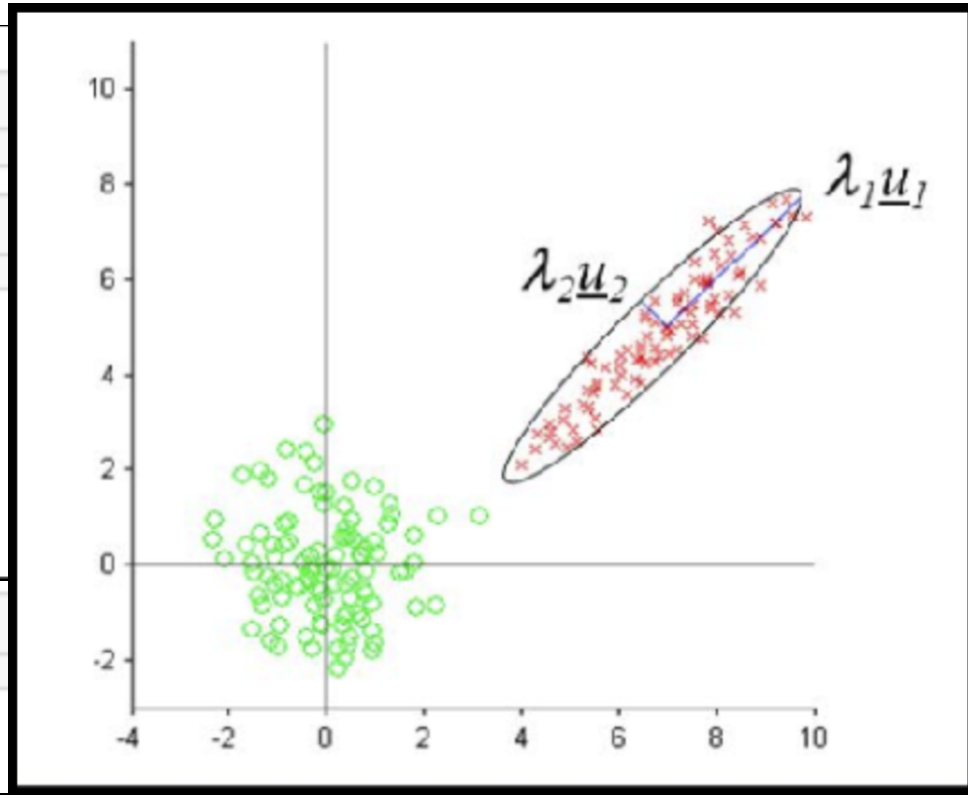
Last executed 2018-12-12 09:35:40 in 13ms

```
array([ 3.85590369e-16, -6.93196168e-17, -5.90549813e-16, -5.95882091e-16,
       -8.49165306e-16, -1.57568821e-15, -8.00508267e-16,  5.55890004e-16,
       -5.16564452e-16,  1.09378357e-15,  3.46598084e-16,  2.31954102e-16,
        2.78611537e-16, -2.51283611e-16,  8.66495210e-18,  3.03939858e-16,
       -3.66594127e-17, -9.27149875e-16, -6.39873386e-16,  2.93275302e-17,
        9.19817992e-17,  6.33208038e-18, -1.99960433e-17,  9.55144336e-16,
       -2.20623011e-16,  6.93196168e-17, -9.46479383e-17,  2.26621824e-16,
        6.93196168e-17,  2.32953905e-16])
```

```
X.std(axis=0)
```

Last executed 2018-12-12 09:36:28 in 19ms

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```



mean of each feature should be 0, standard deviation of each feature should be 1

how clustring works

4

- **Partitioning**
  - **Hard clustering**
  - **Soft Clustering**

K-means (McQueen ’67)

K-medoids (Kaufman & Rausseeuw ’87)

Expectation Maximization (Dempster,Laird,Rubin ’77)

- **Hirarchical**
  - **agglomerative**
  - **devisive**

- **also:**
  - **Density based**
  - **Grid based**
  - **Model based**

5 clustering by partitioning

k-means:

Hard partitioning cluster method

5.1

K-means: *the algorithm*

```
Choose N “centers” guesses: random points in the feature
space
repeat:
  Calculate distance between each point and each center

  Assign each point to the closest center

  Calculate the new cluster centers

untill (convergence):
  when clusters no longer change
```

K-means:

Objective: minimizing the aggregate distance within the cluster.

Order: #clusters #dimensions #iterations #datapoints ***O(KdN)***

CONs:

Its non-deterministic: the result depends on the (random) starting point

It only works where the mean is defined: alternative is K-medoids which represents the cluster by its central member (median), rather than by the mean

Must declare the number of clusters upfront (how would you know it?)

PROs:

Scales linearly with *d* and *N*

K-means: *the objective function*

Objective: minimizing the aggregate distance within the cluster.

Order: #clusters #dimensions #iterations #datapoints ***O(KdN)***

O(KdN): complexity scales linearly with

-*d* number of dimensions

-*N* number of datapoints

-*K* number of clusters

K-means: *the objective function*

Objective: minimizing the aggregate distance within the cluster.

total intra-cluster variance $\sum_k \sum_{i \in k} (\vec{x}_i - \vec{\mu}_k)^2$

Order: #clusters #dimensions #iterations #datapoints ***O(KdN)***

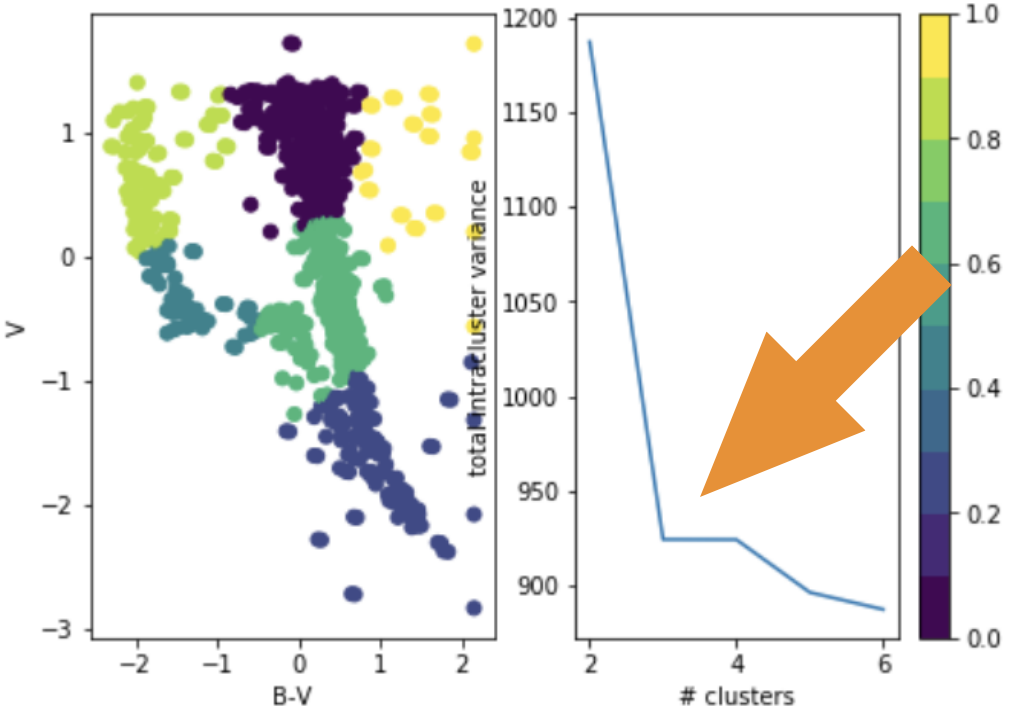
Must declare the number of clusters

either you know it because of domain knowledge

or

you choose it after the fact: "*elbow method*"

https://github.com/fedhere/DSPS/blob/master/lab10/StellarPopClustersLab.ipynb



K-means: *the objective function*

Objective: minimizing the aggregate distance within the cluster.

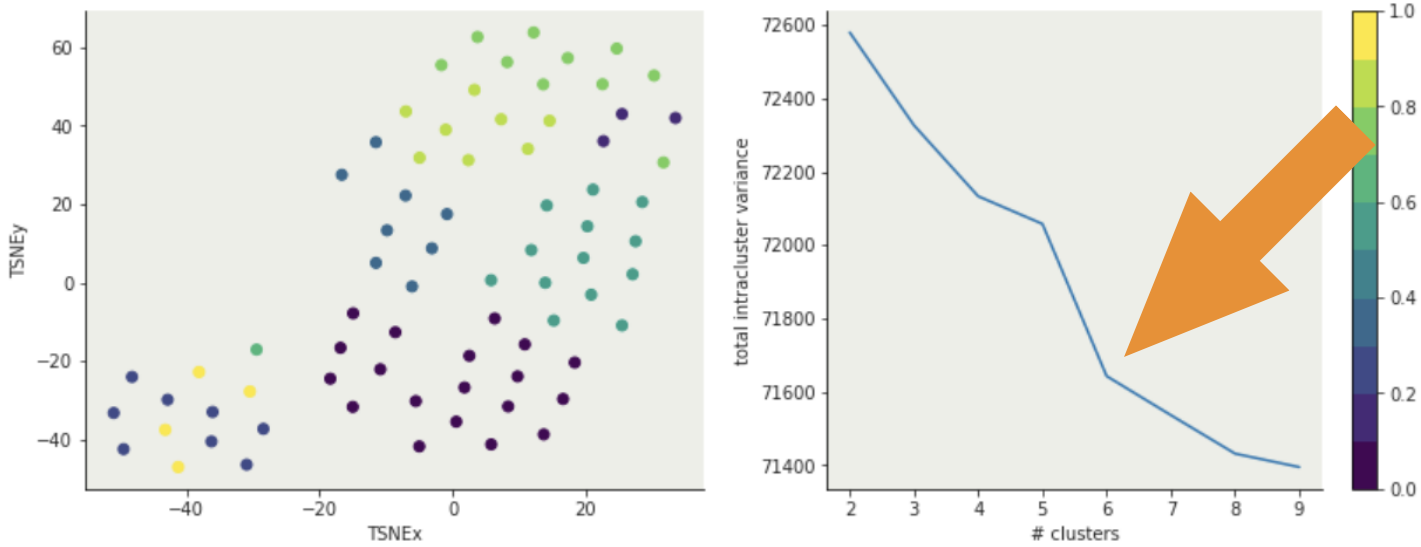
total intra-cluster variance $\sum_k \sum_{i \in k} (\vec{x}_i - \vec{\mu}_k)^2$

Order: #clusters #dimensions #iterations #datapoints ***O(KdN)***

Must declare the number of clusters upfront (how would you know it?)

either domain knowledge or

after the fact: "elbow method"



K-means: *hyperparameters*

class `sklearn.cluster.` **`KMeans`** (*n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None, algorithm='auto'*) ¶

[source]

- *n_clusters* : number of clusters
- *init* : the initial centers or a scheme to choose the center

  'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

  'random': choose k observations (rows) at random from data for the initial centroids.

  If an ndarray is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.
- *n_init* : if >1 it is effectively an ensamble method: runs n times with different initializations
- *random_state* : for reproducibility

expectation-maximization

Soft partitioning cluster method

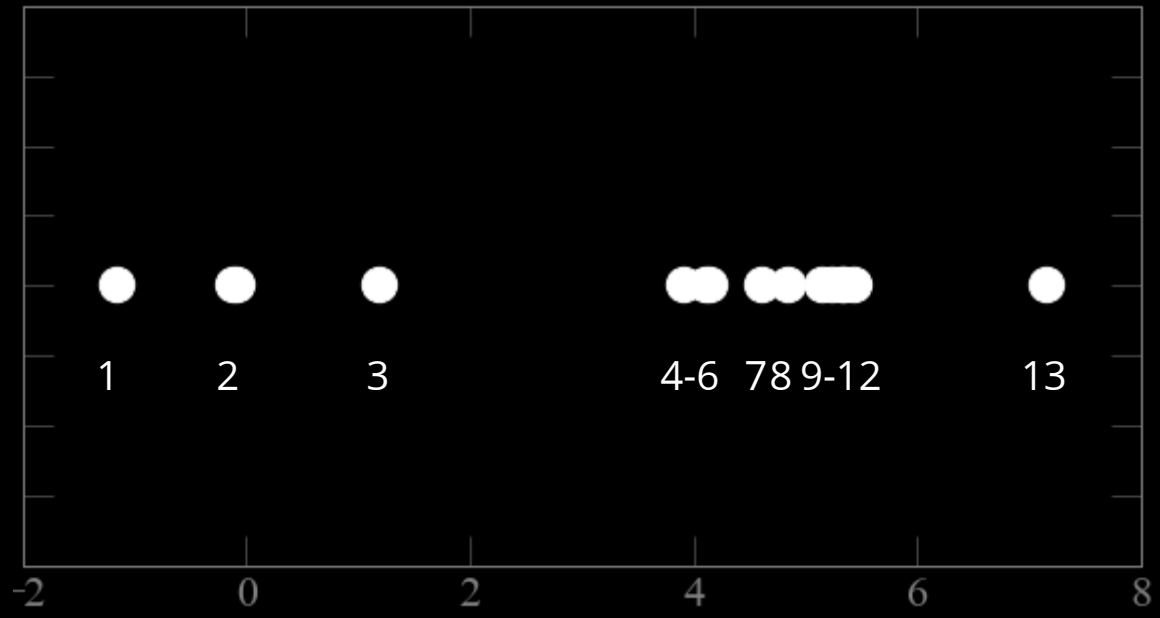
52

Hard clustering : each object in the sample belongs to only 1 cluster

Soft clustering : to each object in the sample we assign a degree of belief that it belongs to a cluster

Soft = probabilistic

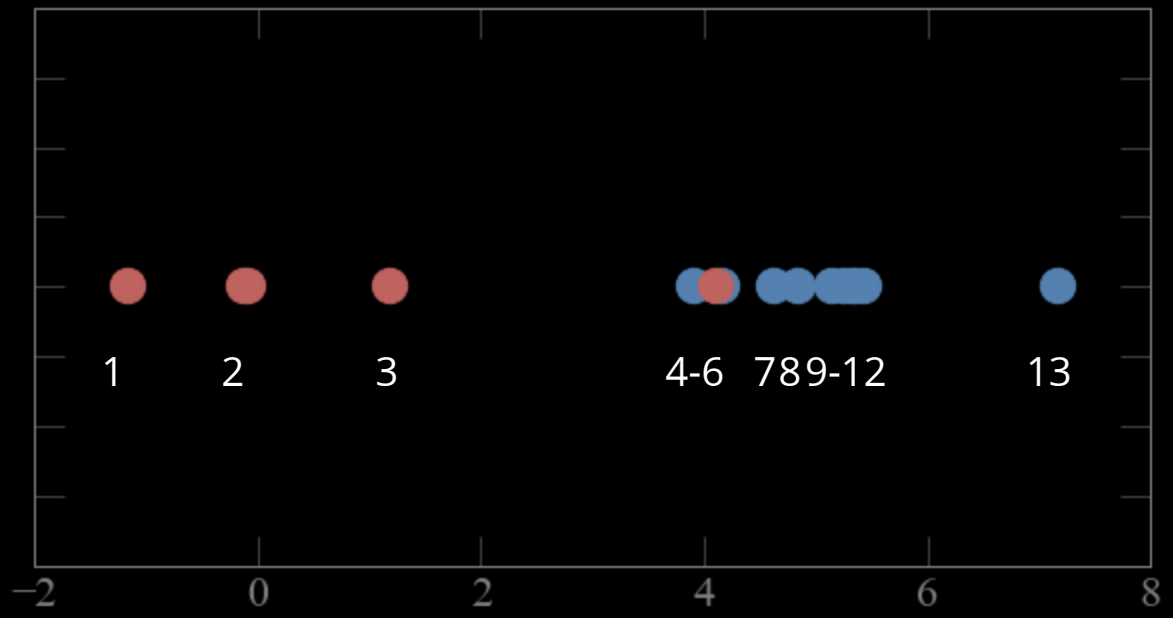
Mixture models



these points come from 2 gaussian distribution.

which point comes from which gaussian?

Mixture models



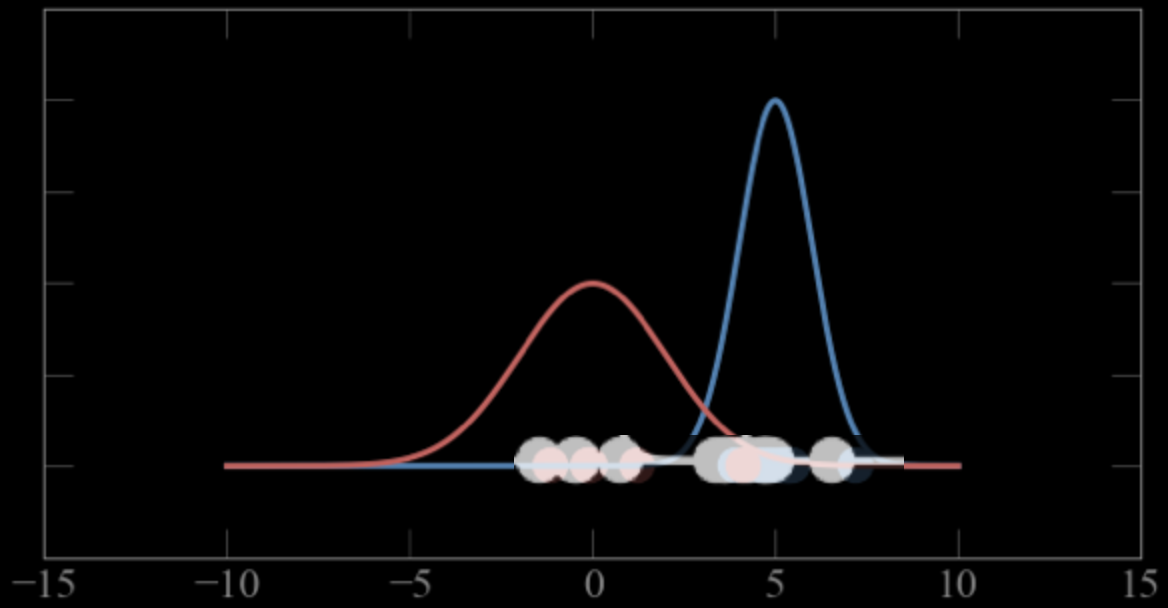
CASE 1:

if i know which point comes from which gaussian

i can solve for the parameters of the gaussian

(e.g. maximizing likelihood)

Mixture models

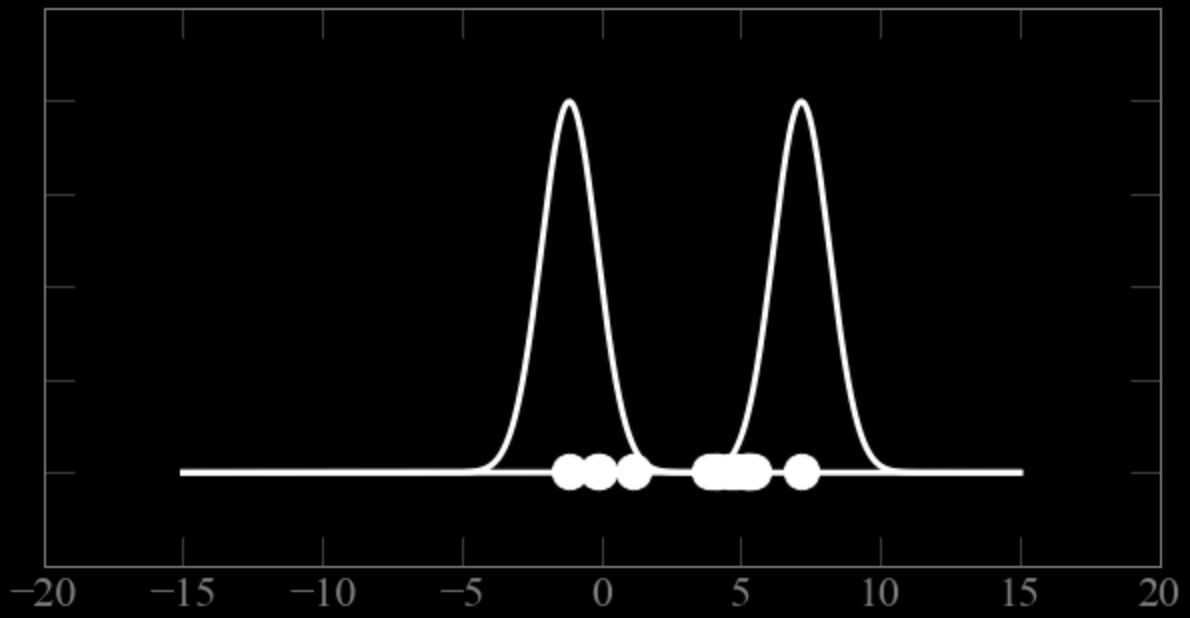


CASE 2:

if i know which the parameters (μ,σ) of the gaussians

i can figure out which gaussian each point is most likely to come from (calculate probability)

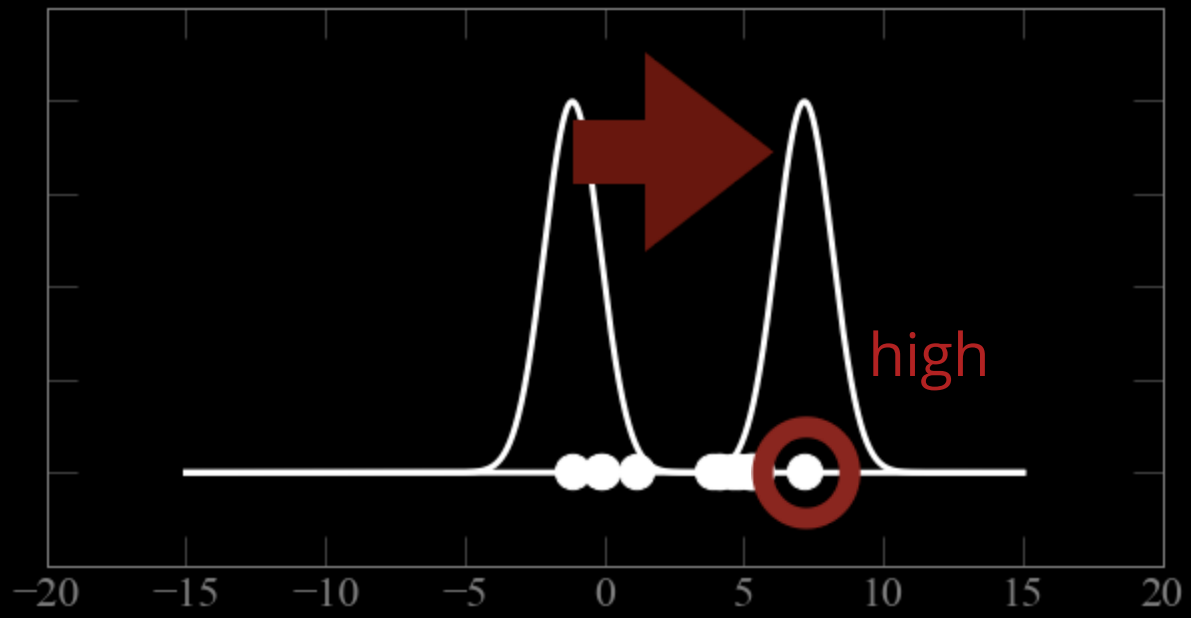
Mixture models:
Expectation maximization



Guess parameters $g = (\mu,\sigma)$ for 2 Gaussian distributions A and B

calculate the probability of each point to belong to A and B

Mixture models:
Expectation maximization

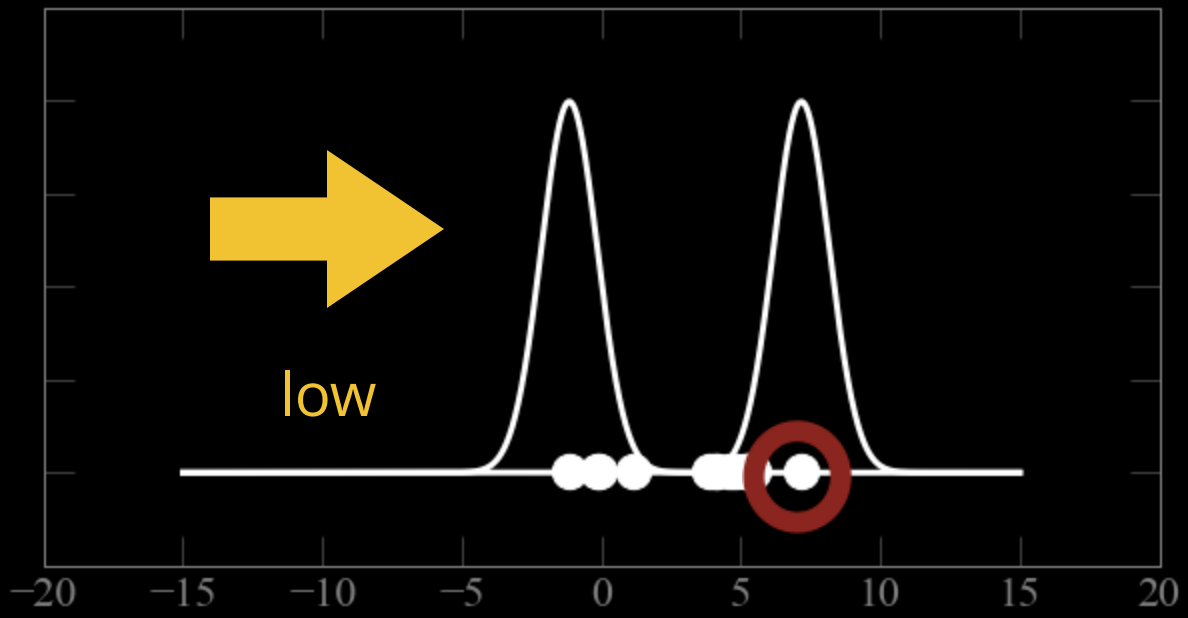


$$P(x_i|\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{x_i-\mu_j}{2\sigma_j^2}\right)$$

Guess parameters *g*= *(μ,σ)* for 2 Gaussian distributions A and B

calculate the probability of each point to belong to A and B

Mixture models:
Expectation maximization

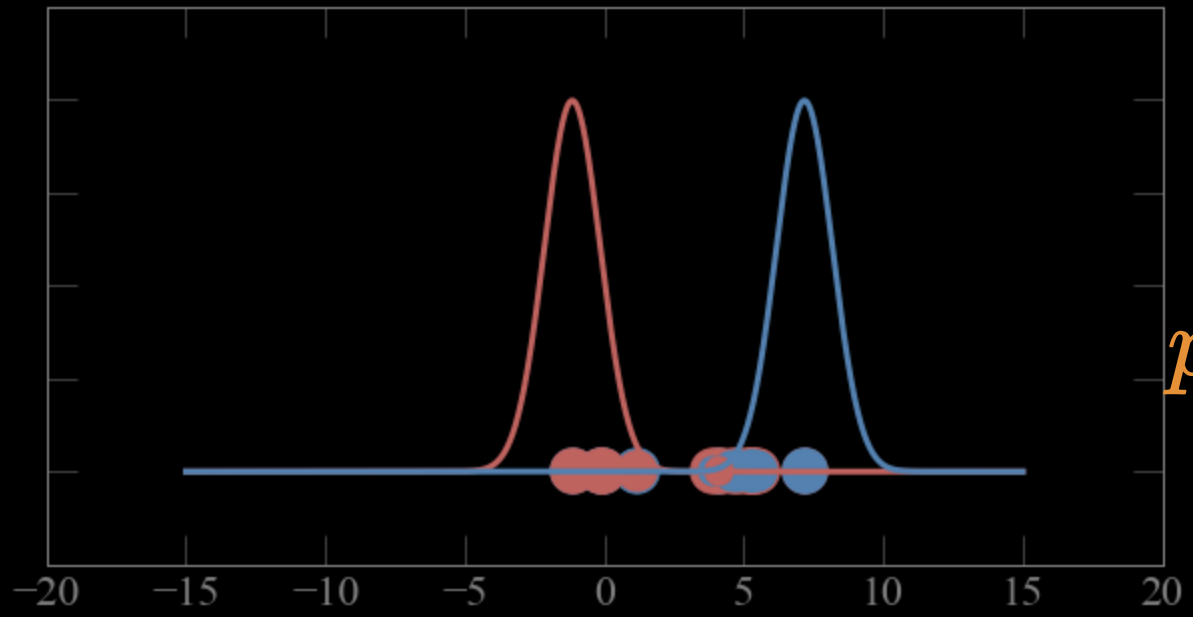


$$P(x_i|\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{x_i-\mu_j}{2\sigma_j^2}\right)$$

Guess parameters *g*= *(μ,σ)* for 2 Gaussian distributions A and B

calculate the probability of each point to belong to A and B

Mixture models:
Expectation maximization



$$P(x_i|\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{x_i-\mu_j}{2\sigma_j^2}\right)$$

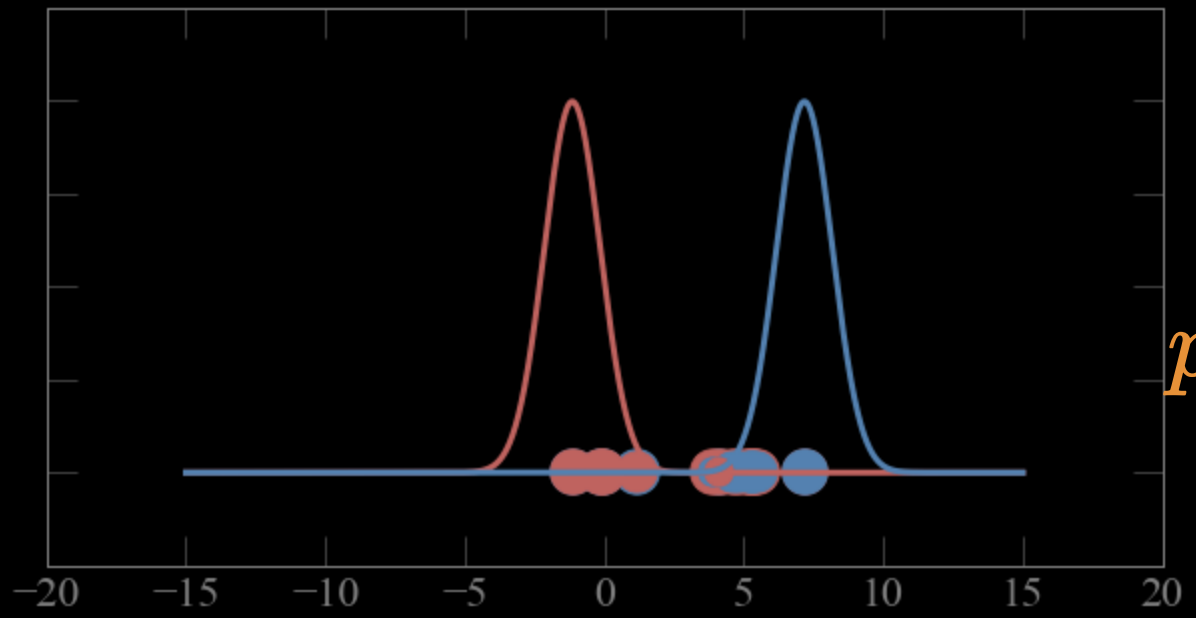
Bayes theorem: P(A|B) = P(B|A) P(A) / P(B)

$$p_{ij} = P(g_1|x_i) = \frac{P(x_i|g_1)|P(g_1)}{P(x_i|g_1)P(g_1)+P(x_i|g_2)P(g_2)}$$

Guess parameters *g*= *(μ,σ)* for 2 Gaussian distributions A and B

1- calculate the probability ***p_ji*** of each point to belong to gaussian ***j***

Mixture models:
Expectation maximization



$$P(x_i|\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{x_i-\mu_j}{2\sigma_j^2}\right)$$

Bayes theorem: P(A|B) = P(B|A) P(A) / P(B)

$$p_{ij} = P(g_1|x_i) = \frac{P(x_i|g_1)|P(g_1)}{P(x_i|g_1)P(g_1)+P(x_i|g_2)P(g_2)}$$

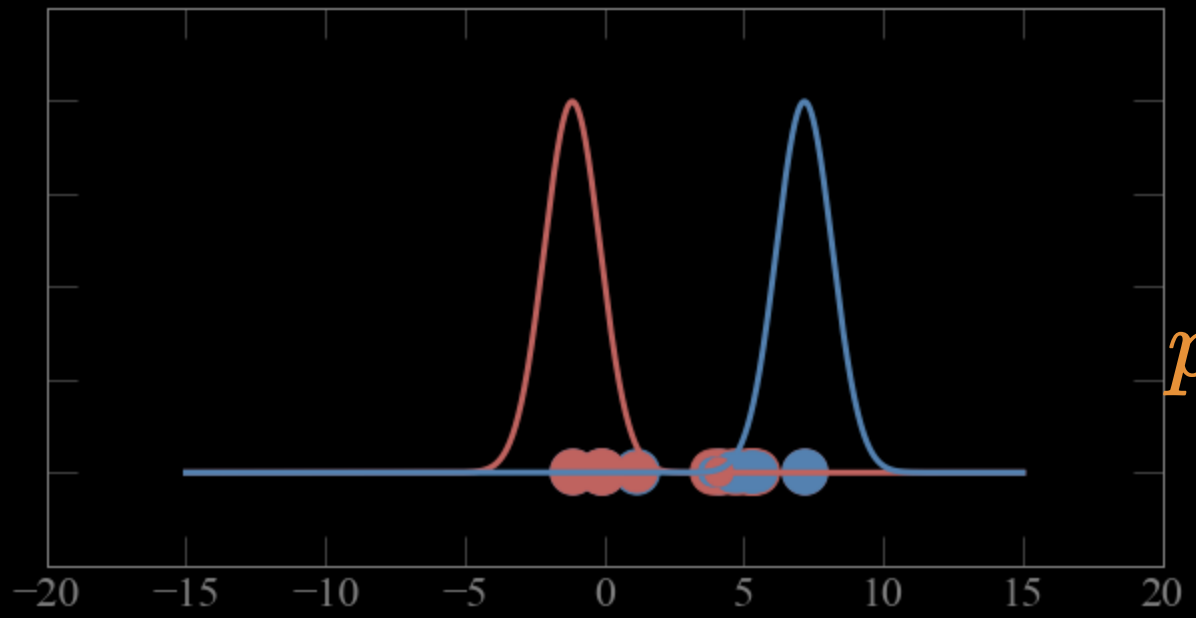
$$\mu_i = \frac{\sum_j P(g_i|x_j)x_j}{\sum_j P(g_i|x_j)} \qquad \sigma_j = \frac{\sum_i P(g_j|x_i)(x_i-\mu_j)^2}{\sum_j P(g_j|x_i)}$$

Guess parameters *g*= *(μ,σ)* for 2 Gaussian distributions A and B

1- calculate the probability ***p_ji*** of each point to belong to gaussian ***j***

2a - calculate the weighted ***mean*** of the cluster, weighted by the *p_ji*

Mixture models:
Expectation maximization



$$P(x_i|\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{x_i-\mu_j}{2\sigma_j^2}\right)$$

Bayes theorem: P(A|B) = P(B|A) P(A) / P(B)

$$p_{ij} = P(g_1|x_i) = \frac{P(x_i|g_1)|P(g_1)}{P(x_i|g_1)P(g_1)+P(x_i|g_2)P(g_2)}$$

$$\mu_i = \frac{\sum_j P(g_i|x_j)x_j}{\sum_j P(g_i|x_j)} \qquad \sigma_j = \frac{\sum_i P(g_j|x_i)(x_i-\mu_j)^2}{\sum_j P(g_j|x_i)}$$

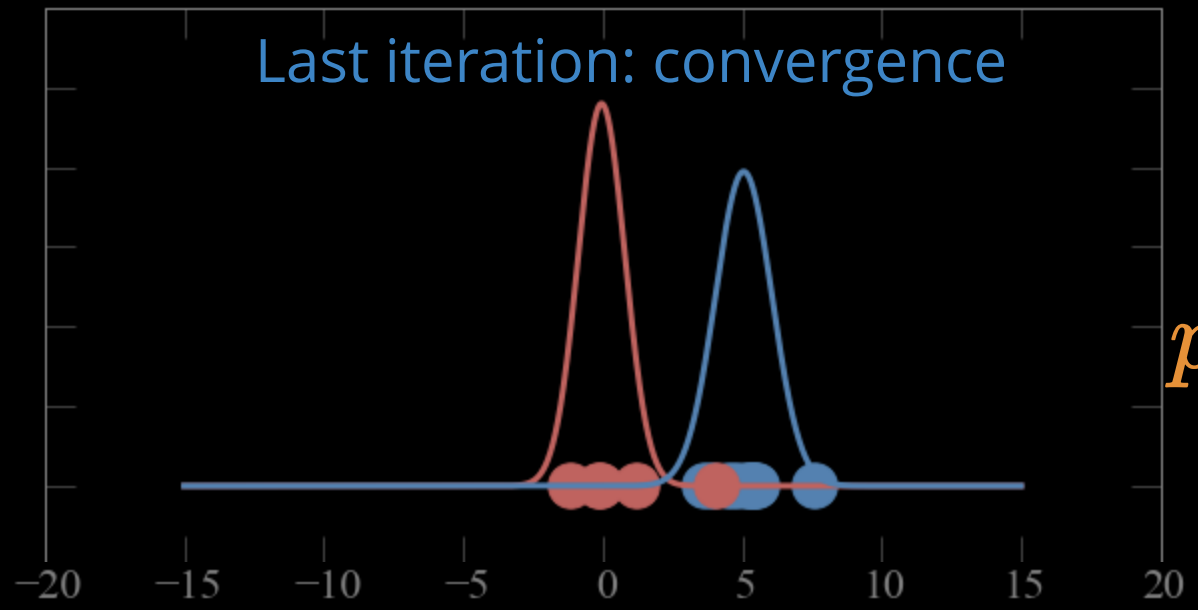
Guess parameters *g*= *(μ,σ)* for 2 Gaussian distributions A and B

1- calculate the probability ***p_ji*** of each point to belong to gaussian ***j***

2a - calculate the weighted ***mean*** of the cluster, weighted by the *p_ji*

2b - calculate the weighted ***sigma*** of the cluster, weighted by the *p_ji*

Mixture models:
Expectation maximization



$$P(x_i|\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{x_i-\mu_j}{2\sigma_j^2}\right)$$

Bayes theorem: P(A|B) = P(B|A) P(A) / P(B)

$$p_{ij} = P(g_1|x_i) = \frac{P(x_i|g_1)|P(g_1)}{P(x_i|g_1)P(g_1)+P(x_i|g_2)P(g_2)}$$

$$\mu_i = \frac{\sum_j P(g_i|x_j)x_j}{\sum_j P(g_i|x_j)} \qquad \sigma_j = \frac{\sum_i P(g_j|x_i)(x_i-\mu_j)^2}{\sum_j P(g_j|x_i)}$$

Alternate expectation and maximization step till convergence

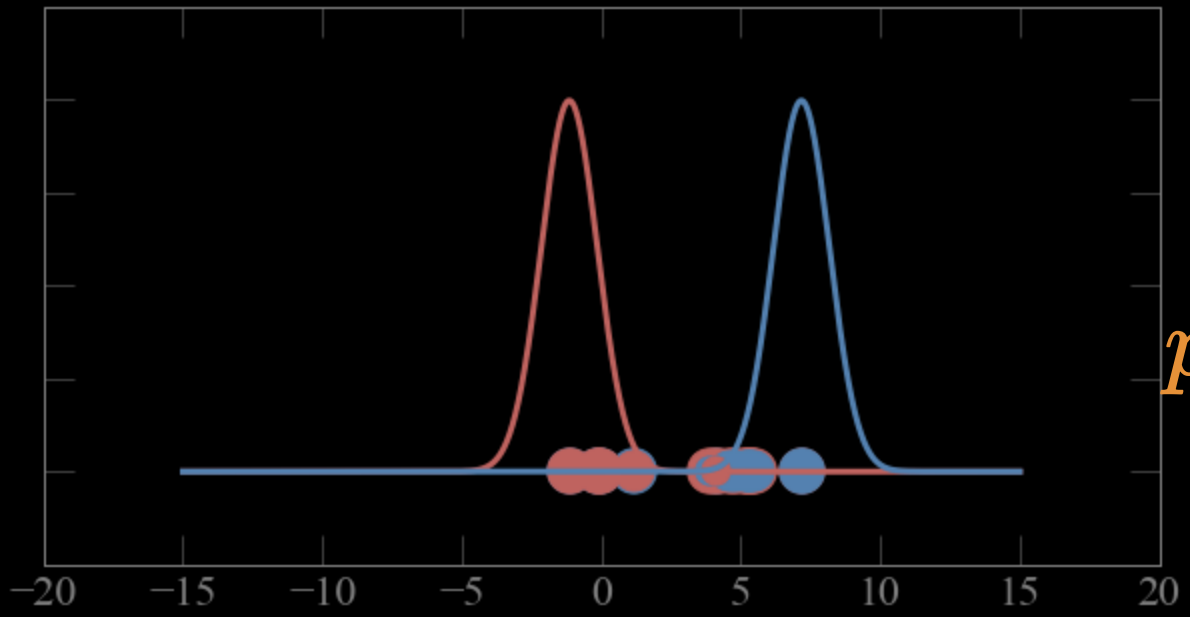
1- calculate the probability ***p_ji*** of each point to belong to gaussian ***j***

2a - calculate the weighted ***mean*** of the cluster, weighted by the *p_ji*

2b - calculate the weighted ***sigma*** of the cluster, weighted by the *p_ji*

} *expectation step* ↓ *maximization step* ↻

Mixture models:
Expectation maximization



$$P(x_i|\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{x_i-\mu_j}{2\sigma_j^2}\right)$$

Bayes theorem: P(A|B) = P(B|A) P(A) / P(B)

$$p_{ij} = P(g_1|x_i) = \frac{P(x_i|g_1)|P(g_1)}{P(x_i|g_1)P(g_1)+P(x_i|g_2)P(g_2)}$$

$$\mu_i = \frac{\sum_j P(g_i|x_j)x_j}{\sum_j P(g_i|x_j)} \qquad \sigma_j = \frac{\sum_i P(g_j|x_i)(x_i-\mu_j)^2}{\sum_j P(g_j|x_i)}$$

Alternate expectation and maximization step till convergence

1- calculate the probability ***p_ji*** of each point to belong to gaussian ***j***

2a - calculate the weighted ***mean*** of the cluster, weighted by the *p_ji*

2b - calculate the weighted ***sigma*** of the cluster, weighted by the *p_ji*

expectation step

maximization step

EM: *the algorithm*

Choose N "centers" guesses (like in K-means)

repeat

Expectation step: Calculate the probability of each distribution given the points

Maximization step: Calculate the new centers and variances as weighted averages of the datapoints, weighted by the probabilities

untill (convergence)

Expectatin Maximization:

Order: #clusters #dimensions #iterations #datapoints #parameters

O(KdNp) (>K-means)

based on Bayes theorem

Its non-deterministic: the result depends on the (random) starting point (like K-mean)

It only works where a probability distribution for the data points can be defines (or equivalently a likelihood) (like K-mean)

Must declare the number of clusters and the shape of the pdf upfront (like K-mean)

Convergence Criteria

General

Any time you have an objective function (or loss function) you need to set up a tolerance : if your objective function did not change by more than ε since the last step you have reached convergence (i.e. you are satisfied)

ε is your tolerance

For clustering:

convergence can be reached if

no more than n data point changed cluster

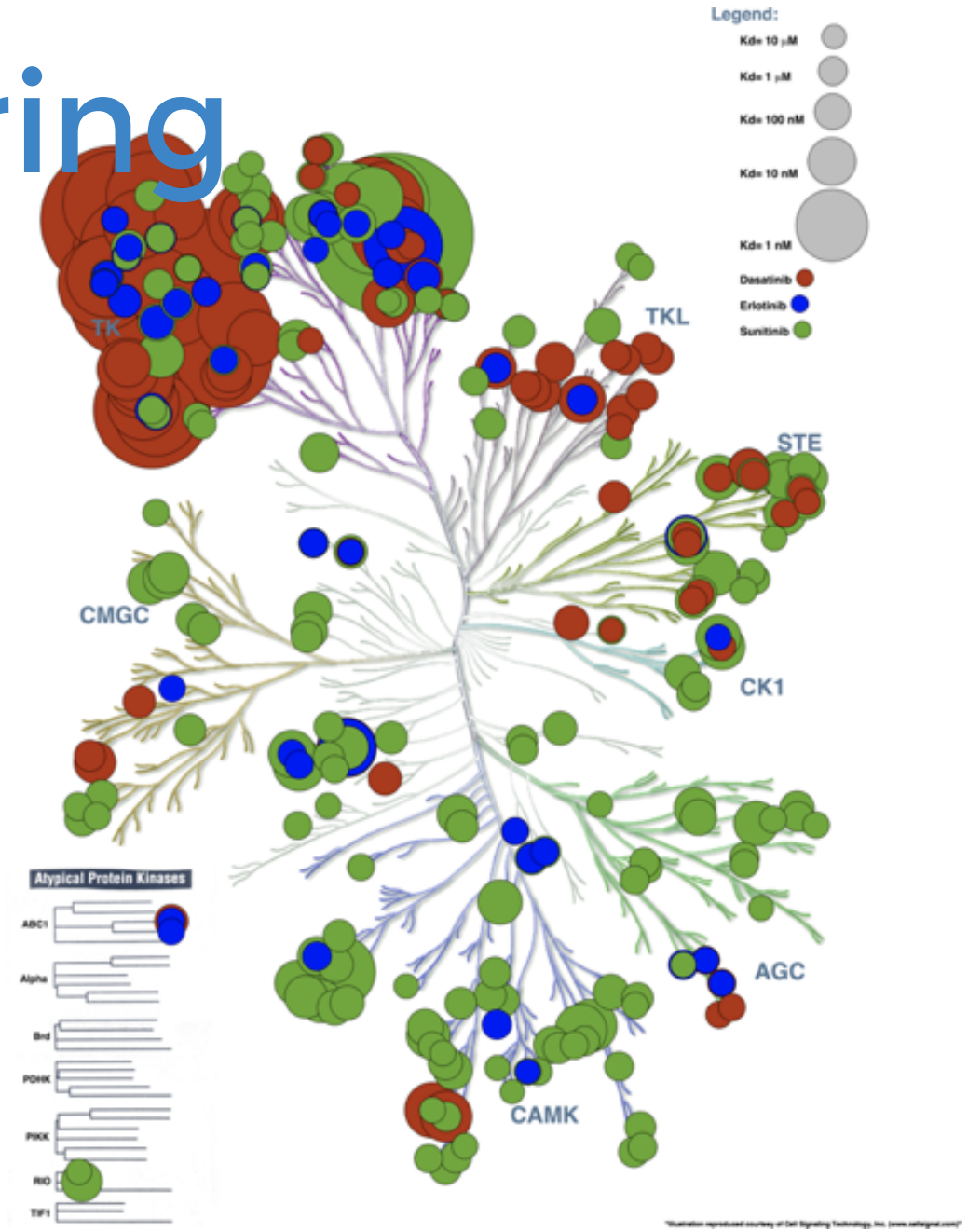
n is your tolerance

Hierarchical clustering

6

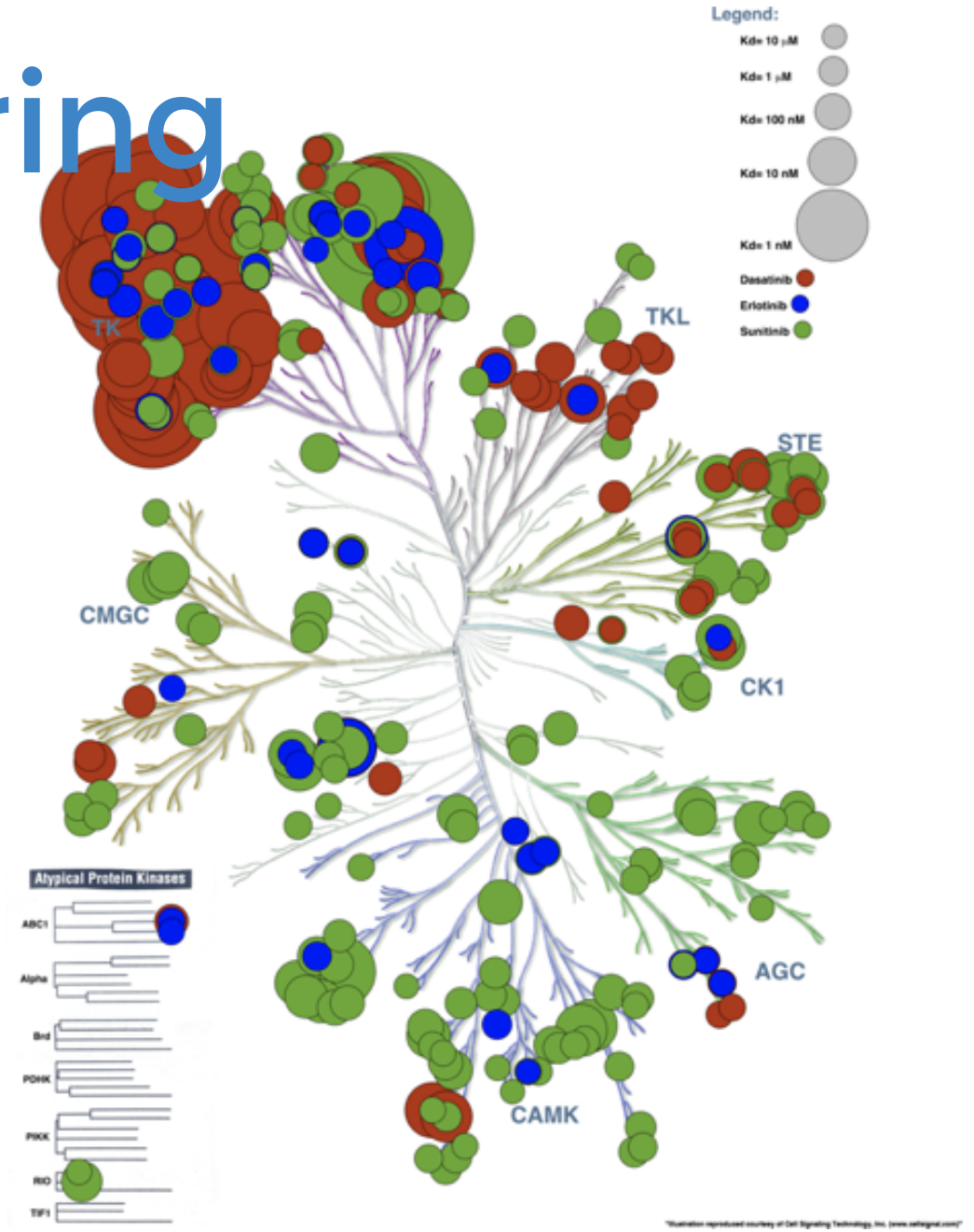
Hierarchical clustering

removes the issue of deciding K (number of clusters)



Hierarchical clustering

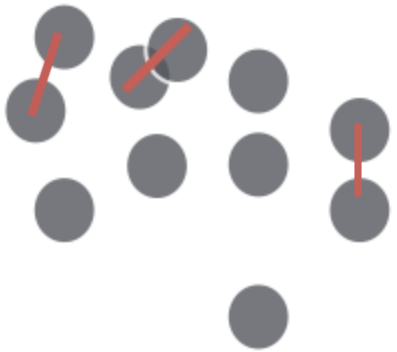
it calculates distance between clusters and single points: *linkage*



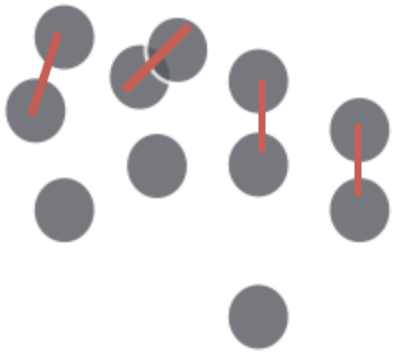
6.1 Agglomerative hierarchical clustering

Hierarchical clustering
agglomerative (bottom up)

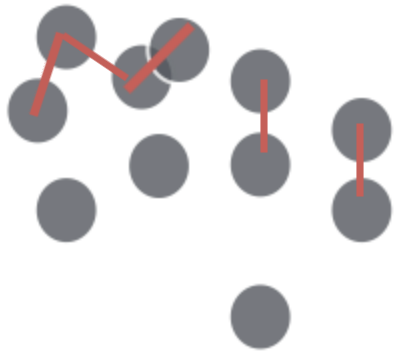
Hierarchical clustering
agglomerative (bottom up)

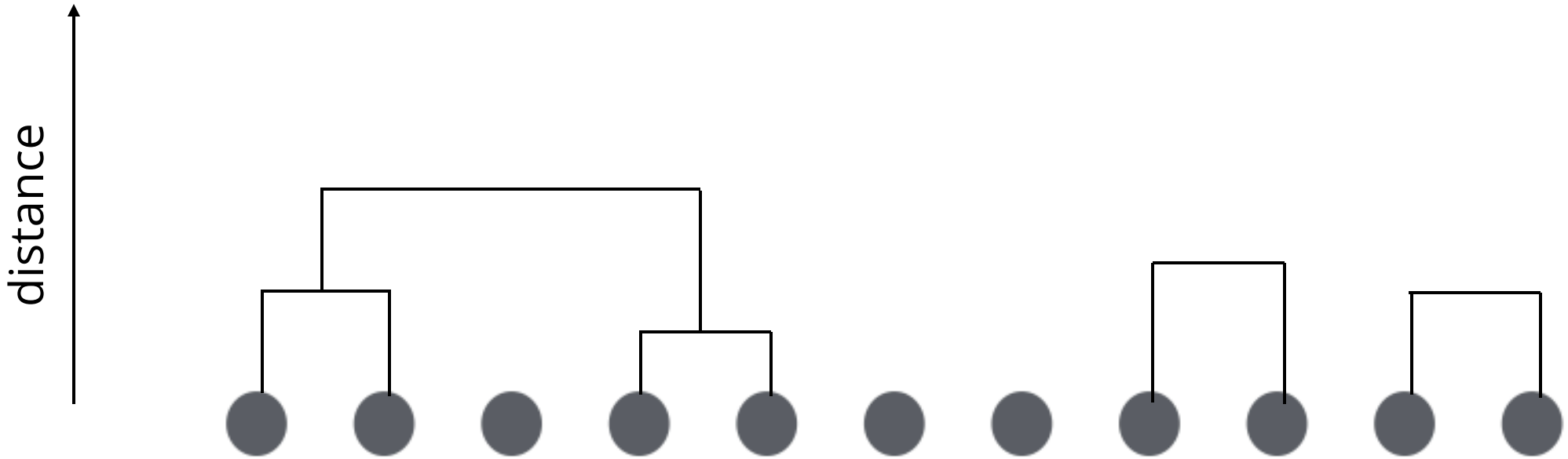


Hierarchical clustering
agglomerative (bottom up)



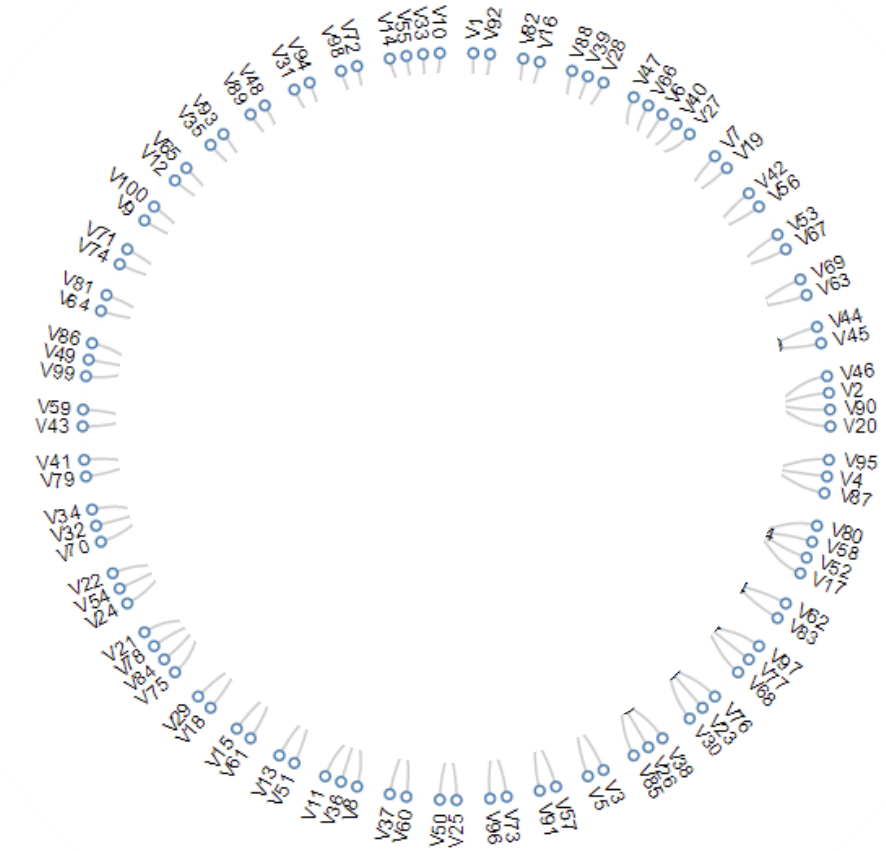
Hierarchical clustering
agglomerative (bottom up)





Hierarchical clustering agglomerative (bottom up)

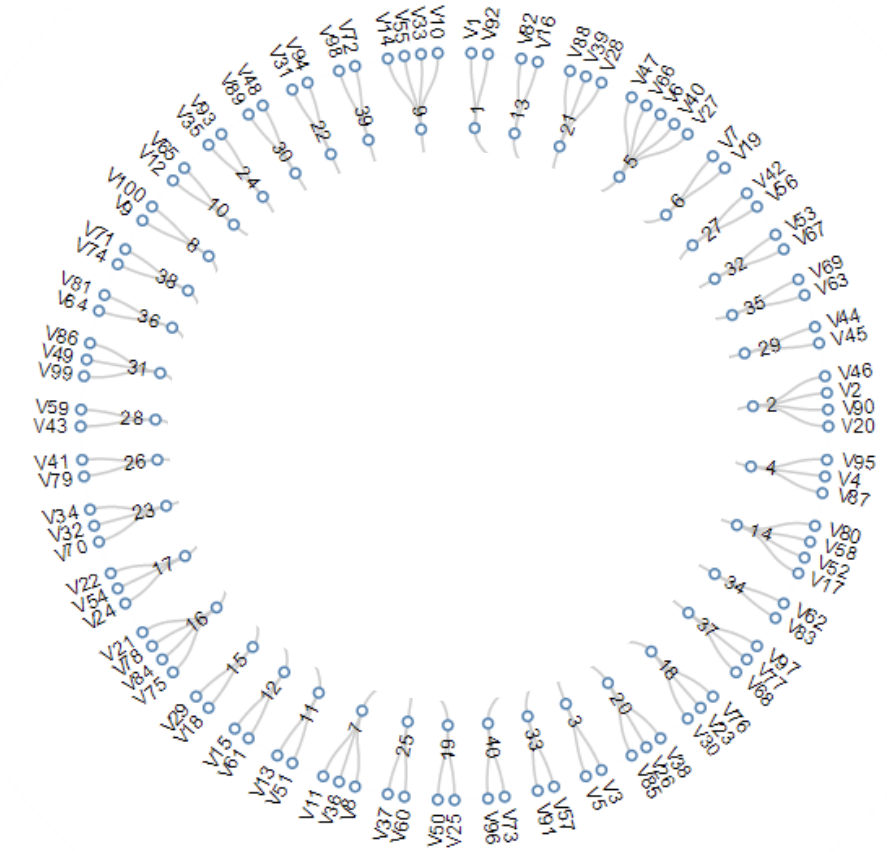
it's deterministic!



Hierarchical clustering agglomerative (bottom up)

it's deterministic!

computationally intense because every ***cluster pair*** distance has to be calculate



Hierarchical clustering agglomerative (bottom up)

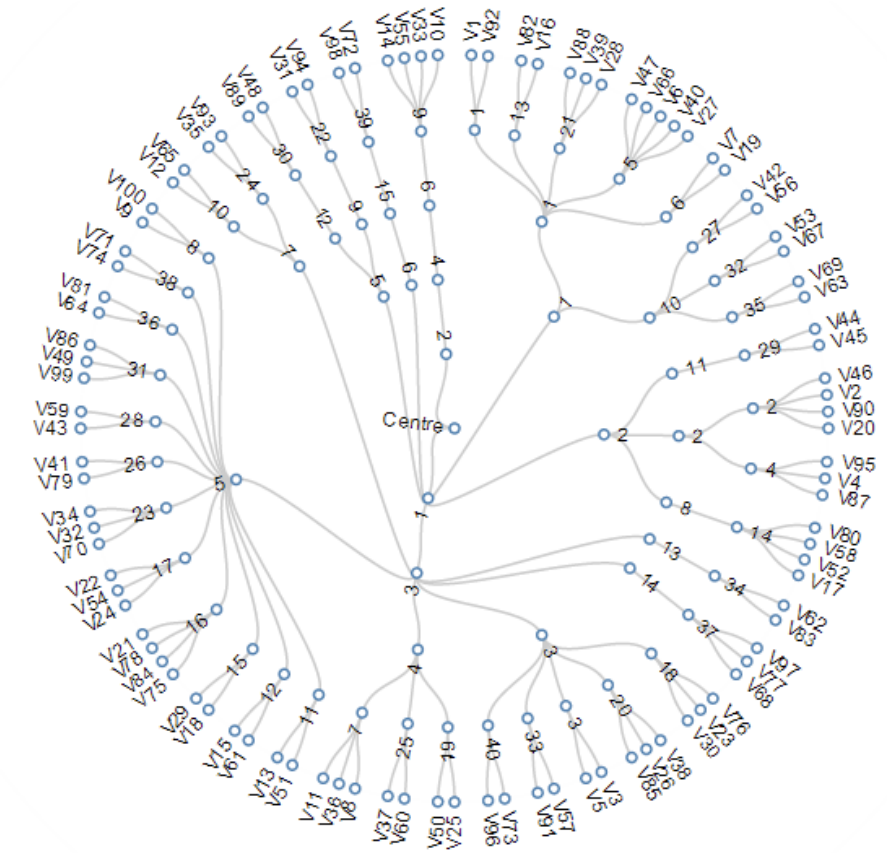
it's deterministic!

computationally intense because every ***cluster pair*** distance has to be calculate

it is slow, though it can be optimize:

complexity

$$O(N^2 d + N^3)$$



Agglomerative clustering: *the algorithm*

```
compute the distance matrix
each data point is a singleton cluster
repeat
    merge the 2 cluster with minimum distance
    update the distance matrix
untill
    only a single (n) cluster(s) remains
```

Agglomerative clustering:

Order: $O(N^2 d + N^3)$

PROs

It's deterministic

CONs

It's greedy (optimization is done step by step and agglomeration decisions cannot be undone)

It's computationally expensive

Agglomerative clustering: *hyperparameters*

class `sklearn.cluster.` **`AgglomerativeClustering`** (*n_clusters=2, affinity='euclidean', memory=None, connectivity=None, compute_full_tree='auto', linkage='ward', pooling_func='deprecated', distance_threshold=None*) ¶ [source]

- *n_clusters* : number of clusters
- *affinity* : the distance/similarity definition
- *linkage* : the scheme to measure distance to a cluster
- *random_state* : for reproducibility

6.2 Divisive hierarchical clustering

Hierarchical clustering
divisive (top down)

Hierarchical clustering
divisive (top down)

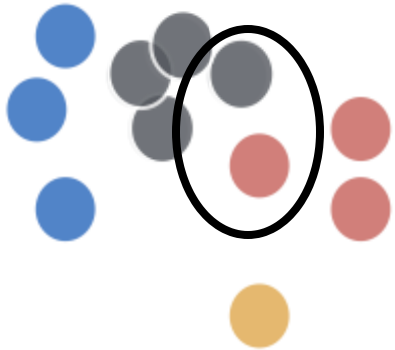
Hierarchical clustering
divisive (top down)

it is
non-deterministic
(like k-mean)

Hierarchical clustering
divisive (top down)

it is
non-deterministic
(like k-mean)

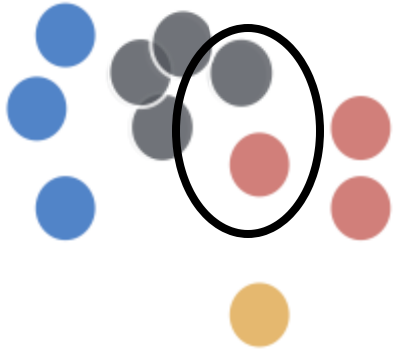
it is greedy -
just as k-means
two nearby points
may end up in
separate clusters



Hierarchical clustering
divisive (top down)

it is
non-deterministic
(like k-mean)

it is greedy -
just as k-means
two nearby points
may end up in
separate clusters



it is high complexity for
exhaustive search
$O(2^N)$
But can be reduced (~k-means)
$O(2Nk)$ or $O(N^2)$

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5751574/

Divisive clustering:
the algorithm

```
Calculate clustering criterion for all subgroups, e.g. min
intracluster variance
repeat
split the best cluster based on criterion above
untill
each data is in its own singleton cluster
```

Divisive clustering:

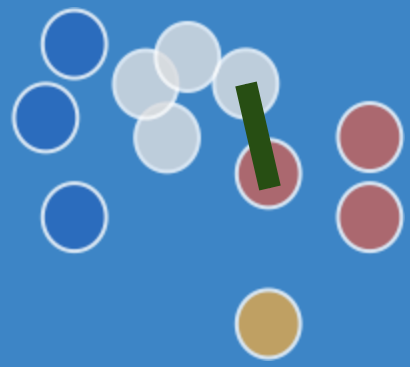
Order: $O(N^2)$ (w K-means procedure)

It's non-deterministic: the result depends on the (random) starting point (like K-mean) unless its exaustive (but that is $O(2^N)$)

or

It's greedy (optimization is done step by step)

linkage:

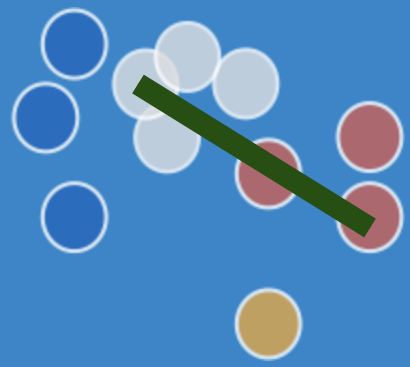


distance between a point and a cluster:

single link distance

D(c1,c2) = min(D(xc1, xc2))

linkage:



distance between a point and a cluster:

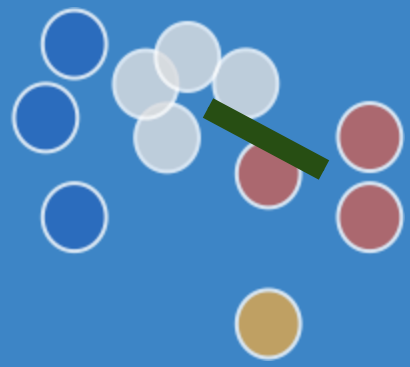
single link distance

D(c1,c2) = min(D(xc1, xc2))

complete link distance

D(c1,c2) = max(D(xc1, xc2))

linkage:



distance between a point and a cluster:

single link distance

$D(c1,c2) = \min(D(x_{c1}, x_{c2}))$

complete link distance

$D(c1,c2) = \max(D(x_{c1}, x_{c2}))$

centroid link distance

$D(c1,c2) = \text{mean}(D(x_{c1}, x_{c2}))$

linkage:

distance between a point and a cluster:

single link distance

D(c1,c2) = min(D(x_{c1}, x_{c2}))

complete link distance

D(c1,c2) = max(D(x_{c1}, x_{c2}))

centroid link distance

D(c1,c2) = mean(D(x_{c1}, x_{c2}))

Ward distance (global measure)

$$D_{tot} = \sum_j \sum_{i,x \in C_j} (x_i - \mu_j)^2$$

Density Based

DBSCAN

7

DBSCAN

Density-based spatial clustering of applications with noise

DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature

DBSCAN

defines cluster membership based on local density: based on Nearest Neighbors algorithm.

DBSCAN:
the algorithm

- A point *p* is a *core point* if at least minPts points are within distance *ε* (including *p*).
- A point *q* is *directly reachable* from *p* if point *q* is within distance *ε* from core point *p*. *Reachable* from *p* if there is a path *p*1, ..., *pn* with *p*1 = *p* and *pn* = *q*, where each *pi*+1 is directly reachable from *pi*.
- All points not reachable from any other point are *outliers* or *noise points*.

DBSCAN:

the algorithm

class `sklearn.cluster.` **`DBSCAN`** (*eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*) [source]

- ***ε*** : minimum distance to join points
- **min_sample** : minimum number of points in a cluster, otherwise they are labeled outliers.
- **metric** : the distance metric
- **p** : float, optional The power of the Minkowski metric

DBSCAN:

the algorithm

class `sklearn.cluster.` **`DBSCAN`** (*eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*) [source]

- ***ε*** : minimum distance to join points
- **min_sample** : minimum number of points in a cluster, otherwise they are labeled outliers.
- **metric** : the distance metric
- **p** : float, optional The power of the Minkowski metric

its extremely sensitive to these parameters!

DBSCAN:

the algorithm

```
for each point P count neighbours within minPts: label=C
for each point P ~= C measure distance d to all Cs
        if d<minD: label = DR
for each point P not C and not DR
        if distance d to C or DR > minD: label = outlier
        if distance d to C or DR <= minD: find path to
closet C and cluster
```

DBSCAN clustering:

Order: $O(N^2)$

PROs

Deterministic.

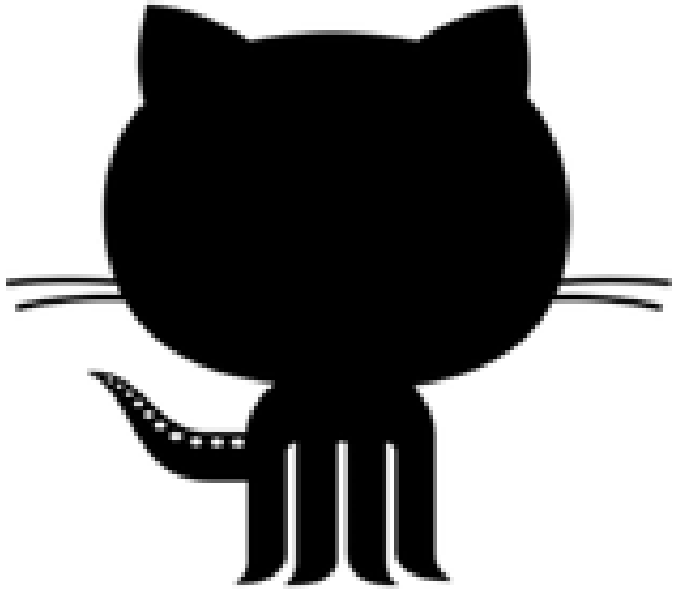
Deals with noise and outliers

Can be used with any definition of distance or similarity

PROs

Not entirely deterministic.

Only works in a constant density field



https://github.com/fedhere/DSPS/blob/master/lab10/StellarPopClustersLab.ipynb

Clustering : unsupervised learning where all features are observed for all datapoints. The goal is to partition the space into maximally homogeneous maximally distinguished groups

clustering is easy, but interpreting results is tricky

Distance : A definition of distance is required to group observations/ partition the space.

Common distances over continuous variables

- Minkowski (inlcudes Euclidian = Minkowski(2)
- Great Circle (for coordinated on a sphere, e.g. earth or sky)

Common distances over categorical variables:

- Simple Distance Matrix
- Jaccard Distance

Whitening Models assume that the data is not correlated. If your data is correlated the model results may be invalid. And your data always has correlations.

Solution: - whiten the data by using the matrix that diagonalizes the covariance matrix. This is ideal but computationally expensive if possible at all

- scale your data so that each feature is mean=0 stdev=2.

Partition clustering:

Hard: K-means *O(KdN)* , needs to decide the number of clusters, non deterministic simple efficient implementation but the need to select the number of clusters is a significant flaw

Soft: Expectation Maximization *O(KdNp)* , needs to decide the number of clusters, need to decide a likelihood function (parametric), non deterministic

Hierarchical:

Divisive: Exhaustive $O(2^N)$; $O(N^2)$ at least non deterministic

Agglomerative: $O(N^2d + N^3)$**,** deterministic, greedy. Can be run through and explore the best stopping point. Does not require to choose the number of clusters a priori

Density based

DBSCAN: Density based clustering method that can identify outliers, which means it can be used in the presence of noise. Complexity $O(N^2)$. Most common (cited) clustering method in the natural sciences.

encoding categorical variables:

variables have to be encoded as numbers for computers to understand them. You can encode categorical variables with integers or floating point but you implicitly impart an order. The standard is to ***one-hot-encode*** which means creating a binary (True/False) feature (column) for each category of a categorical variables but this *increases the feature space and generated covariance.*

model diagnostics for classifiers: Fraction of True Positives and False Positives are the metrics to evaluate classifiers. Combinations of those numbers include Accuracy (TP/ (TP+FP)), Precision (TP/(TP+FN)), Recall ((TP+TN)/(TP+TN+FP+FN)).

ROC curve: (TP vs FP) is a holistic metric of a model. It can be used to guide the choice of hyperparameters to find the "sweet spot" for your problem

a comprehensive review of clustering methods

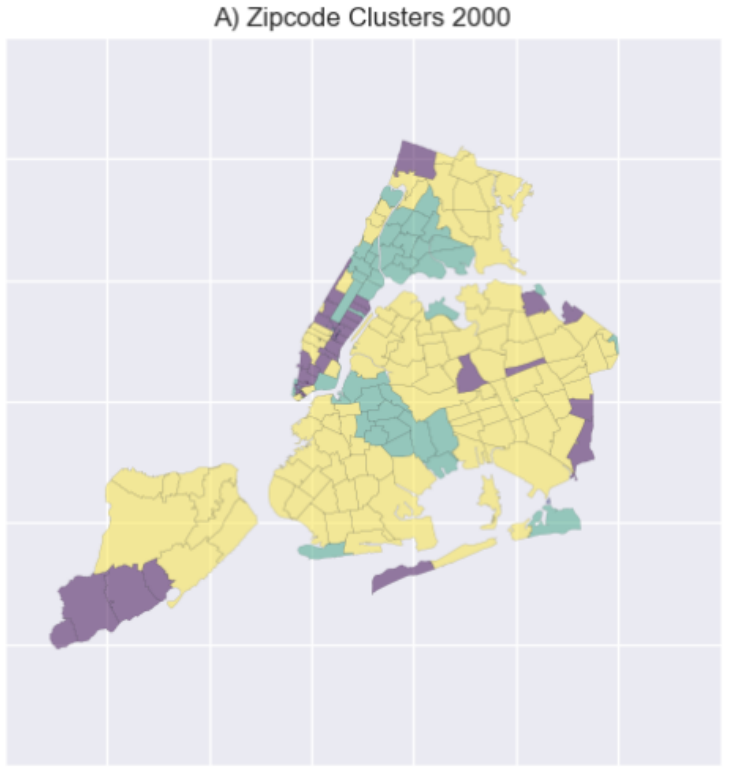
Data Clustering: A Review, Jain, Mutry, Flynn 1999

https://www.cs.rutgers.edu/~mlittman/courses/lightai03/jain99data.pdf

required reading: a blog post on how to generate and interpret a scipy dendrogram by Jörn Hees

https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/

Motivation:



One of the things that make NYC so interesting to study is its diversity on many axes: ethnic, economic, cultural. Neighborohoods tend to be segregated and clustered on most socioeconomic characteristics, however these characteristics do evolve in time. The goal is to identify NYC neighborhoods that are homogeneous by clustering their socioeconomic characteristics in 2000, and in 2010, and find the neighborhoods that change cluster, indicating that they evolve differently with respect to their cluster peers (e.g. gentrification).

https://github.com/fedhere/PUS2020_FBianco/tree/master/HW11

Visualization review

- Work on this alone

You will recieve an email with instruction. Review their plots according to the things we discussed last lecture. You can discuss your review with others but each of you should submit a review for each of the plots

There will be detailed instructions in the email on how to review, what structure the review should have, what to focus on, etc. Please comply to the instructions. Upload the review on your github DSPS HW9 repo AND ALSO in your classmate's HW8 repo, forking and submitting a pull request, NOT JUST your fork of their repo! Please note the numbers: I will grade what is in your HW9 repo and check that it is also in your classmate HW8 repo.

Each review will be reviewed and graded by me. (Please take this homework seriously: one sentence generic reviews will be graded 0)