Lecture 05:

Unsupervised Learning

Markus Hohle

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

Course Map

Lecture 1: Course Overview and Introduction to Machine Learning

Lecture 2: Bayesian Methods in Machine Learning

Lecture 3: Dimensionality Reduction: Principal Component Analysis

Lecture 4: Linear and Non-linear Regression and Classification

Lecture 5: Unsupervised Learning: K-Means, GMM, Trees

Lecture 6: Adaptive Learning and Gradient Descent Optimization Algorithms

classic ML tools & algorithms

Lecture 7: Introduction to Artificial Neural Networks - The Perceptron

Lecture 8: Introduction to Artificial Neural Networks - Building Multiple Dense Layers

Lecture 9: Convolutional Neural Networks (CNNs) - Part I

Lecture 10: CNNs - Part II

Lecture 11: Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs)

Lecture 12: Combining LSTMs and CNNs

Lecture 13: Running Models on GPUs and Parallel Processing

Lecture 14: Project Presentations

Lecture 15: Transformer

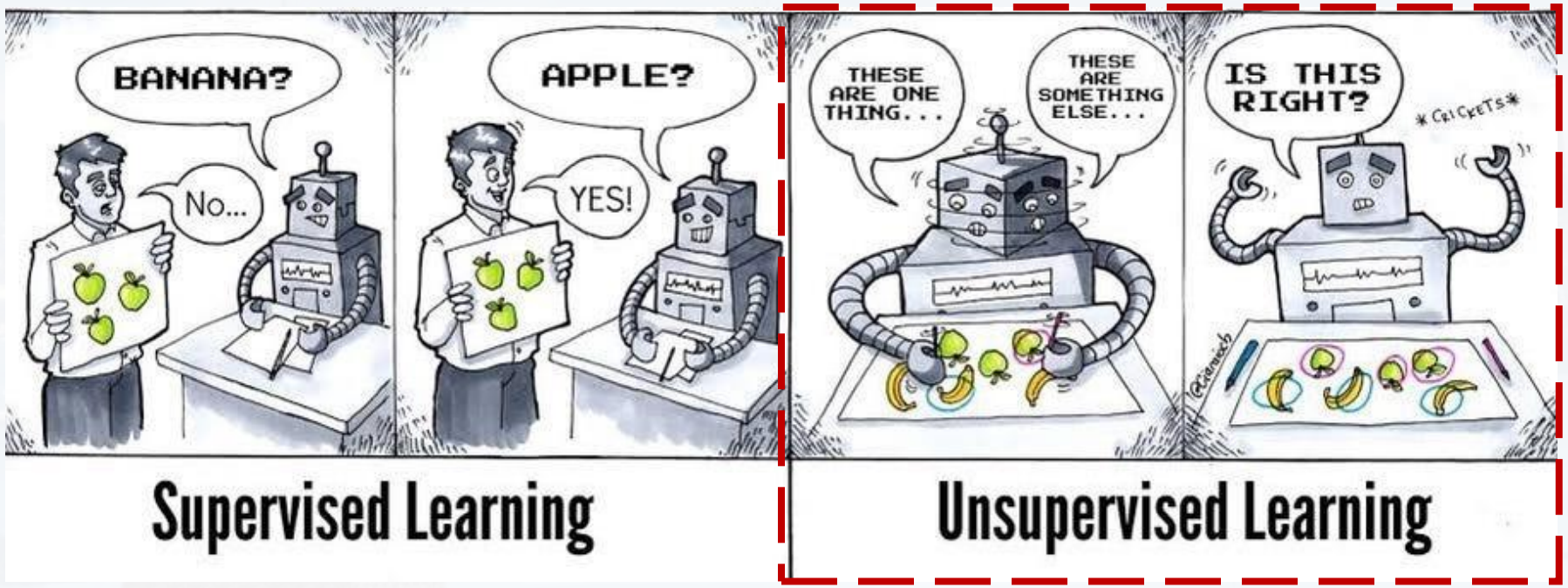
Lecture 16: GNN

ANNs/AI/Deep Learning

Machine Learning Algorithms: Unsupervised Learning

So far, there has been a **training** data set and a **test** data set…

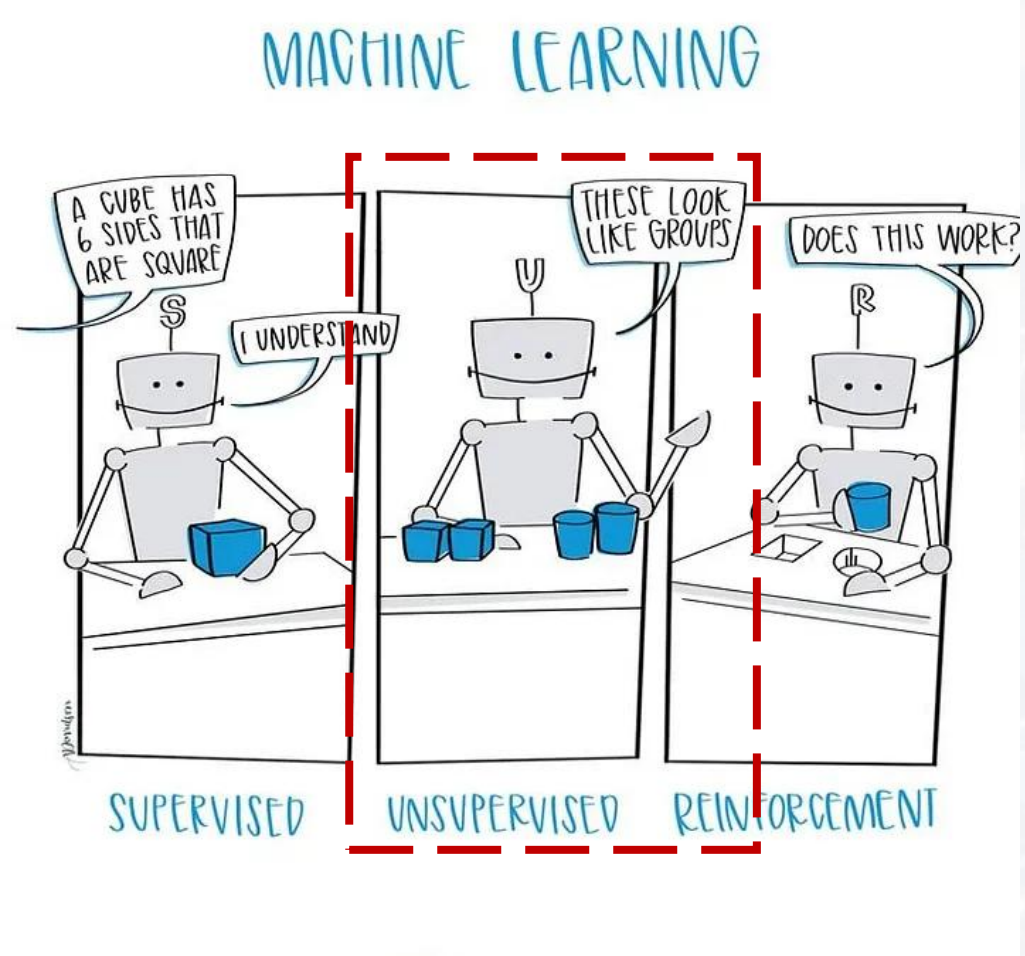
… but maybe there are ways to learn *without* training data



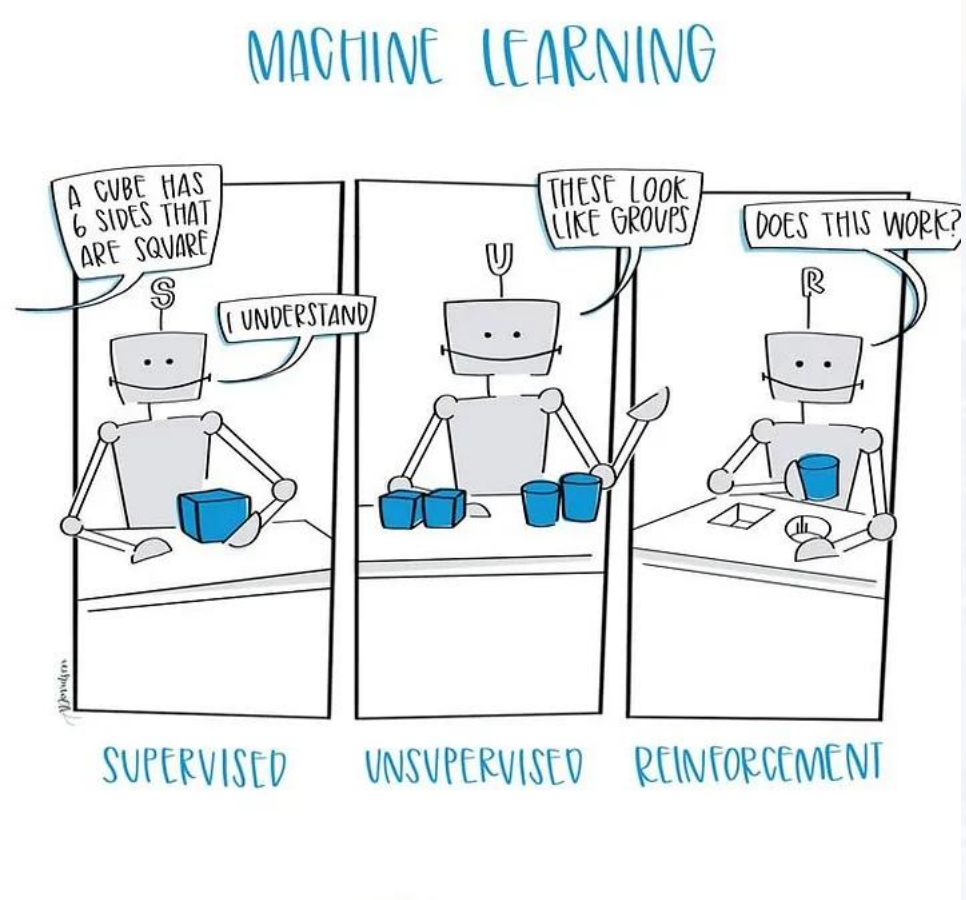
Machine Learning Algorithms: Unsupervised Learning

So far, there has been a **training** data set and a **test** data set…

… but maybe there are ways to learn *without* training data



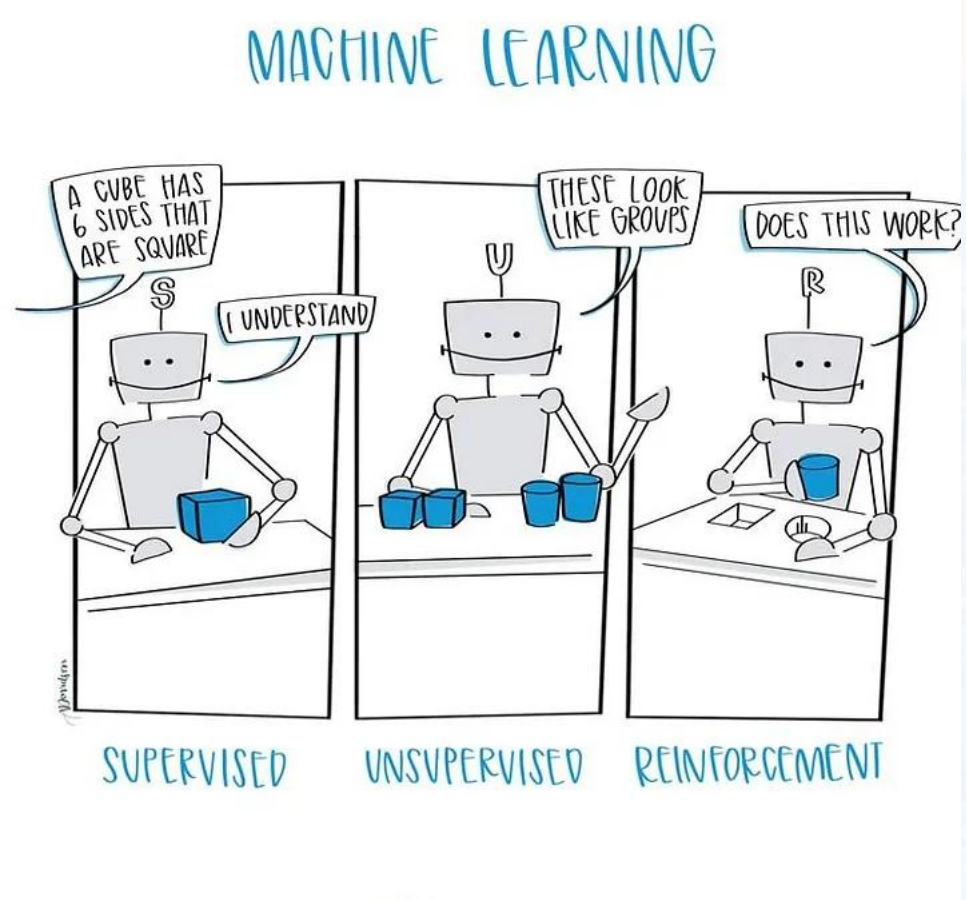
Machine Learning Algorithms:



Outline

- K - means
- GMM
- trees

Machine Learning Algorithms:



Outline

- **K - means**
- GMM
- trees

Machine Learning Algorithms: K - Means

<u>idea:</u> finding the barycenter of a cluster and thereby assign the datapoints accordingly

K	**: number of cluster**
N	**: number of observations**

goal: minimizing

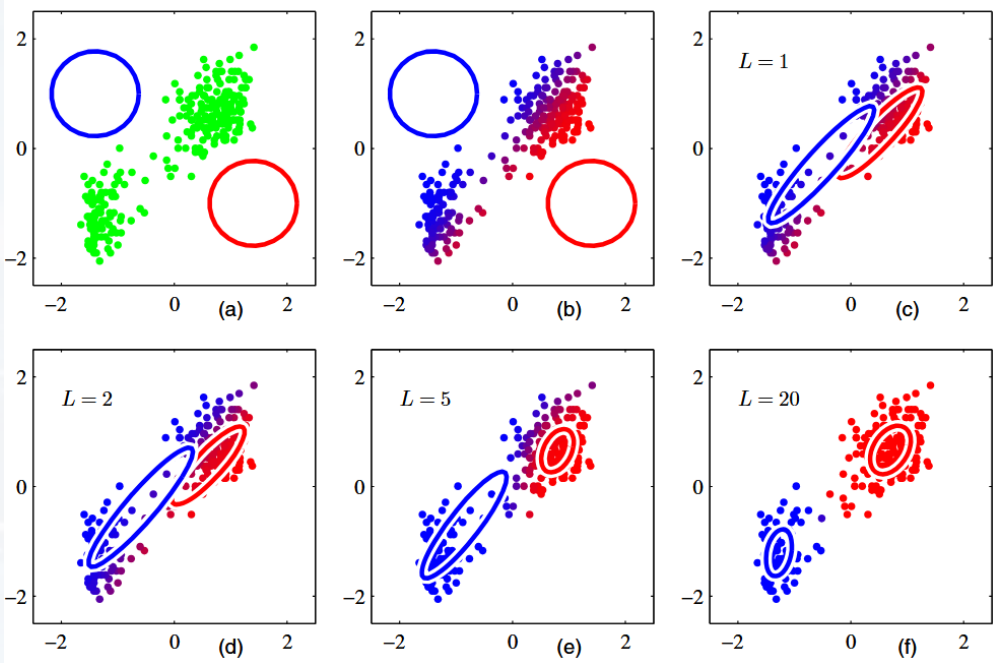
$$\boxed{J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \; \|x_n - \mu_k\|^2}$$

μ_k: barycenter of cluster k

indicator function $r_{n,k} \in \{0, 1\}$

assigning x_n to its closed mean

$$r_{n,k} = \begin{cases} 1, & if\ k = \ argmin_j \|x_n - \mu_j\|^2 \\ 0, & else \end{cases}$$



$$\frac{\partial J}{\partial \mu_k} = 0 \longrightarrow \mu_k = \frac{\sum_{n=1}^{N} r_{n,k} x_n}{\sum_{n=1}^{N} r_{n,k}}$$

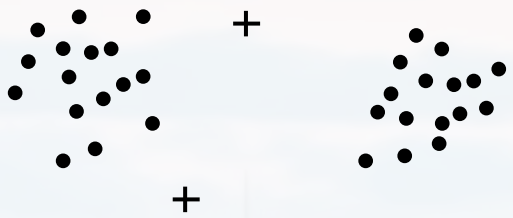
K – means is an iterative process

idea:

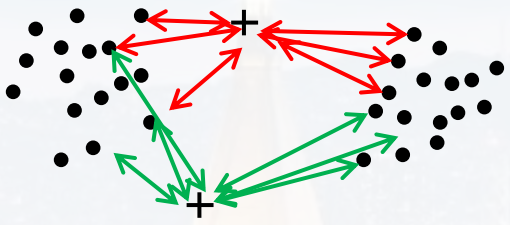
K	: number of cluster
N	: number of observations
μ_k	: barycenter of the cluster

$$r_{n,k} = \begin{cases} 1, & if\ k = argmin_j \|x_n - \mu_j\|^2 \\ 0, & else \end{cases}$$

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \ \|x_n - \mu_k\|^2$$



a) assign *k* means randomly



b) calculate *distance* from each point to each mean

c) assign each point to its closest mean

d) update the means accordingly

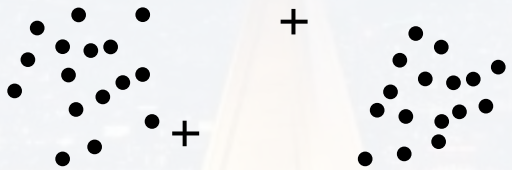
idea:

K	: number of cluster
N	: number of observations
μ_k	: barycenter of the cluster

$$r_{n,k} = \begin{cases} 1, & if\ k = argmin_j \|x_n - \mu_j\|^2 \\ 0, & else \end{cases}$$

d) update the means accordingly

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} \ \|x_n - \mu_k\|^2$$



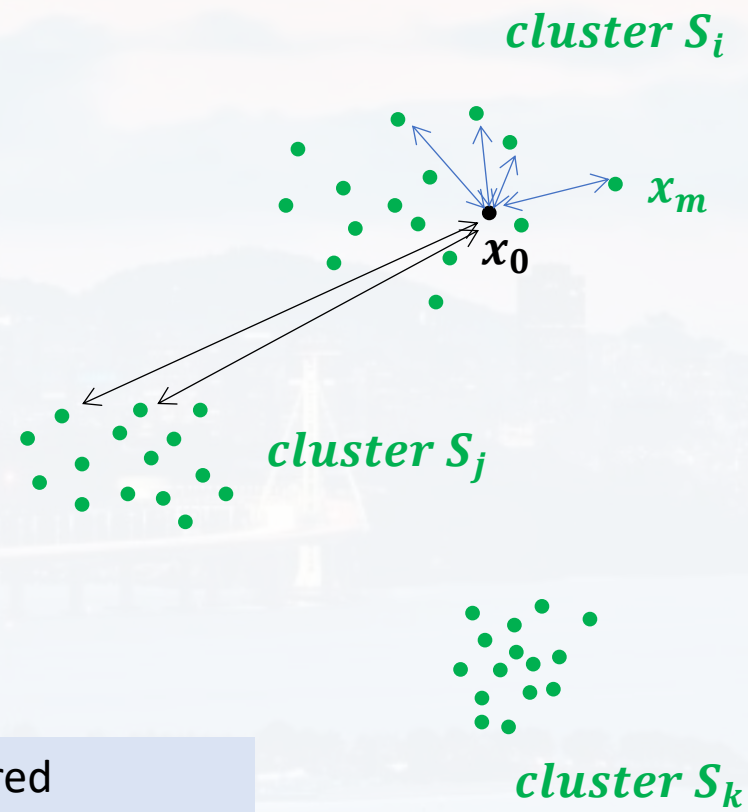
e) go back to b)

problem: ***k = number of cluster,*** is a hyperparameter. How do I know the correct value for ***k***?

→ silhouette Ψ

- distance $\boldsymbol{d_1}$ of a data point x_0 to ***its assigned cluster*** $\boldsymbol{S_i}$
  vs distance $\boldsymbol{d_2}$ to ***closest cluster (here*** $\boldsymbol{S_j}$***)***

$$\Psi(x_0) = \begin{cases} 0 & if\ d_1 = 0 \\ \dfrac{d_2 - \boldsymbol{d_1}}{max[\boldsymbol{d_1}; d_2]} & \end{cases}$$



- average over all points → ψ_{tot}

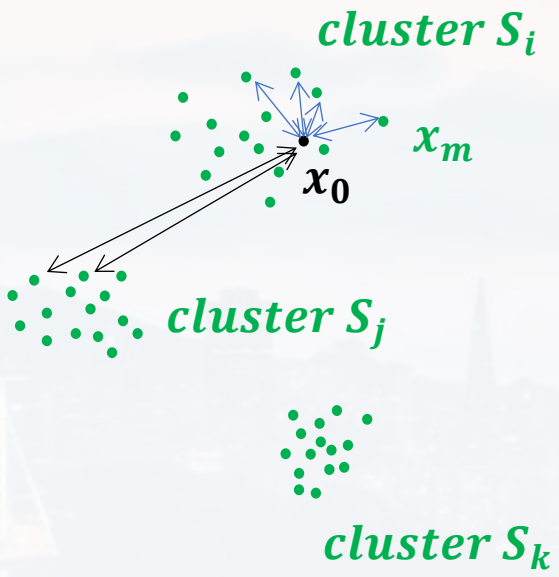
if	$\psi_{tot} = 0.75 \ldots 1.00$	→ well clustered
	$\psi_{tot} = 0.50 \ldots 0.75$	→ medium clustered
	$\psi_{tot} = 0.25 \ldots 0.50$	→ poorly clustered
	$\psi_{tot} < 0.25$	→ data has no structure

problem: ***k*** is a hyperparameter. How do I know the correct value for ***k***?

→ silhouette Ψ

- distance d_1 of a data point x_0 to ***its assigned cluster*** $\boldsymbol{S_i}$
  vs distance d_2 to ***closest cluster (here*** $\boldsymbol{S_j}$***)***

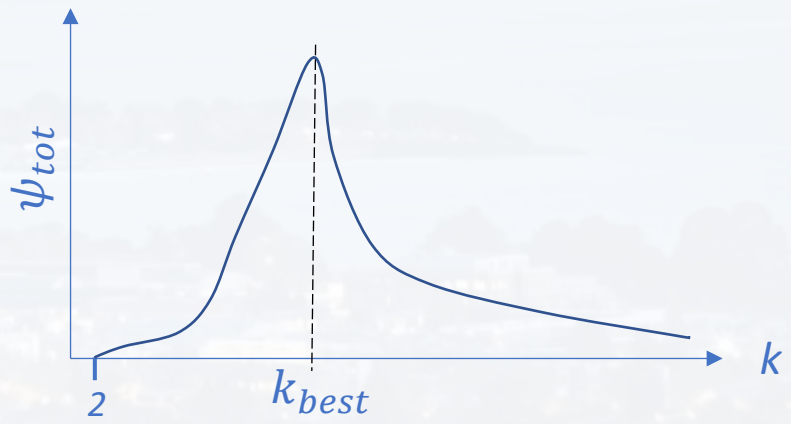
$$\Psi(x_0) = \begin{cases} 0 & if\ d_1 = 0 \\ \dfrac{d_2 - d_1}{max[d_1; d_2]} & \end{cases}$$



- average over all points → ψ_{tot}

$\psi_{tot} = 0.75$	… 1.00	→ well clustered
$\psi_{tot} = 0.50$	… 0.75	→ medium clustered
$\psi_{tot} = 0.25$	… 0.50	→ poorly clustered
$\psi_{tot} < 0.25$		→ data has no structure

ideal world →



Machine Learning Algorithms:

K - Means

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

from pyclustering.utils.metric import *
from nltk.cluster.kmeans import KMeansClusterer
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn import datasets
```

our standard libraries

for having different distances available

calling the famous “iris” data set

calculating silhouette coefficient for different k

performing k-means

Machine Learning Algorithms: K - Means

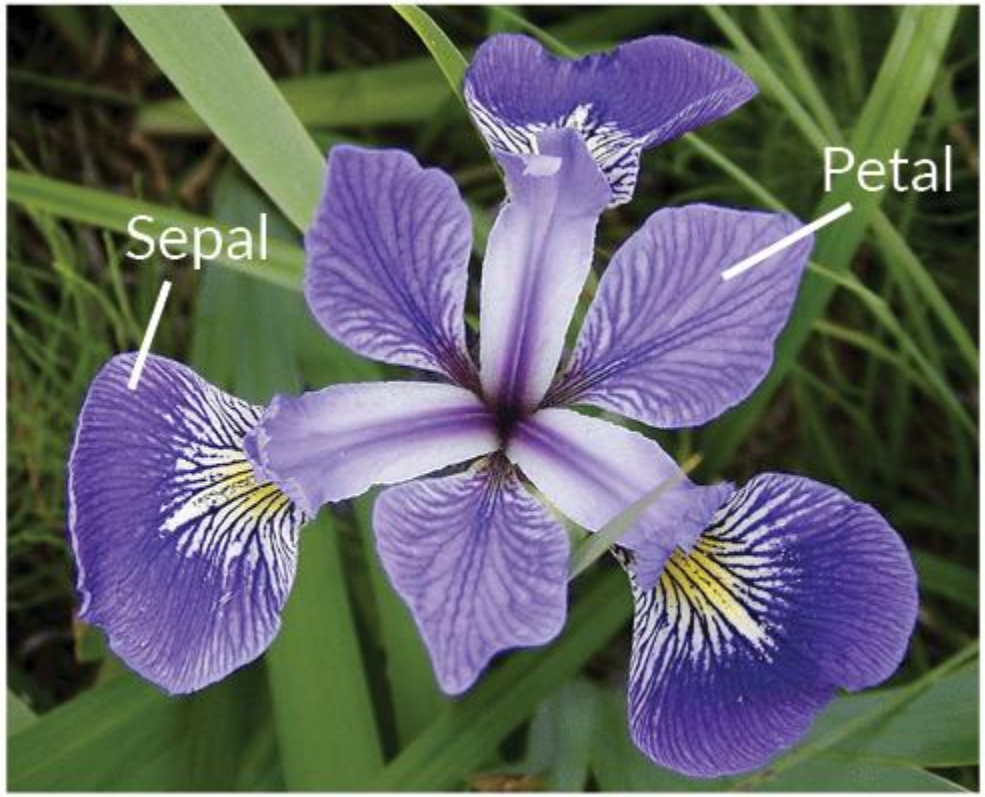
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

from pyclustering.utils.metric import *
from nltk.cluster.kmeans import KMeans(
from sklearn.metrics import silhouette_
from sklearn import datasets

iris = datasets.load_iris()

iris.DESCR
```

```
Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================
```

ideal world: three distinct cluster

Machine Learning Algorithms:


```
iris = datasets.load_iris()

iris.DESCR
```



Iris Versicolor

Iris Setosa

Iris Virginica

Machine Learning Algorithms: K - Means

loading & exploring the data:

```
iris = datasets.load_iris()

iris.DESCR
iris.feature_names
iris.target_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

four features → 4D

```
array(['setosa', 'versicolor', 'virginica']
```

check out the Jupyter Notebook **Walk_Through_Kmeans**

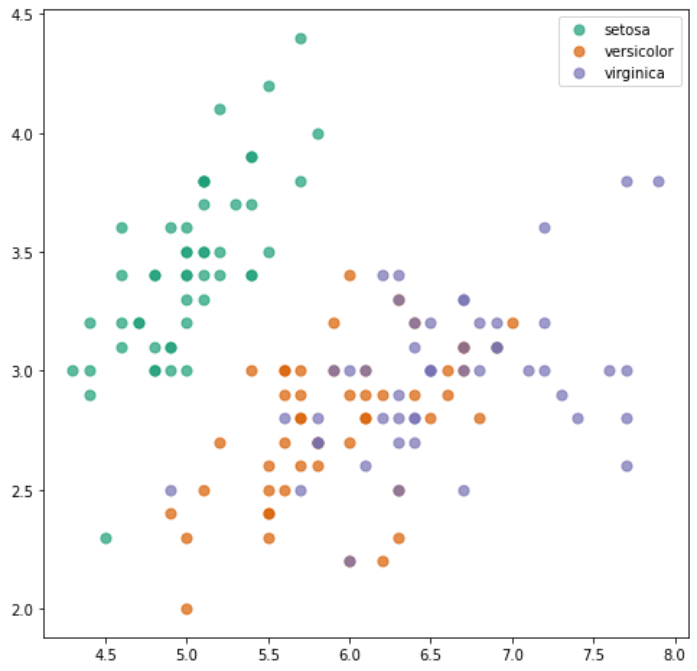
- plotting the data
- running k-means
- evaluating the result

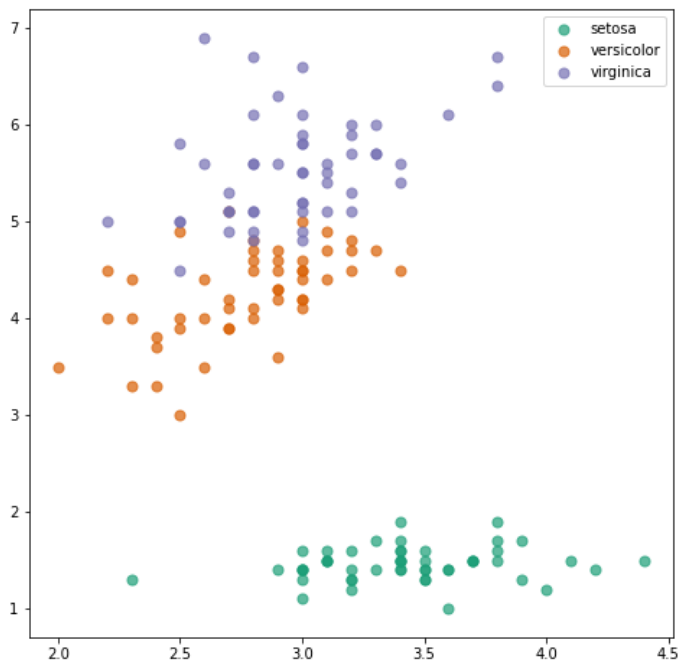
Machine Learning Algorithms: K - Means

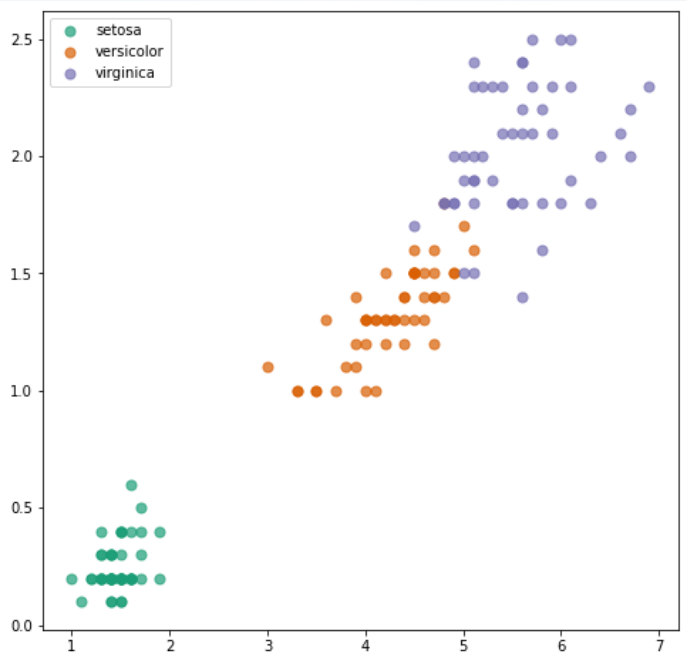
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

4D dataset → plotting two components

- **plotting the data**
- running k-means
- evaluating the result







Machine Learning Algorithms: K - Means

```
nClust      = 3
rep         = 25
dist        = distance_metric(type_metric.EUCLIDEAN)


my_model    = KMeansClusterer(nClust, distance = dist,\
                  repeats   = rep,\
                  avoid_empty_clusters = True)

PredLabels = my_model.cluster(X2D,\
                  assign_clusters = True)

Center      = my_model.means()
```

- plotting the data
- **running k-means**
- evaluating the result

we need to “guess” the number of cluster

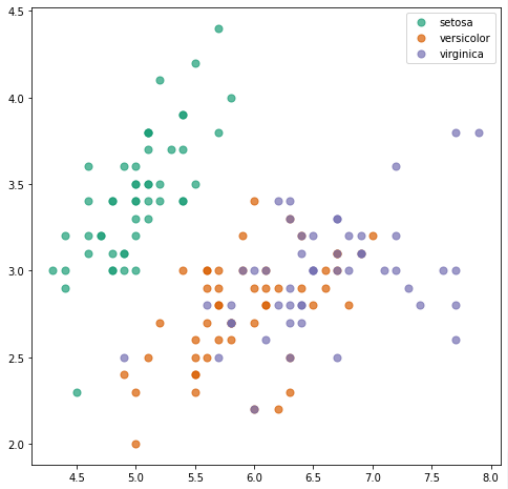
the **initial** means are assigned randomly.
→ repeat the procedure 25 times
→ avoiding local minimum,

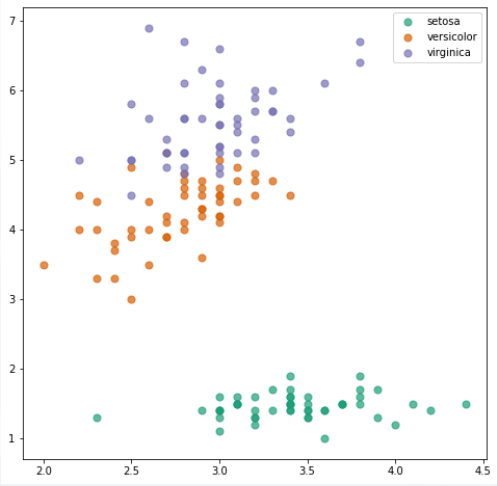
The features are meassured in cm, i. e. the correct distance to pick here is Euclidean

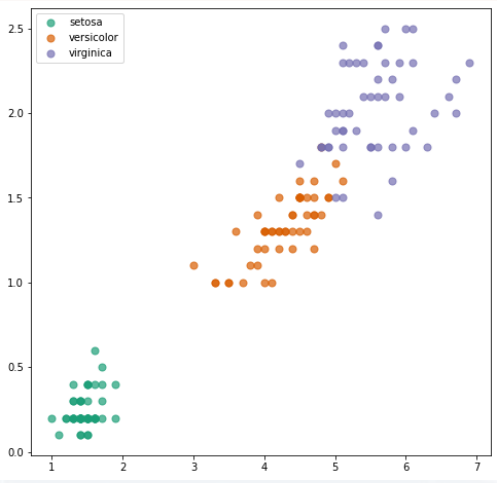
Machine Learning Algorithms: K - Means

- plotting the data
- **running k-means**
- evaluating the result

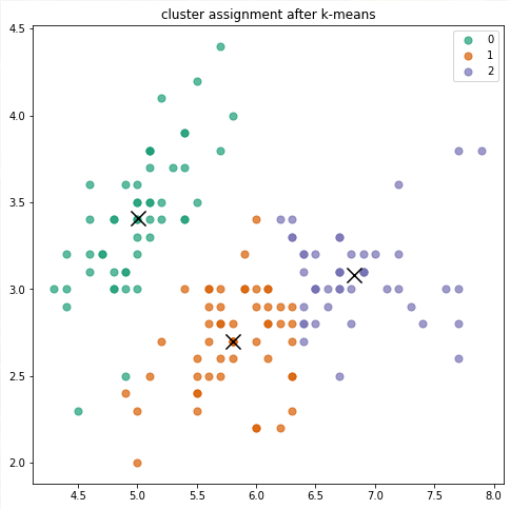
true classes

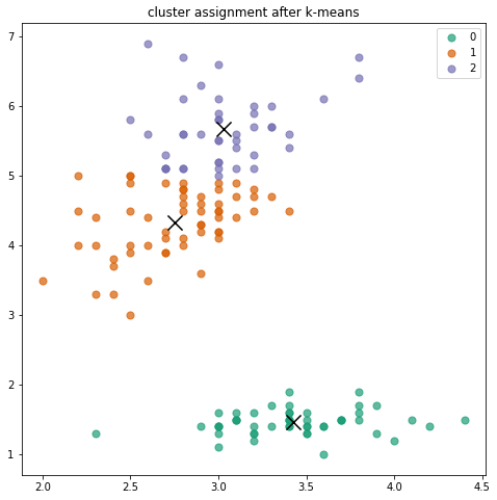


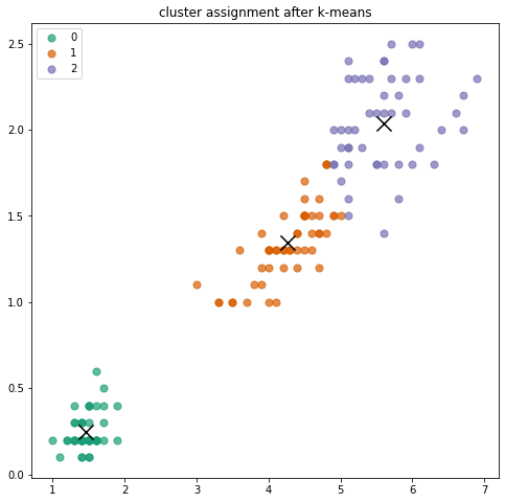




assigned classes





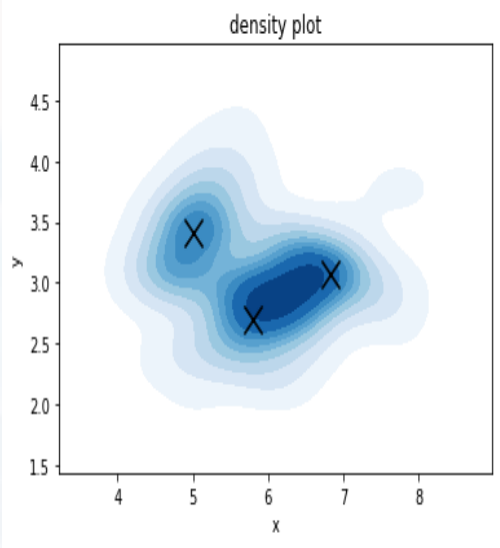


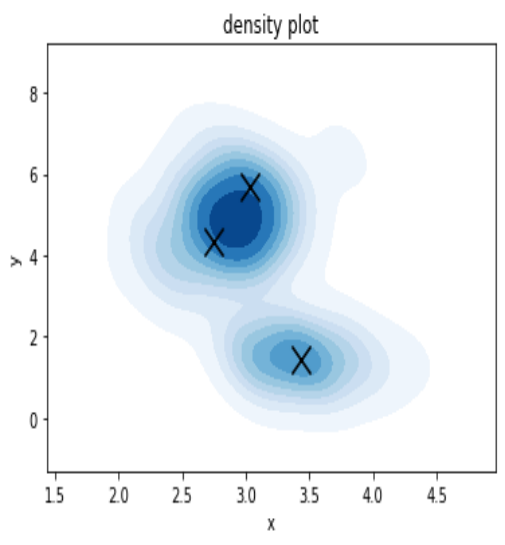
Machine Learning Algorithms:

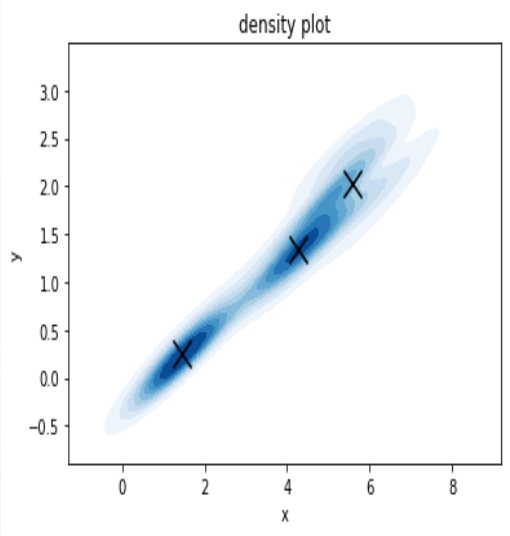
K - Means

- plotting the data
- **running k-means**
- evaluating the result

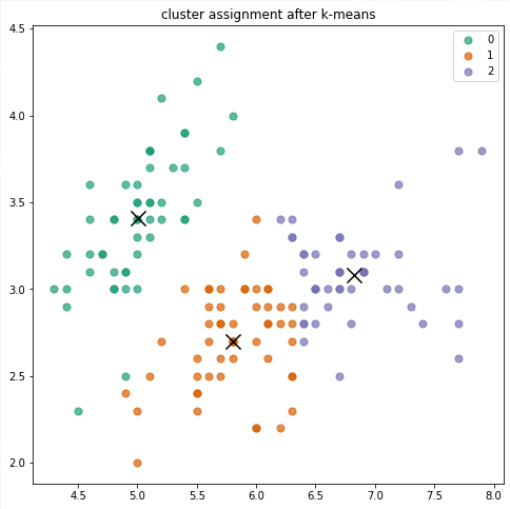
density

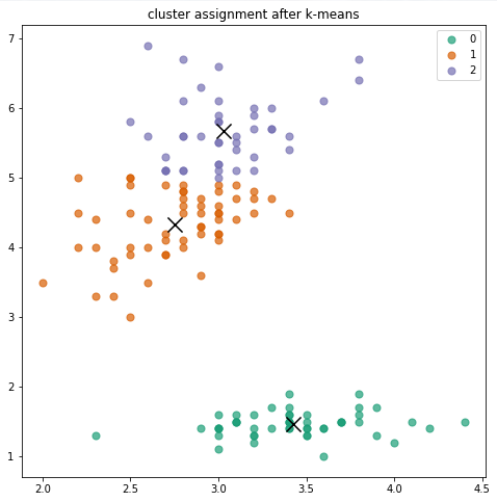


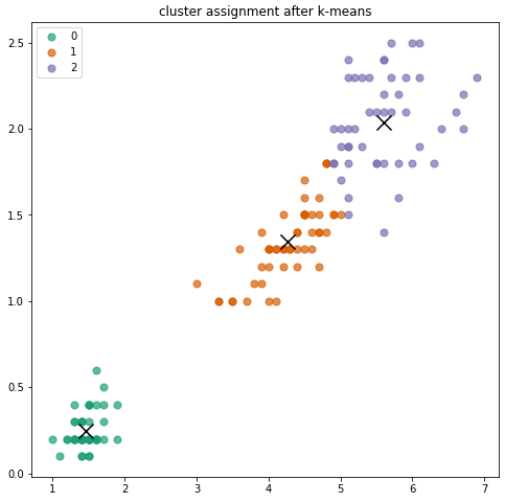




assigned classes





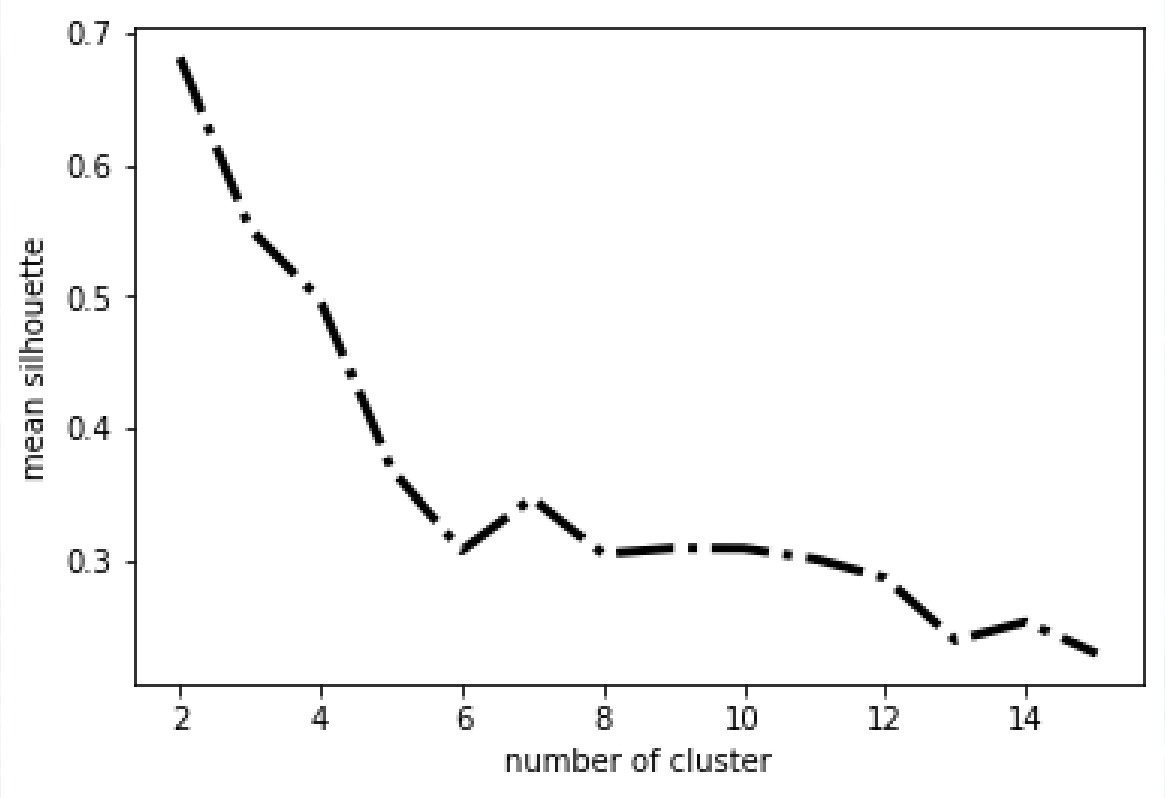


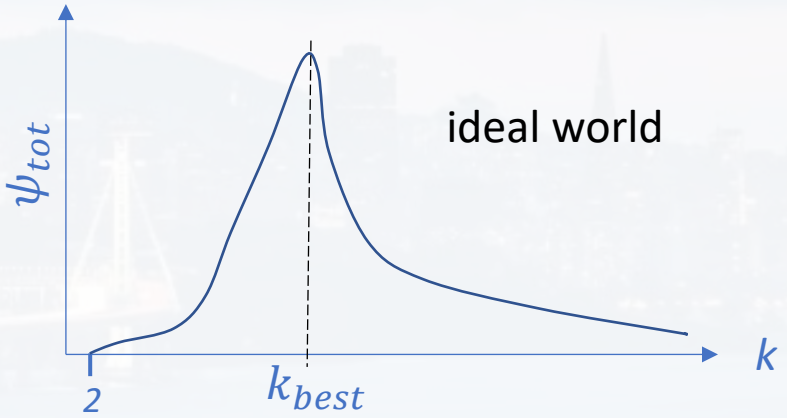
Machine Learning Algorithms:

- plotting the data
- running k-means
- **evaluating the result**

we run k-means now for the **full 4D** dataset
+ evaluate clustering with silhouette

```
silhouette_score(X, Labels)
```



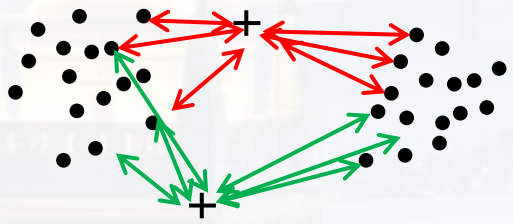


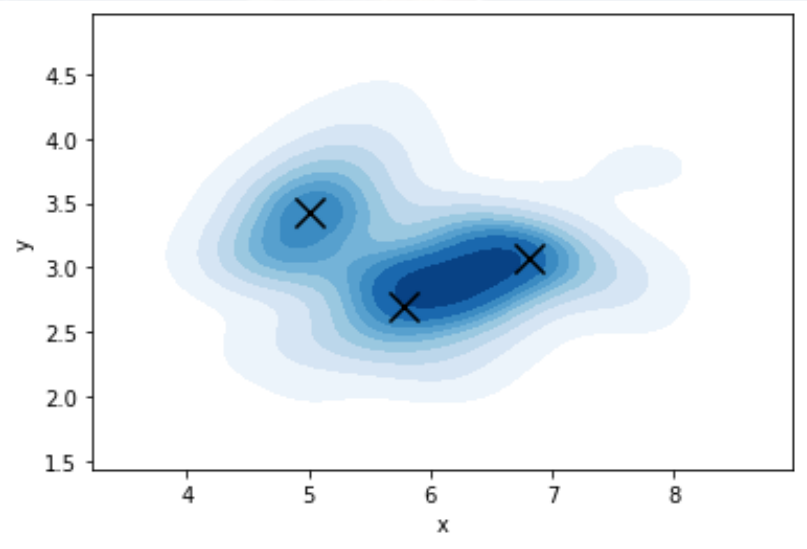
accuracy for 4D k = 3: 90%

summary:

- simple and fast
- unsupervised
- *k* has to be given → silhouette for determining best *k*
- problems if cluster have unusual shapes

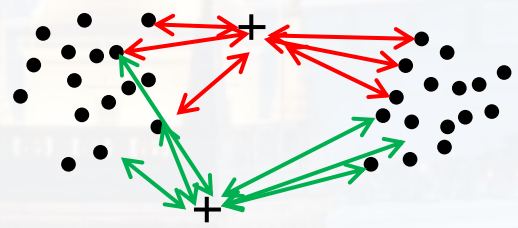
(elongated, hollow inside, scattered)

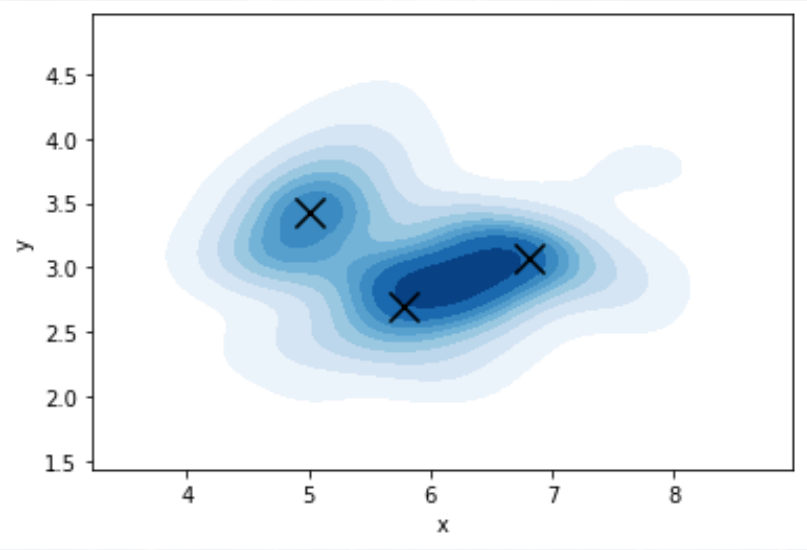




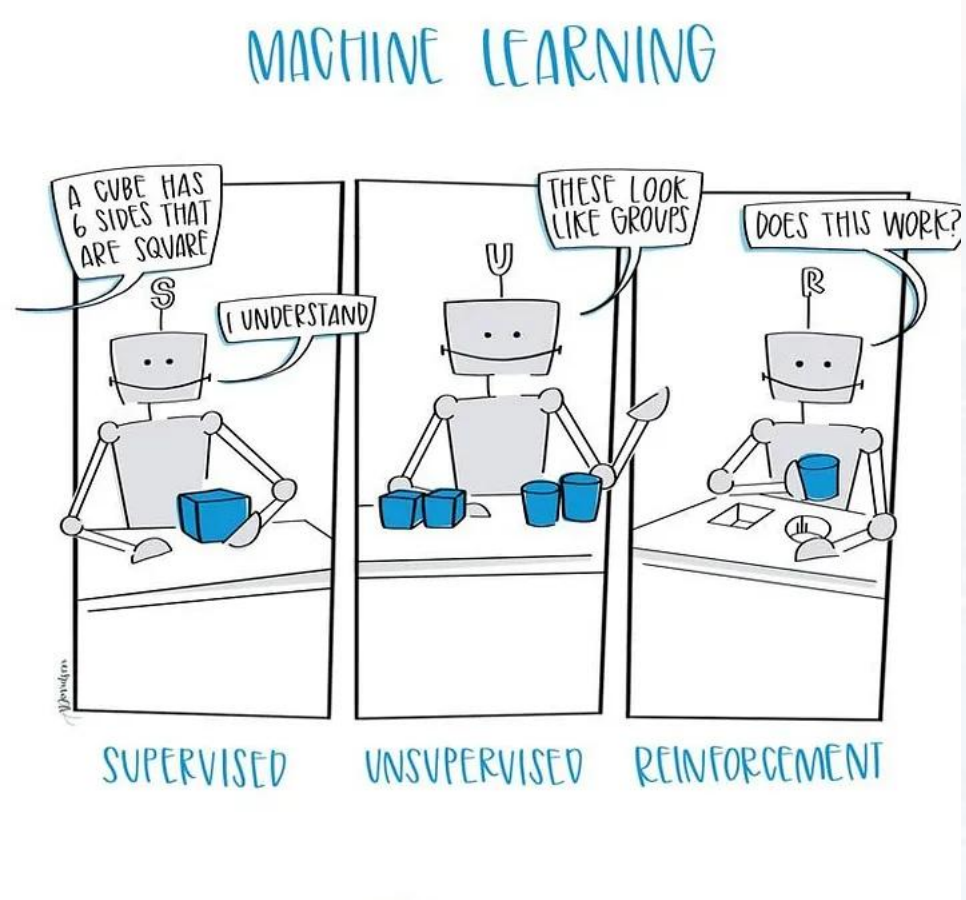
<u>topics for the discussion/office hour:</u>

- What is a *distance*?
- Which are different distances?
- When to use which distance?





Machine Learning Algorithms:



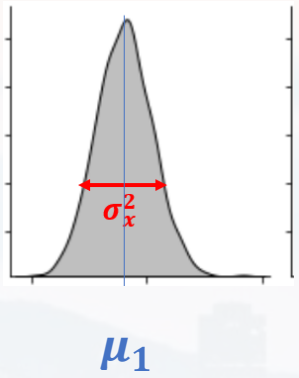
Outline

- K - means
- **GMM**
- trees

Gaussian **M**ixture **M**odels

<u>one feature</u>

$$N_1(x_1) = \frac{1}{\sqrt{2\pi\,\sigma_{x1}^2}} \exp\left\{-\frac{1}{2}\left(\frac{x_1 - \mu_1}{\sigma_{x1}}\right)^2\right\}$$

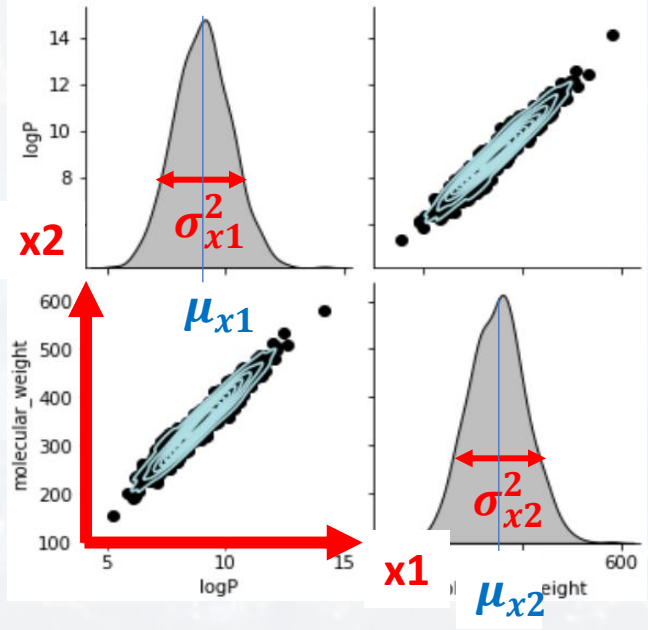


<u>two features</u>

$$\Sigma = \begin{pmatrix} \sigma_{x1}^2 & cov(x_1, x_2) \\ cov(x_2, x_1) & \sigma_{x2}^2 \end{pmatrix}$$ covariance matrix

$$\begin{pmatrix} x_1 - \mu_{x1} \\ x_2 - \mu_{x2} \end{pmatrix}^T \Sigma^{-1} \begin{pmatrix} x_1 - \mu_{x1} \\ x_2 - \mu_{x2} \end{pmatrix}$$ see PCA lecture

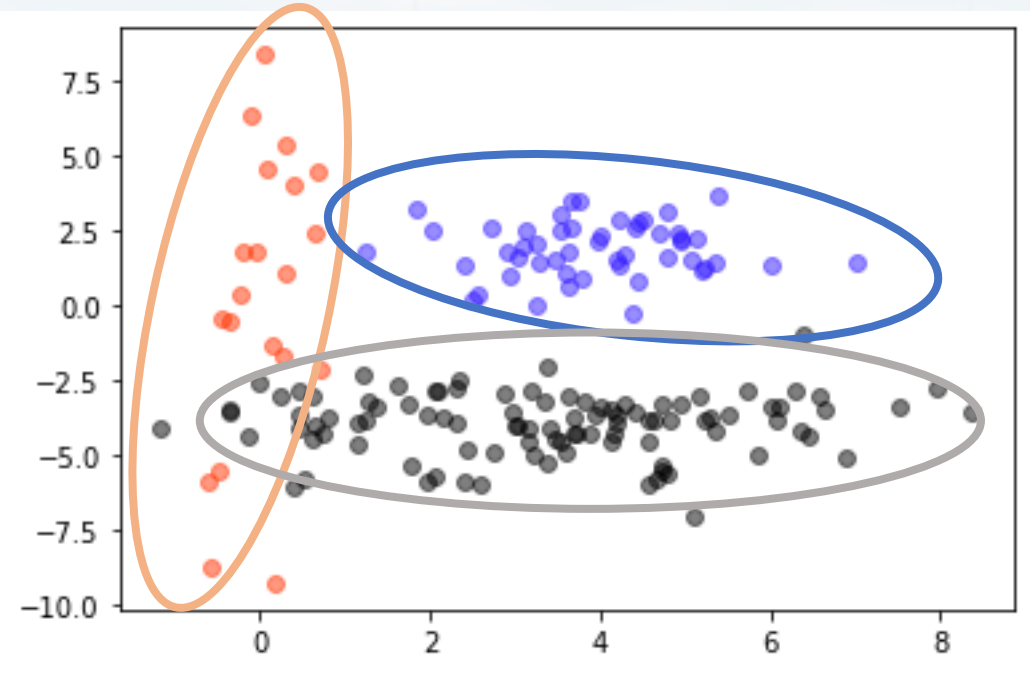
$$N_2(x_1, x_2) = \frac{1}{2\pi\,\det(\Sigma)^{1/2}} \exp\left\{-\frac{1}{2}\left[\begin{pmatrix} x_1 - \mu_{x1} \\ x_2 - \mu_{x2} \end{pmatrix}^T \Sigma^{-1} \begin{pmatrix} x_1 - \mu_{x1} \\ x_2 - \mu_{x2} \end{pmatrix}\right]\right\}$$



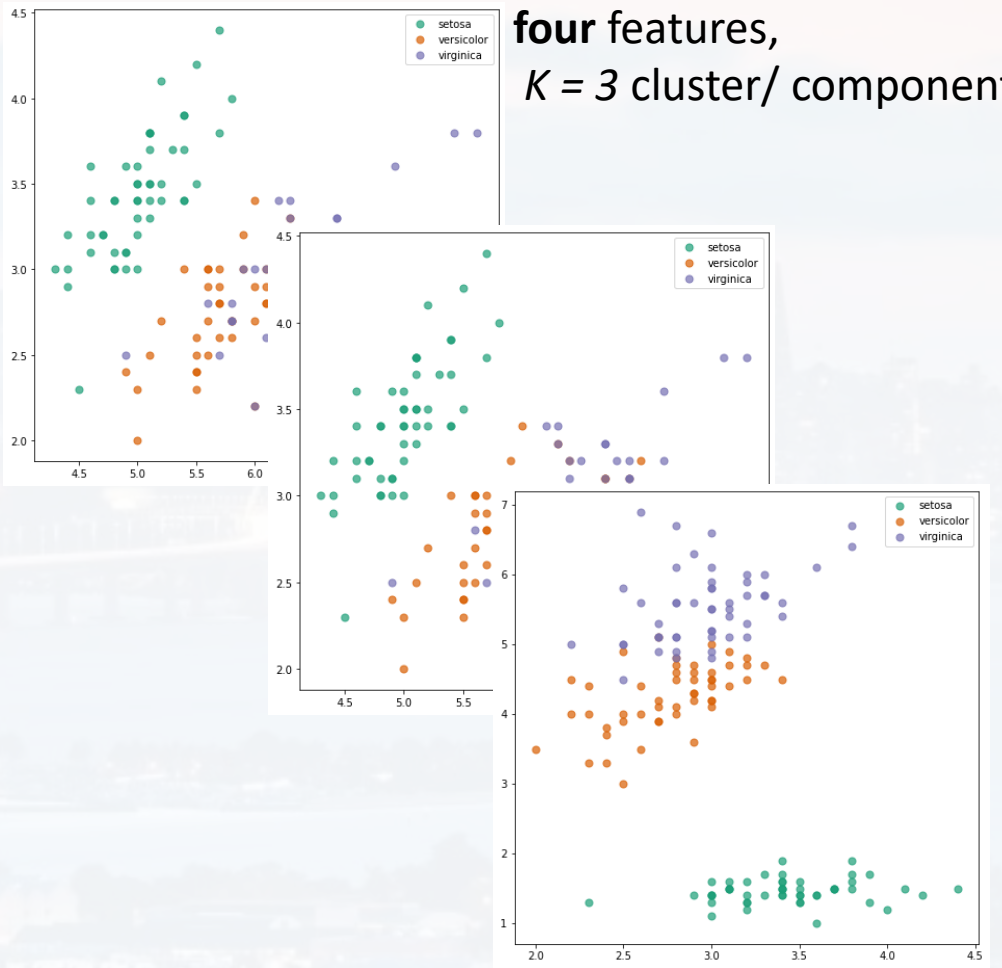
Machine Learning Algorithms:

Gaussian Mixture Models

two features, *K = 3* cluster/components



four features,
K = 3 cluster/ components

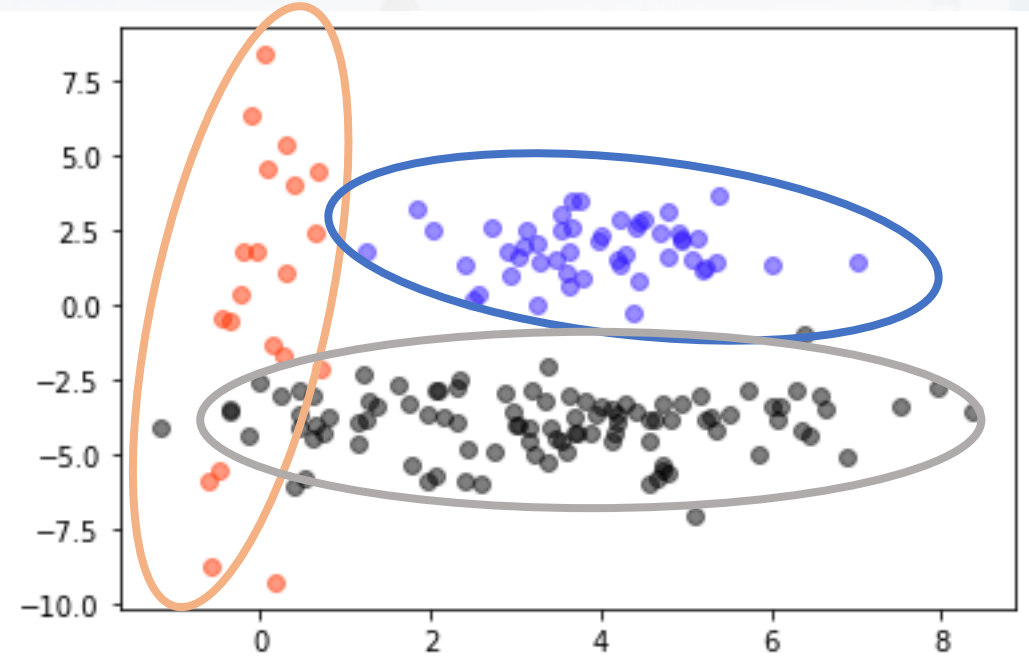


Gaussian **M**ixture **M**odels

K	: number of cluster
π_k	: mixing coefficient
C_k	: cluster k

f features

$$\mathcal{N}_k(x|\mu_k,\Sigma_k) = \frac{1}{(2\pi)^{f/2}\det(\Sigma_k)^{1/2}}\, exp\left[-\frac{1}{2}(x-\mu_k)^T\Sigma_k{}^{-1}(x-\mu_k)\right]$$



two features, $K = 3$ cluster/components

$$P(x) = \sum_{k=1}^{K} P(x|\boldsymbol{C_k})\, P(\boldsymbol{C_k})$$

$$= \sum_{k=1}^{K} \mathcal{N}_k(x|\mu_k,\Sigma_k)\,\boxed{\pi_k}$$

probability to pick a cluster C_k

Gaussian **M**ixture **M**odels

N	**: number of observations**
K	**: number of cluster**
$\boldsymbol{\pi_k}$	**: mixing coefficient**

$$P(x) = \sum_{k=1}^{K} P(x|\boldsymbol{C_k})\, P(\boldsymbol{C_k}) = \sum_{k=1}^{K} \mathcal{N}_k(x|\mu_k, \Sigma_k)\, \pi_k$$

marginalization

indicator variable $z_k \,\epsilon\, \{0, 1\}$

goal: $$P(z_k = 1|x) = \frac{P(z_k = 1)\, P(x|z_k = 1)}{P(x)} = \frac{\pi_k\, \mathcal{N}_k(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \mathcal{N}_j\left(x|\mu_j, \Sigma_j\right) \pi_j}$$

Bayes Theorem

via maximizing likelihood by finding best θ

$$L = ln[P(x|\pi, \mu, \Sigma)] = \ln\left[\prod_{n=1}^{N}\left\{\sum_{k=1}^{K} \pi_k\, \mathcal{N}_k(x_n|\mu_k, \Sigma_k)\right\}\right]$$

model parameter $\theta = \{\pi, \mu, \Sigma\}$

$$= \sum_{n=1}^{N} ln\left\{\sum_{k=1}^{K} \pi_k\, \mathcal{N}_k(x_n|\mu_k, \Sigma_k)\right\}$$

Machine Learning Algorithms:

Gaussian **M**ixture **M**odels

idea: fitting the data to a GMM → analytical functions to **calculate** probabilities for labels

<u>different algorithms:</u>

- Bayesian
- **E**xpectation **M**aximization
- …

```
my_model   = GaussianMixture(n_components = k, random_state = 0).fit(X)

Center     = my_model.means_
PredLabels = my_model.predict(X)
```

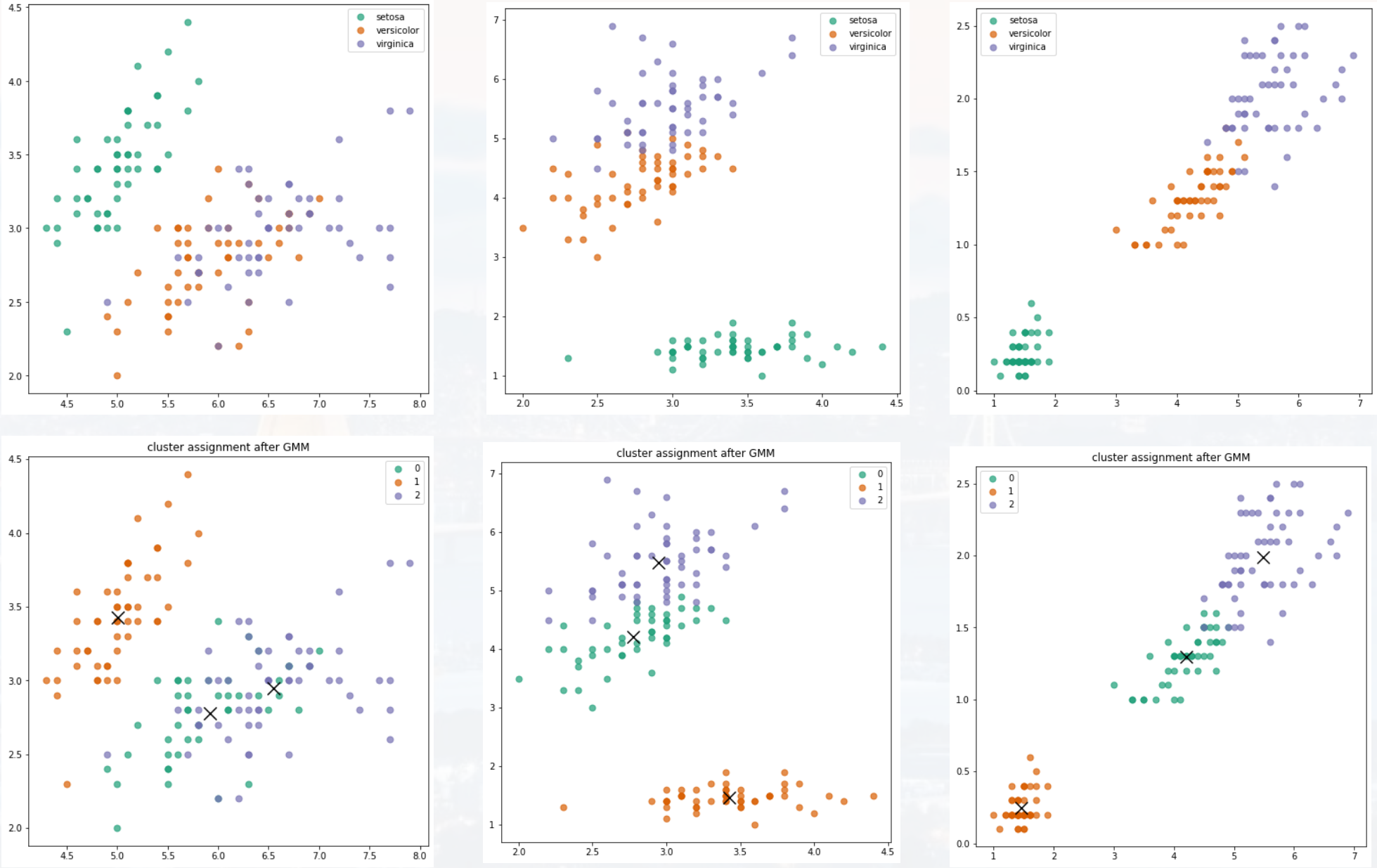
setting initial labels randomly

check out **`Walk_Through_GMM.ipynb`**

Machine Learning Algorithms: GMM

true classes

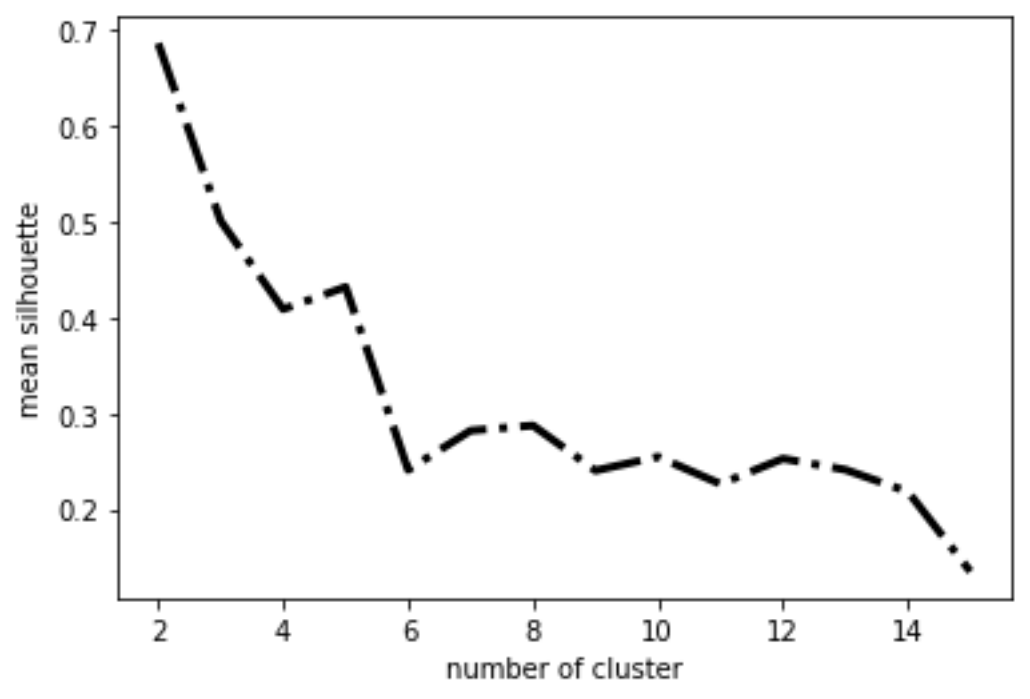
assigned classes



Machine Learning Algorithms:

GMM

we run the GMM now for the **full 4D** dataset
+ evaluate clustering with silhouette

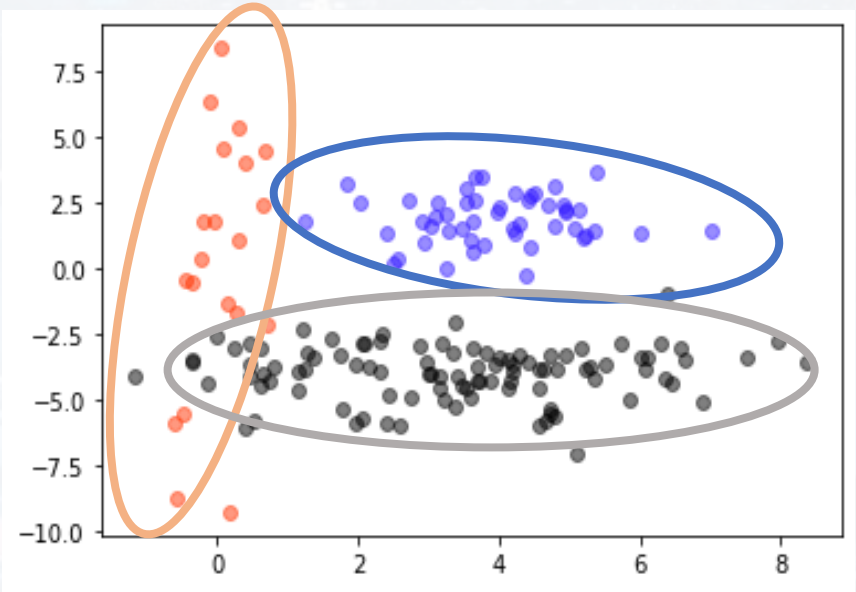


accuracy for 4D k = 3: 93%

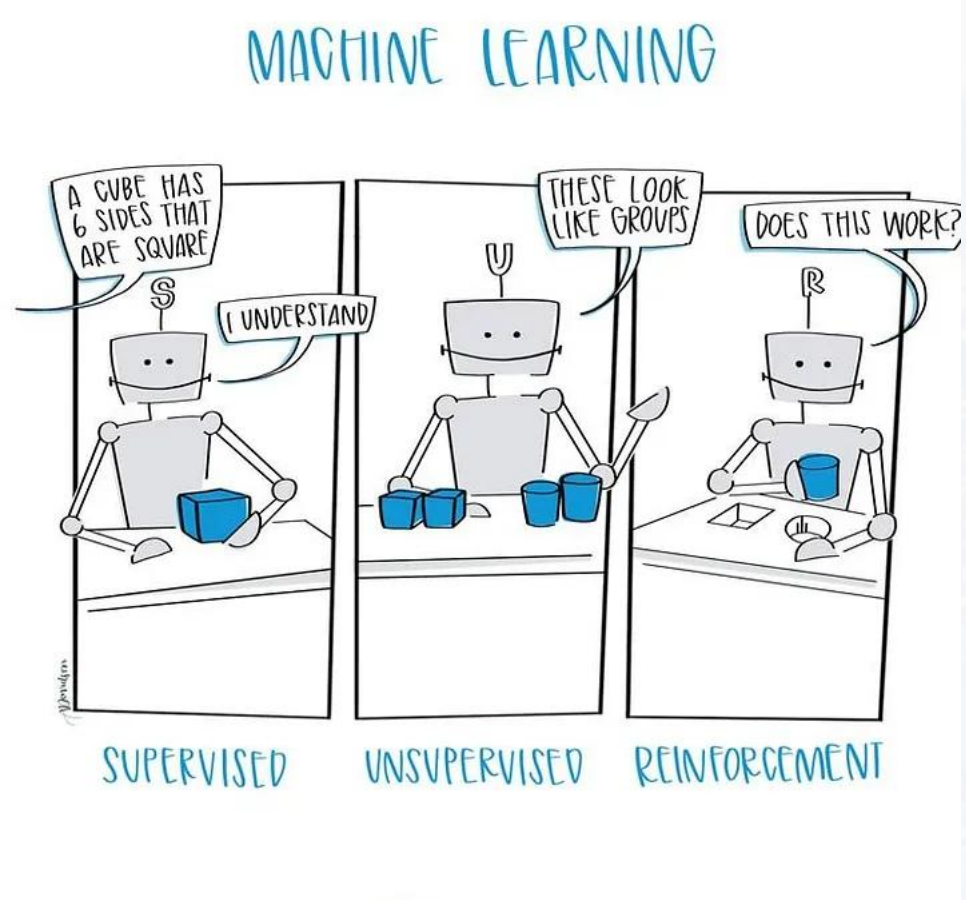
Machine Learning Algorithms:

<u>topics for the discussion/office hour:</u>

- EM algorithm
- mean, variance and covariance in more detail



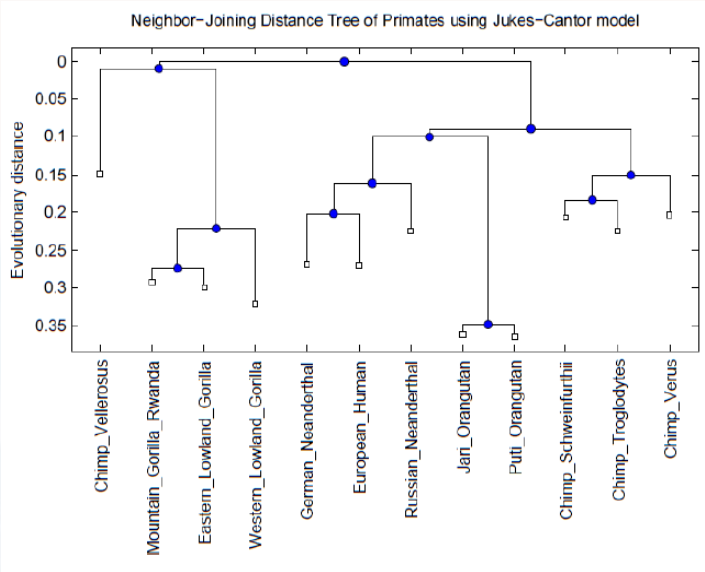
Machine Learning Algorithms:

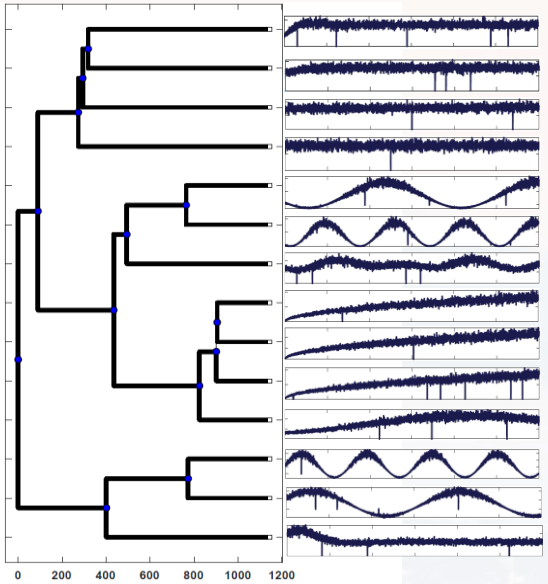


Outline

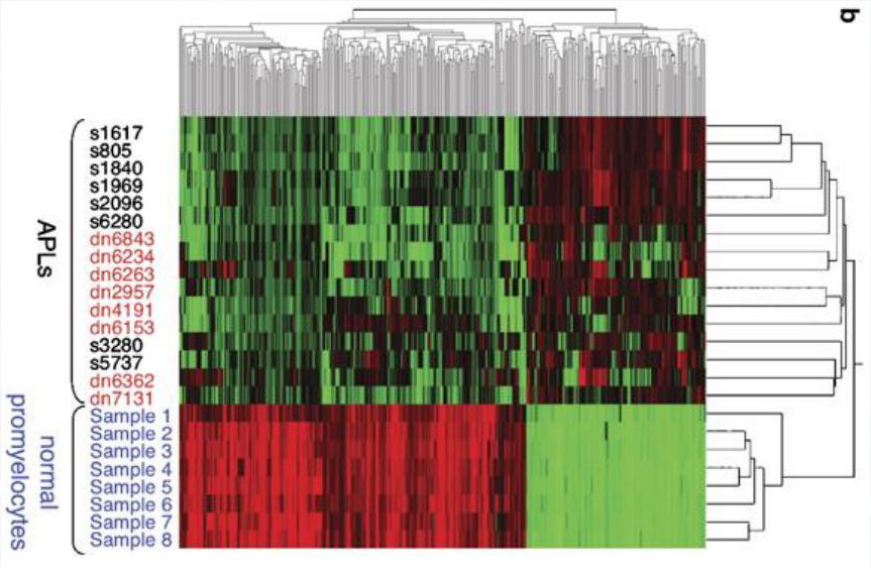
- K - means
- GMM
- **trees**

Machine Learning Algorithms: Trees



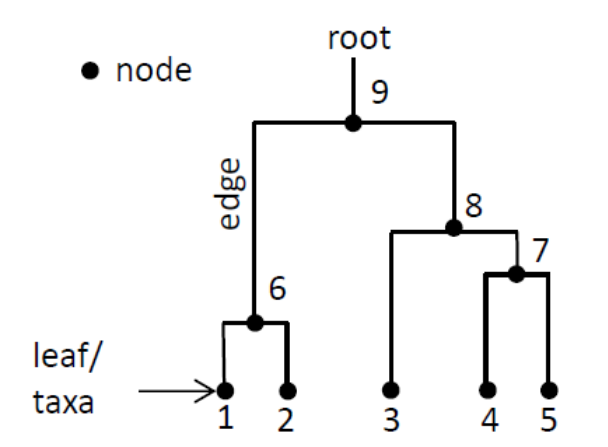


- What is a tree?
- Different kinds of trees...?
- How to build a tree?
- Why do we need trees?
- Examples...

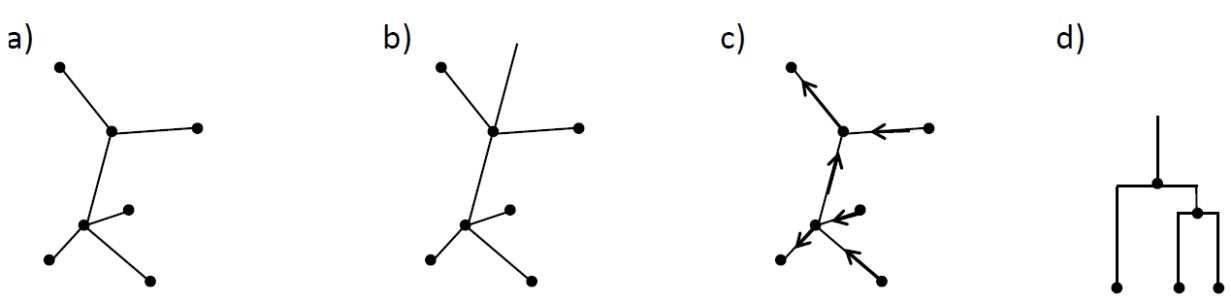


Machine Learning Algorithms: Trees

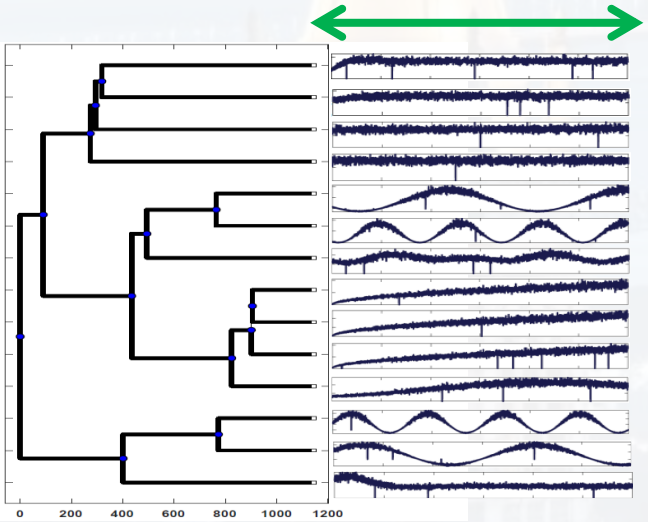
trees are a subclass of graphs **(but: not fully connected → "hierarchy", no loops)**:



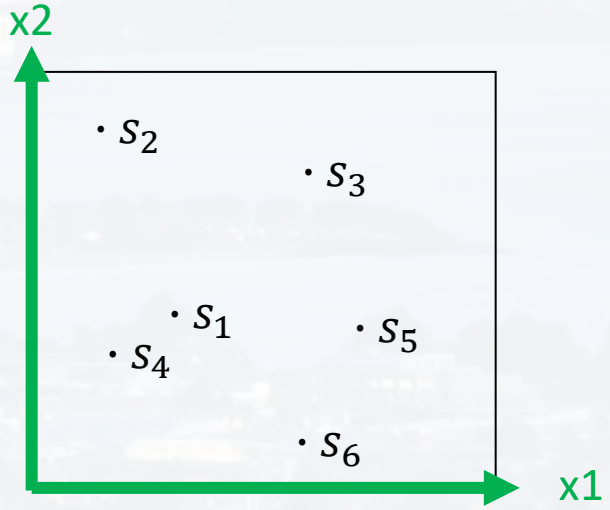
a) unrooted, undirected multinary tree
b) rooted, undirected multinary tree
c) unrooted, directed multinary tree
d) rooted, undirected binary tree



N timepoints



each sample s is a vector of N rows, hence, a data point in N-D



Machine Learning Algorithms:

constructing trees:

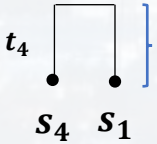
- calculating a distance between each pair of samples

	s_1	s_2	s_3	s_4	s_5	s_6
s_1	0	$d(s_1, s_2)$	$d(s_1, s_3)$	$d(s_1, s_4)$	$d(s_1, s_5)$	$d(s_1, s_6)$
s_2		0	$d(s_2, s_3)$	$d(s_2, s_4)$	$d(s_2, s_5)$	$d(s_2, s_6)$
s_3			0	$d(s_3, s_4)$	$d(s_3, s_5)$	$d(s_3, s_6)$
s_4				0	$d(s_4, s_5)$	$d(s_4, s_6)$
s_5					0	$d(s_5, s_6)$
s_6						0

Question: What could be a proper distance definition here?

...once a distance has been defined...

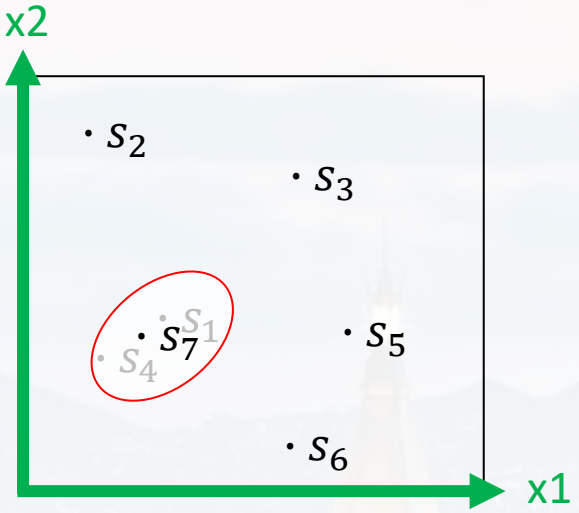
→ find the closest pair



$$t_1 = t_4 = \frac{1}{2} d(s_1, s_4)$$

Machine Learning Algorithms:

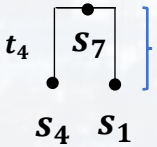
constructing trees:



- calculating a distance between each pair of samples

	s_1	s_2	s_3	s_4	s_5	s_6
s_1	0	$d(s_1, s_2)$	$d(s_1, s_3)$	$d(s_1, s_4)$	$d(s_1, s_5)$	$d(s_1, s_6)$
s_2		0	$d(s_2, s_3)$	$d(s_2, s_4)$	$d(s_2, s_5)$	$d(s_2, s_6)$
s_3			0	$d(s_3, s_4)$	$d(s_3, s_5)$	$d(s_3, s_6)$
s_4				0	$d(s_4, s_5)$	$d(s_4, s_6)$
s_5					0	$d(s_5, s_6)$
s_6						0

→ treat it as a new cluster $s_{1,4}$
→ use average of distance from cluster elements

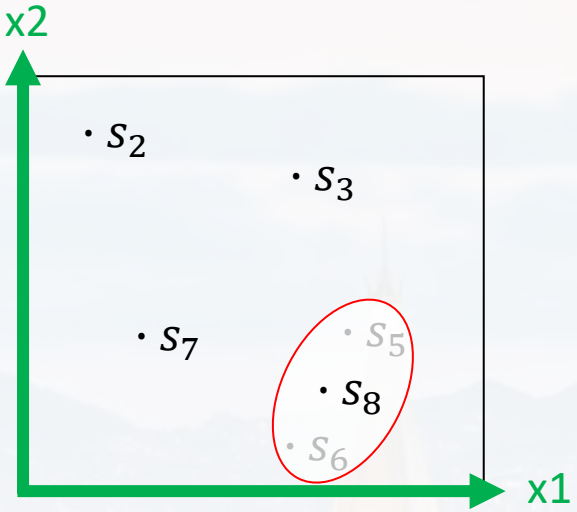


$t_1 = t_4 = \frac{1}{2} d(s_1, s_4)$

Machine Learning Algorithms:

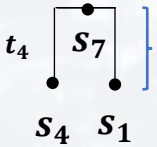
<u>constructing trees:</u>

- calculating a distance between each pair of samples

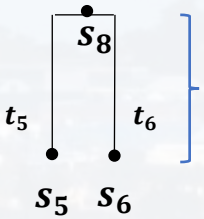


	s_1	s_2	s_3	s_4	s_5	s_6
s_1	0	$d(s_1, s_2)$	$d(s_1, s_3)$	$d(s_1, s_4)$	$d(s_1, s_5)$	$d(s_1, s_6)$
s_2		0	$d(s_2, s_3)$	$d(s_2, s_4)$	$d(s_2, s_5)$	$d(s_2, s_6)$
s_3			0	$d(s_3, s_4)$	$d(s_3, s_5)$	$d(s_3, s_6)$
s_4				0	$d(s_4, s_5)$	$d(s_4, s_6)$
s_5					0	$d(s_5, s_6)$
s_6						0

→ find the closest pair
(now including the cluster)



$t_1 = t_4 = \frac{1}{2} d(s_1, s_4)$

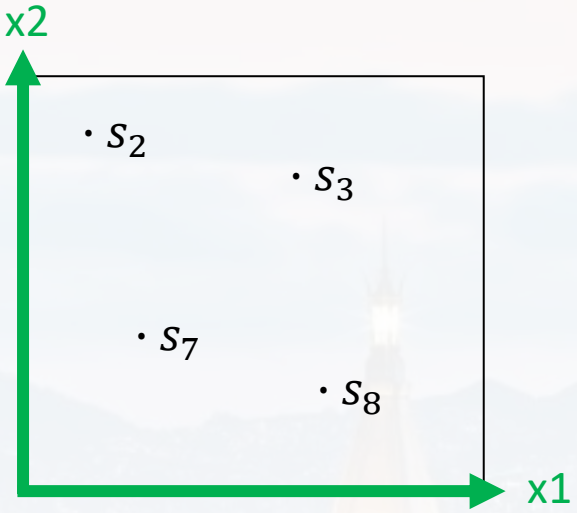


$t_6 = t_5 = \frac{1}{2} d(s_5, s_6)$

Machine Learning Algorithms:

<u>constructing trees:</u>

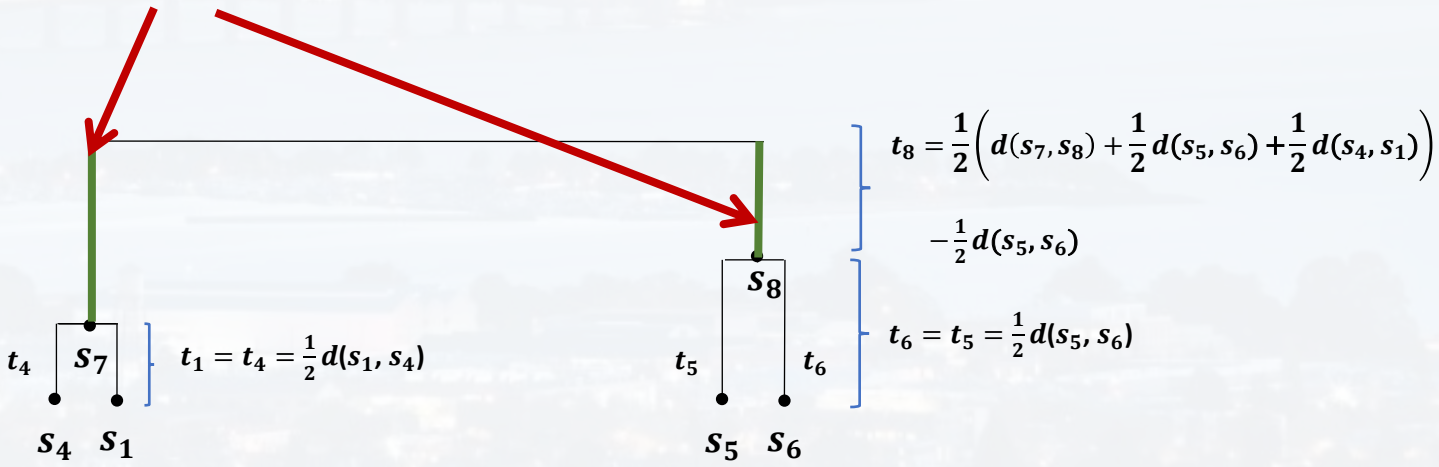
- calculating a distance between each pair of samples



	s_1	s_2	s_3	s_4	s_5	s_6
s_1	0	$d(s_1, s_2)$	$d(s_1, s_3)$	$d(s_1, s_4)$	$d(s_1, s_5)$	$d(s_1, s_6)$
s_2		0	$d(s_2, s_3)$	$d(s_2, s_4)$	$d(s_2, s_5)$	$d(s_2, s_6)$
s_3			0	$d(s_3, s_4)$	$d(s_3, s_5)$	$d(s_3, s_6)$
s_4				0	$d(s_4, s_5)$	$d(s_4, s_6)$
s_5					0	$d(s_5, s_6)$
s_6						0

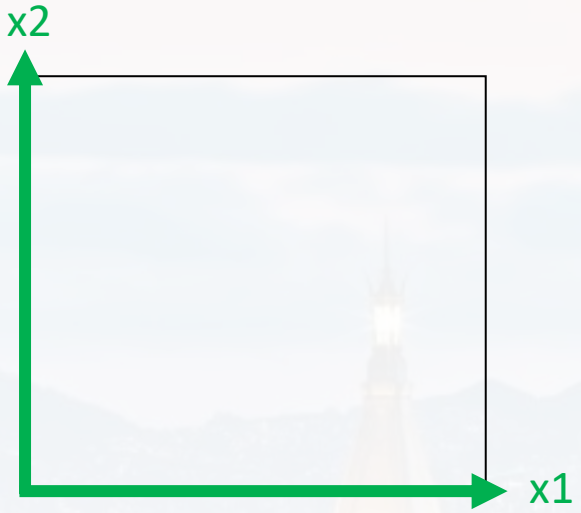
$$d(s_7, s_8) = \frac{d(s_5, s_7) + d(s_6, s_7)}{2} = \frac{d(s_1, s_5) + d(s_4, s_5) + d(s_1, s_6) + d(s_4, s_6)}{4}$$

→ and so on....



Machine Learning Algorithms:

constructing trees:

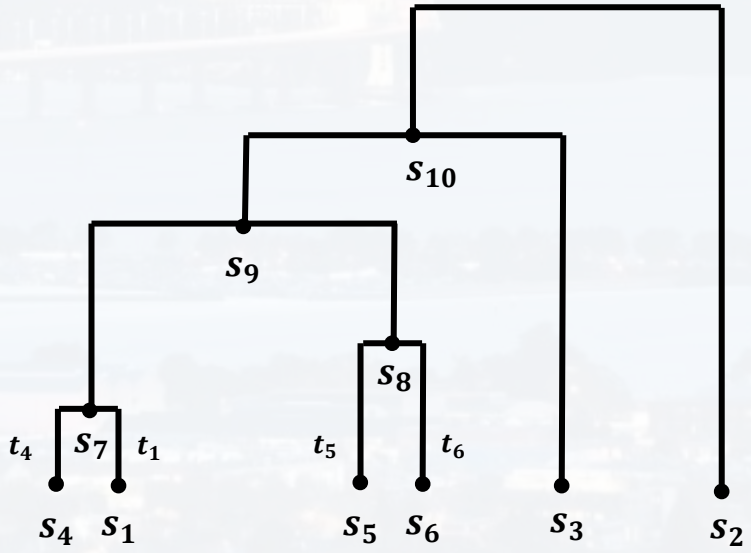


- calculating a distance between each pair of samples

	s_1	s_2	s_3	s_4	s_5	s_6
s_1	0	$d(s_1, s_2)$	$d(s_1, s_3)$	$d(s_1, s_4)$	$d(s_1, s_5)$	$d(s_1, s_6)$
s_2		0	$d(s_2, s_3)$	$d(s_2, s_4)$	$d(s_2, s_5)$	$d(s_2, s_6)$
s_3			0	$d(s_3, s_4)$	$d(s_3, s_5)$	$d(s_3, s_6)$
s_4				0	$d(s_4, s_5)$	$d(s_4, s_6)$
s_5					0	$d(s_5, s_6)$
s_6						0

....finally

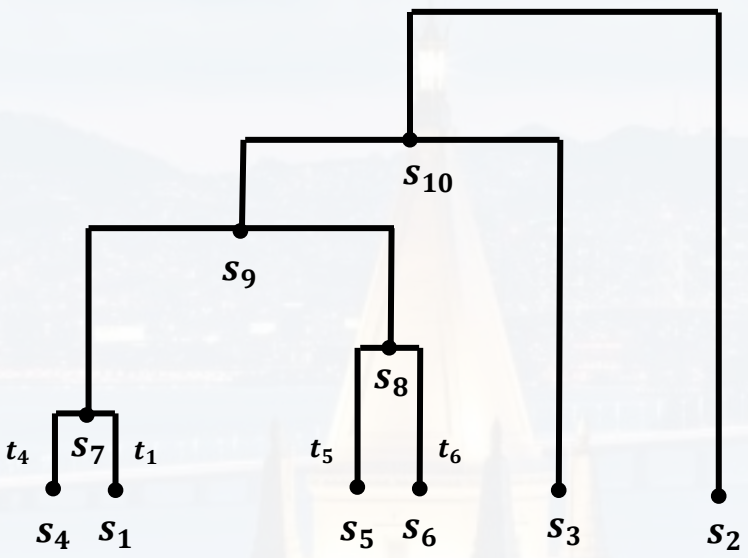
→ **U**nweighted **P**air **G**roup **M**ethod Using **A**rithmetic **A**verages (UPGMA)

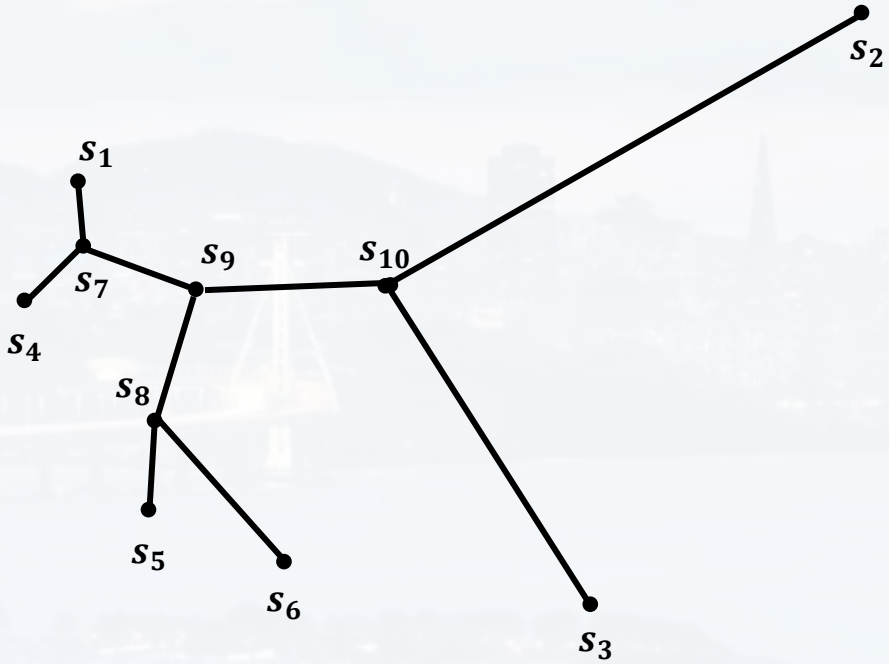


Machine Learning Algorithms:

constructing trees:

→ **U**nweighted **P**air **G**roup **M**ethod Using **A**rithmetic **A**verages (UPGMA)





note: sometimes these diagrams might be misleading when interpreting the distances between the nodes

Machine Learning Algorithms: Trees

distances:

"A distance ***d*** is a function that assigns a **positive real number** to a set and/or to elements of a set."

properties:

- **d(x , y) ≥ 0**
- **d(x , y) = 0 ⇔ $x = y$**
- **d(x , y) = d(y , x)**
- **d(x , z) ≤ d(x , y) + d(y , z)**

frequently used distances:

- Euclidean
- cityblock
- cosine
- correlation
- hamming
- Jukes Cantor

Machine Learning Algorithms:

distances:

"A distance ***d*** is a function that assigns a **positive real number** to a set and/or to elements of a set."

properties:

- **d(x , y) ≥ 0**
- **d(x , y) = 0** $\Leftrightarrow x = y$
- **d(x , y) = d(y , x)**
- **d(x , z) ≤ d(x , y) + d(y , z)**

- Euclidean
- cityblock
- cosine
- correlation
- hamming
- Jukes Cantor

```
from pyclustering.utils.metric import *
```

```
type_metric.
```

CANBERRA
CHEBYSHEV
CHI_SQUARE
EUCLIDEAN
EUCLIDEAN_SQUARE
GOWER
MANHATTAN

Machine Learning Algorithms: Trees

<u>distances:</u>

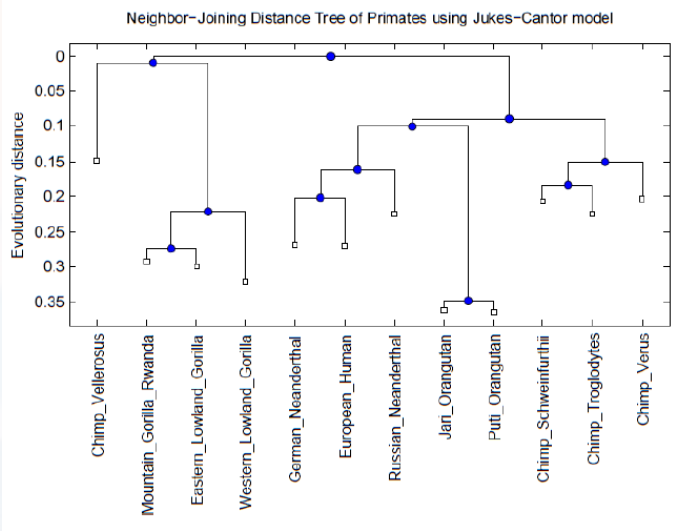
The Jukes-Cantor distance

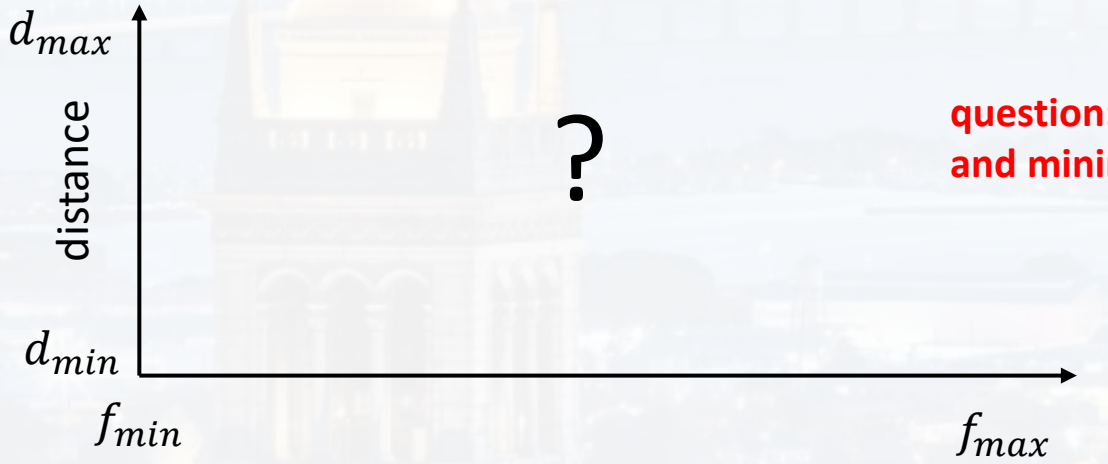
question: What could be a measure of distance here?

Seq1 = {AAGTCTTTGAATCGTATGGGCGCGGCTTTCAAAA}
Seq2 = {AGGTCTATGAATCGTATGGGTTCGGCTTTCAAAA}

fraction ***f*** of different sites

if ***f*** is large → ***d*** is large
if ***f*** is small → ***d*** is small



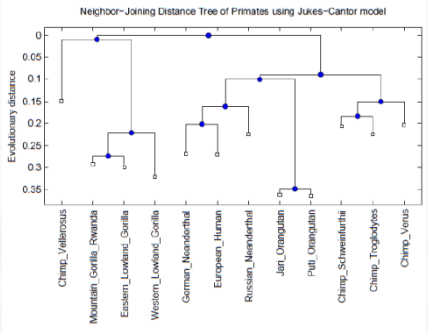


question: What are the maximum and minimum here?

Machine Learning Algorithms: Trees

distances:

f : fraction of different sites



The Jukes-Cantor distance

Seq1 = {AAGTCTTTGAATCGTATGGGCGCGGCTTTCAAAA}
Seq2 = {AGGTCTATGAATCGTATGGGTTCGGCTTTCAAAA}

$f_{min} = 0 \rightarrow d_{min} = 0$ (sequences are identical)

for ***N*** letters (***N = 4*** nucleotides, ***N = 20*** amino acids)

$f_{max} = 1 - \frac{1}{N} \rightarrow d_{\max} = \infty$

(sequences have no relations, matches are coincidence)

- the probability that a site **has been** mutated, given a mutation rate λ equals:

$$P(any|\lambda) = 1 - e^{-\lambda t}$$

- the probability that this mutation will lead to **a particular letter** is

$$P(part|\lambda) = \frac{1}{N}(1 - e^{-\lambda t})$$

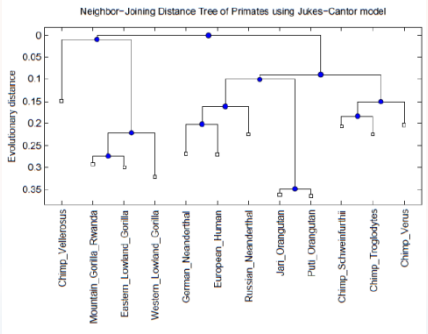
- the probability that this mutation will lead to a **letter other than the previous one** is

$$P(part|\lambda) = \frac{N-1}{N}(1 - e^{-\lambda t})$$

Machine Learning Algorithms: Trees

distances:

f : fraction of different sites



The Jukes-Cantor distance

Seq1 = {AAGTCTTTGAATCGTATGGGCGCGGCTTTCAAAA}
Seq2 = {AGGTCTATGAATCGTATGGGTTCGGCTTTCAAAA}

$f_{min} = 0 \rightarrow d_{min} = 0$ (sequences are identical)

for ***N*** letters (***N = 4*** nucleotides, ***N = 20*** amino acids)

- the probability that this mutation will lead to a **letter other than the previous one** is

$$P(part|\lambda) = \frac{N-1}{N}(1 - e^{-\lambda t})$$

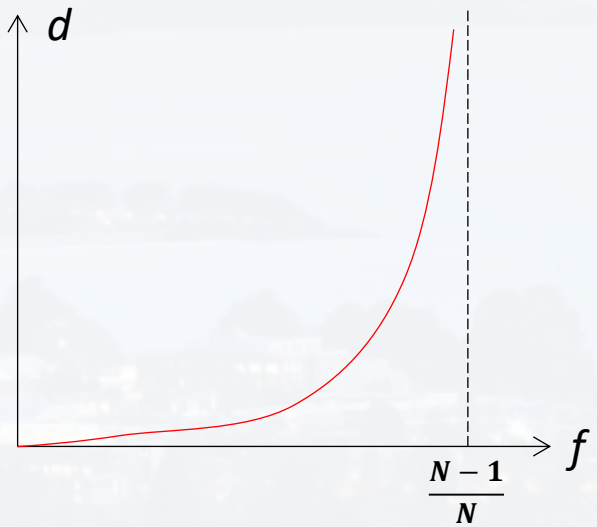
$\lambda t \sim d$ since cumulative mutations is what makes species different!

$$d \sim \lambda t = -ln\left(1 - \frac{N}{N-1} f\right)$$

often:

$$\boxed{d = -\frac{N-1}{N} ln\left(1 - \frac{N}{N-1} f\right)}$$

Jukes – Cantor distance



Machine Learning Algorithms:

distances:

Jukes – Cantor distance $$d(x,\bar{x}) = -\frac{N-1}{N} ln\left(1 - \frac{N}{N-1} f(x,\bar{x})\right)$$

Euclidean distance $$d(x,\bar{x})^2 = (x_1 - \bar{x}_1)^2 + (x_2 - \bar{x}_2)^2 \dots (x_N - \bar{x}_N)^2$$

cityblock $$d(x,\bar{x}) = |x_1 - \bar{x}_1| + |x_2 - \bar{x}_2| \dots |x_N - \bar{x}_N|$$

cosine $$d(x,\bar{x}) = 1 - \frac{x \cdot \bar{x}}{\sqrt{(x \cdot x)(\bar{x} \cdot \bar{x})}}$$

correlation $$d(x,\bar{x}) = 1 - \frac{cov(x,\bar{x})}{\sqrt{var(x)var(\bar{x})}}$$

...and many others

note: for sequence alignment and phylogenetic trees, chemical properties of the AA or NT have to be taken into account → **score matrices**

Machine Learning Algorithms: Trees

Python libraries:

libraries from the *Bio* package

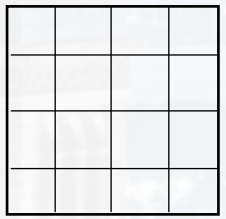
```
from Bio.Phylo.TreeConstruction import DistanceCalculator, DistanceTreeConstructor
from Bio import Phylo
from scipy import spatial
```

for plotting a phylogenetic tree

for rearranging a distance matrix

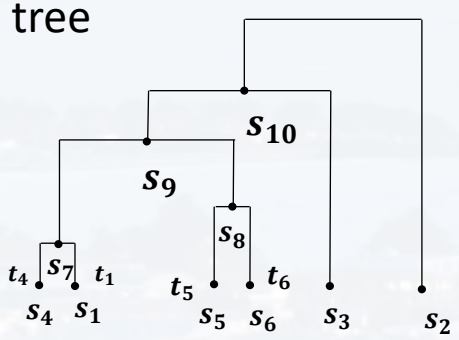
general idea:

distance matrix



linkage algorithm (e.g. UPGMA)

tree



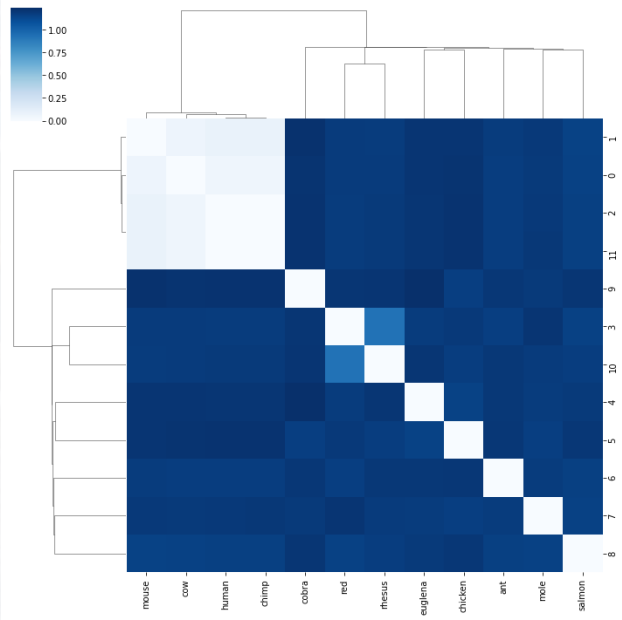
Machine Learning Algorithms: Trees

Python libraries:

also most heatmap tools have some abilities to construct trees: `sns.clustermap`

```
sns.clustermap(D_df, cmap = 'Blues', \
                     row_cluster = True, col_cluster = True, \
                     metric = 'euclidean', method = 'average', \
                     yticklabels = True)
plt.show()
```

similar to UPGMA



Machine Learning Algorithms: Trees

topics for the discussion/office hour:

- distances
- random forest
- graphs

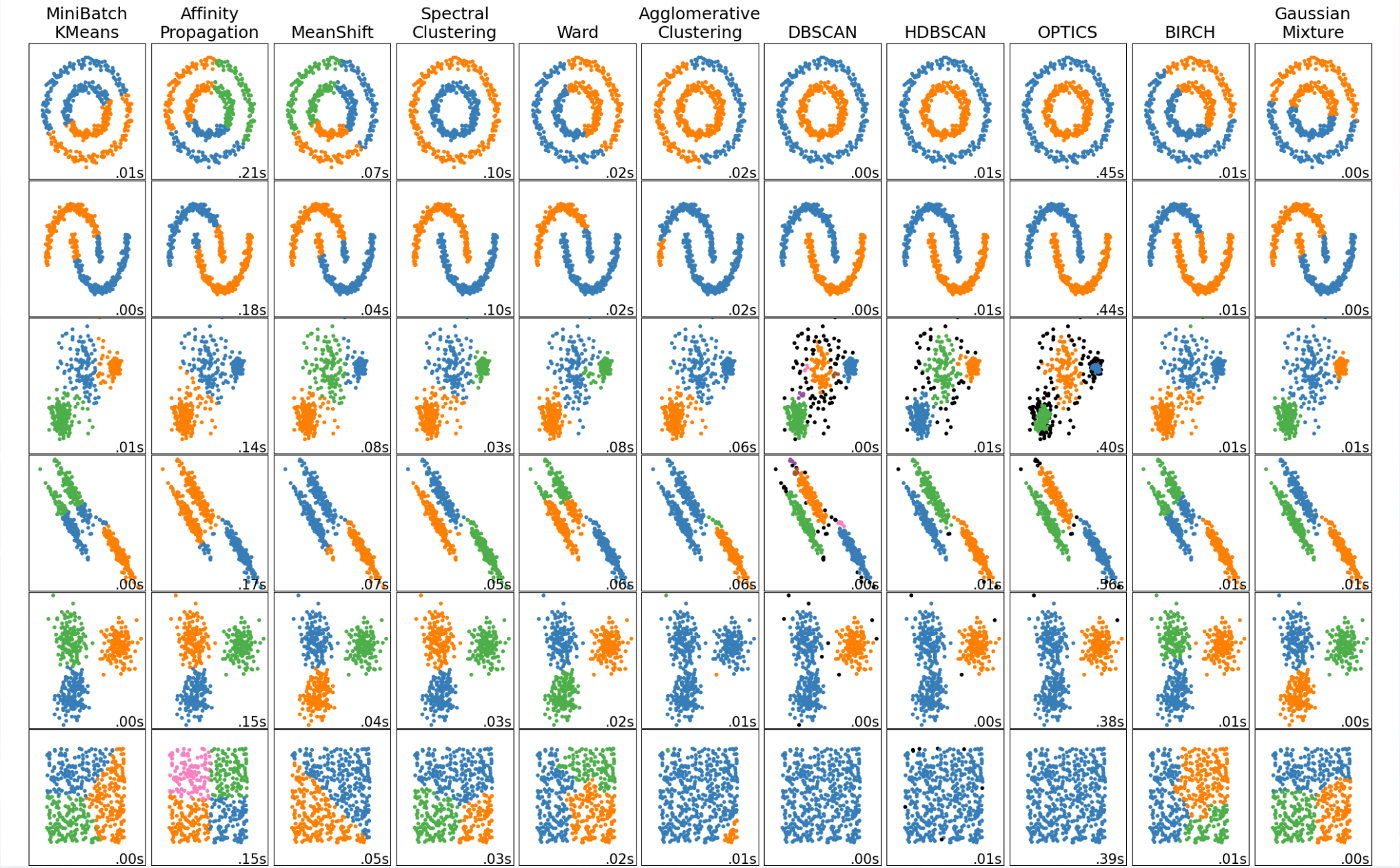
see also

`Walk_Through_Tree.ipynb`

for more details

Machine Learning Algorithms: Unsupervised Learning

there is a lot more…



Thank you very much for your attention!