Unsupervised Learning: Clustering validity

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ

Contents

- The three fundamental questions that need to be addressed in any typical clustering scenario are:
  1. **how many clusters** are present, if any
  2. which **clustering technique** is suitable for the given data set, and
  3. **how real or good** is the clustering itself.
- The tasks of determining the number of clusters [1.] and also the **validity of the clusters** formed [3.] are generally addressed by means of the so-called **validity indices**
  – They can also be useful for **comparing** the output of different clustering algorithms [2.]
- There are validity indices for specific algorithms, e.g. *fuzzy partition coefficient*
- Validity indices can be classified as:
  – **internal**: they assess only clusters plausibility, most of then quantify how good a particular partitioning is in terms of
    – **compactness**, considered as the overall proximity among the cluster elements, and
    – **separation** between clusters
  – **external**: they assume the availability of class labels (≡ ground truth)

- In the following, we will overview some clustering validation approaches:
  - **clusterability measures**:
    - Scatter Plot Matrix (SPLOM) and the Parallel Coordinates Plot
    - Hopkins statistic
    - Visual Assessment of [clustering] Tendency (VAT)
  - **visual tools**: Elbow method and the Silhouette coefficient
  - **internal indices**: Dunn index and Davies-Bouldin index
  - **external indices**: Homogeneity, Completeness and V-measure

clusterability

adequacy of the clustering

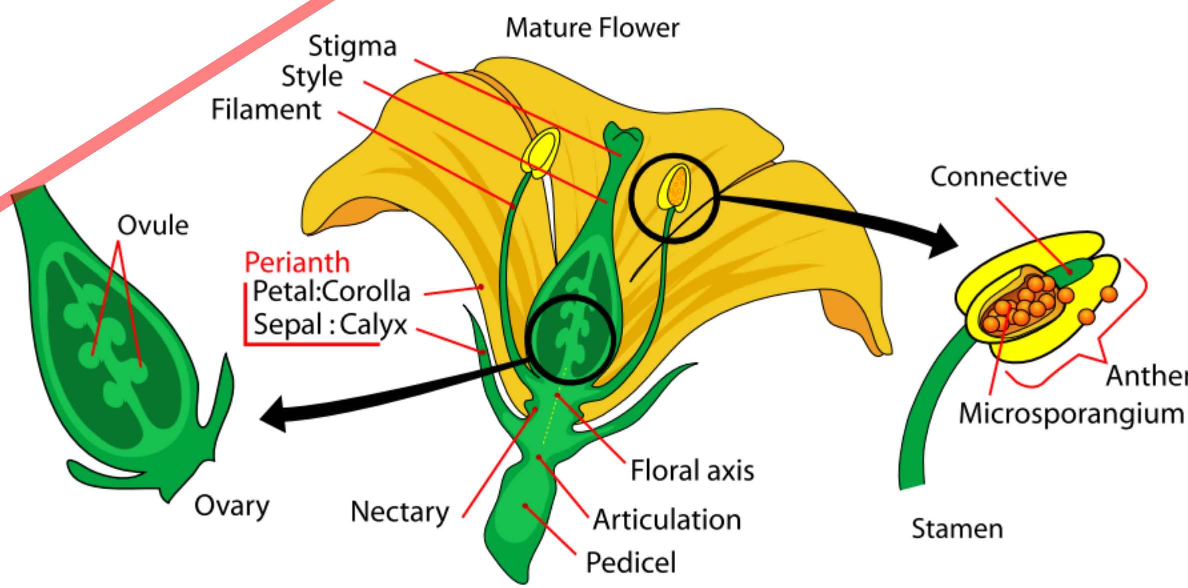
among many others:

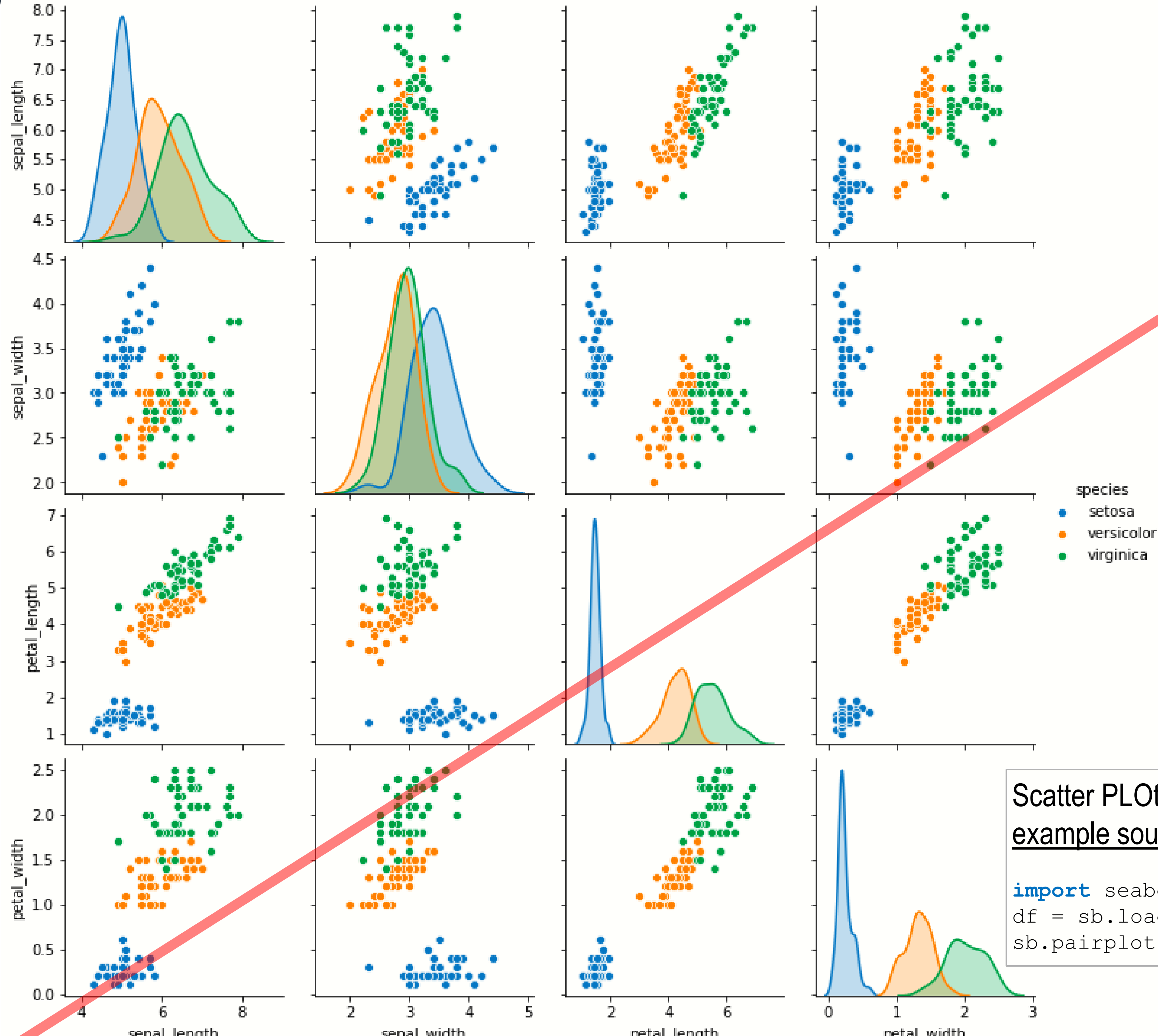
- Calinski-Harabasz Index - internal
- Fowlkes-Mallows score - external
- Rand Index and Adjusted Rand Index (ARI) - external
- Mutual Information, Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI) – external
- etc.

Contents

Is there structure in the data?

- Before attempting any clustering task on the data, we should test whether the data is structured in clusters
- Among many others, the **Scatter Plot Matrix (SPLOM)** and the **parallel coordinates plot** are standard visualization tools, though of limited capability
  - e.g. for the Iris flower data set (Fisher's Iris data set)
  - multivariate data set by the British statistician and biologist Ronald Fisher (1936)
  - 150 samples under four attributes:
    - sepal length
    - sepal width
    - petal length
    - petal width
  - 3 species:
    - setosa
    - versicolor
    - virginica

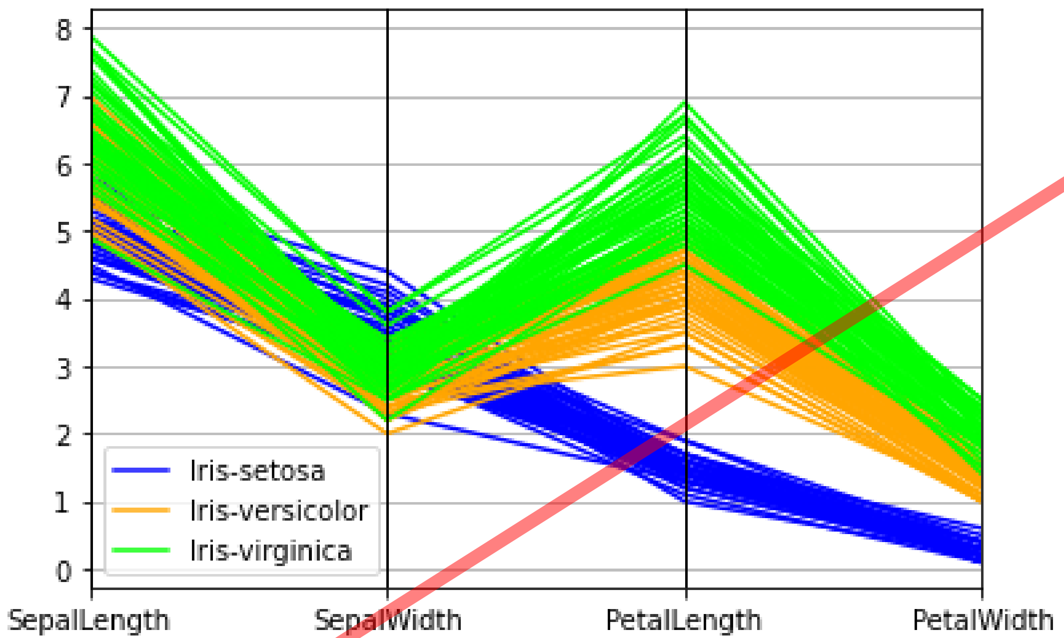




Scatter PLOt Matrix
<u>example source code</u>:

```
import seaborn as sb
df = sb.load_dataset('iris')
sb.pairplot(df, hue='species')
```

Is there structure in the data?



<u>example source code</u> (of parallel coordinates plot):

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sb
df = sb.load_dataset('iris')
pd.plotting.parallel_coordinates(df, 'species', color=('#0000FF', '#FFA500', '#00FF00'))
plt.legend(loc='lower left')
plt.show()
```

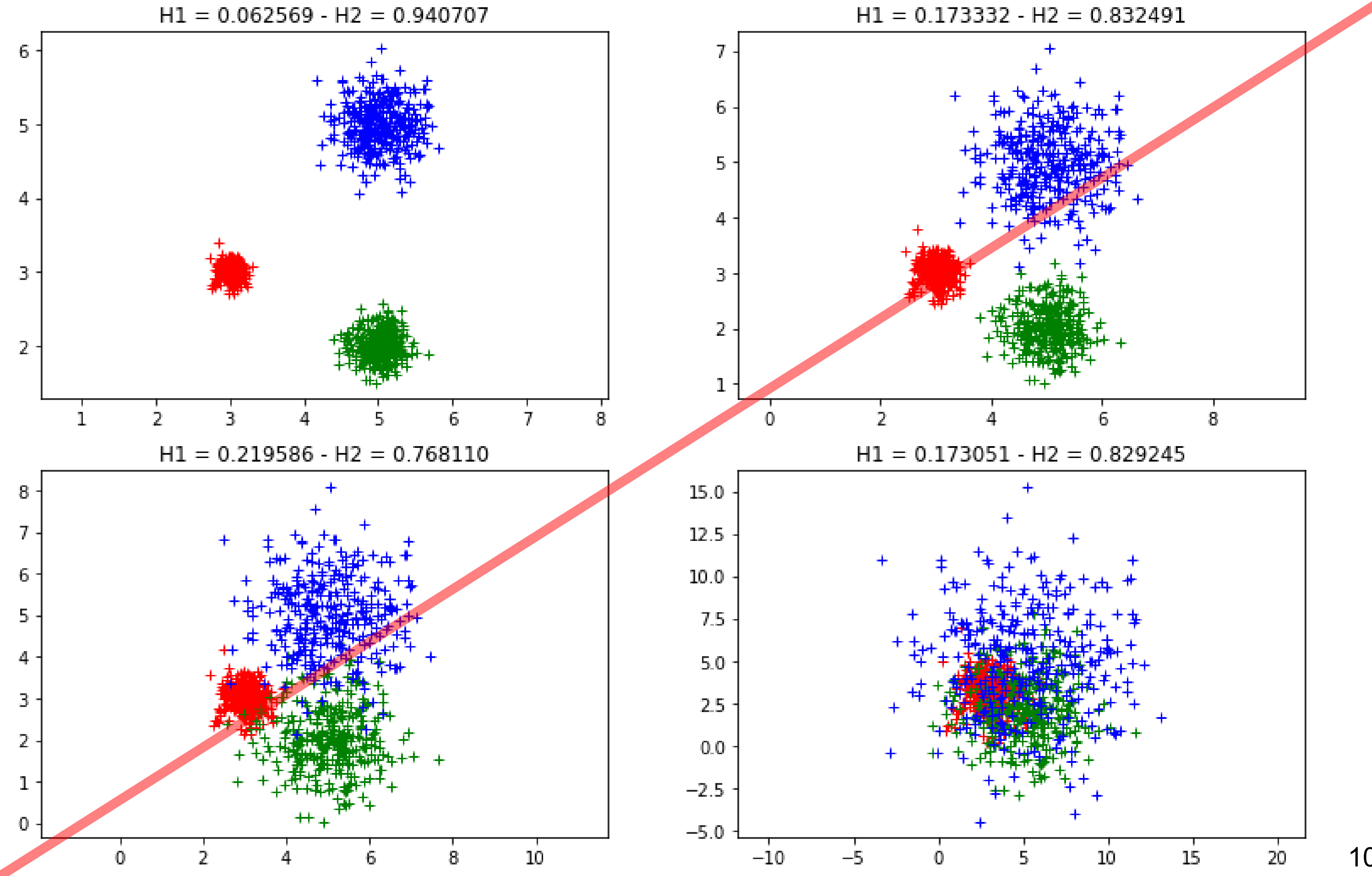
Is there structure in the data?

- We can also test the hypothesis of the existence of groups versus a dataset consisting of samples uniformly distributed – **Hopkins statistic**:
  1. Get n samples p_i from the dataset D and compute the distance to the nearest neighbor $d(p_i)$
  2. Generate n points q_i uniformly distributed in the feature space and compute their distance $d(q_i)$ to the nearest neighbor in D
  3. Compute any of the two following quotients:

$$H_1 = \frac{\sum_{i=1}^{n} d(p_i)}{\sum_{i=1}^{n} d(p_i) + \sum_{i=1}^{n} d(q_i)} \qquad H_2 = \frac{\sum_{i=1}^{n} d(q_i)}{\sum_{i=1}^{n} d(p_i) + \sum_{i=1}^{n} d(q_i)}$$

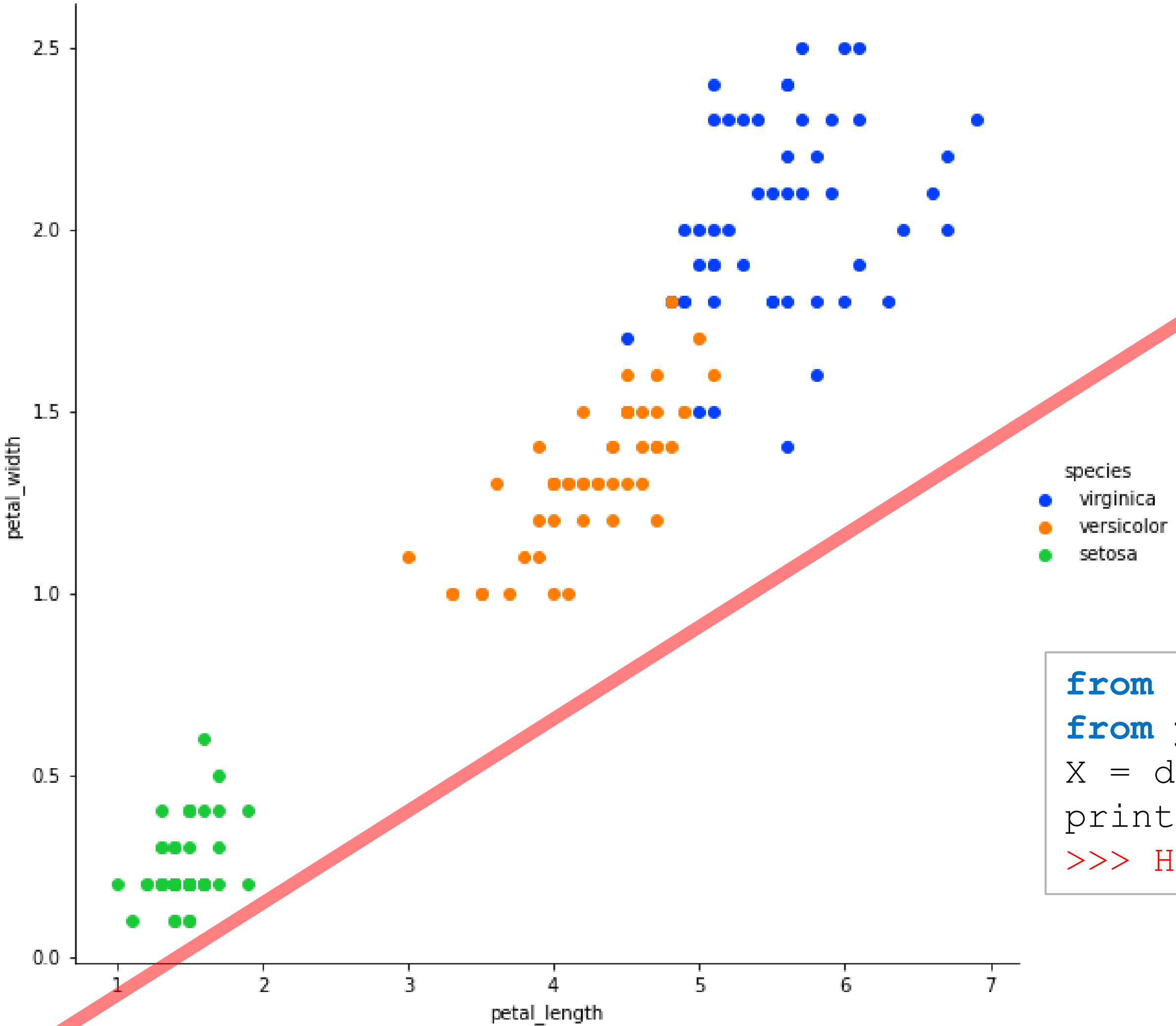
  4. If data are uniformly distributed (= no structure) the values of H_1 and H_2 get around 0.5. Otherwise:
     - H_1 takes values close to 0 for *clusterable* datasets
     - H_2 takes values close to 1 for *clusterable* datasets

Is there structure in the data?



Is there structure in the data?

- Example source code (Iris dataset):




```
from sklearn import datasets
from pyclustertend import hopkins
X = datasets.load_iris().data
print('H1 = ', hopkins(X,150))
>>> H1 = 0.1764
```

Is there structure in the data?

- **VAT** (Visual Assessment of [clustering] Tendency) follows a visual approach based on re-ordering the proximity matrix, e.g. using a dissimilarity

	x1	x2	x3	x4	x5
x1	0	0.73	0.19	0.71	0.16
x2	0.73	0	0.59	0.12	0.78
x3	0.19	0.59	0	0.55	0.19
x4	0.71	0.12	0.55	0	0.74
x5	0.16	0.78	0.19	0.74	0

⇨

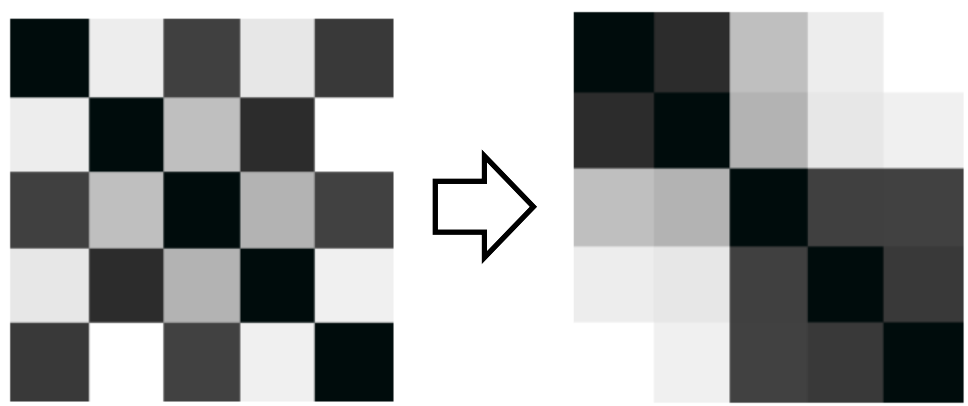
	x2	x4	x3	x1	x5
x2	0	0.12	0.59	0.73	0.78
x4	0.12	0	0.55	0.71	0.74
x3	0.59	0.55	0	0.19	0.19
x1	0.73	0.71	0.19	0	0.16
X5	0.78	0.74	0.19	0.16	0

  – By reordering the elements of this matrix we get a reordered proximity matrix which tries to accumulate smaller dissimilarity values around the diagonal of the matrix in square contiguous regions

    black = min. distance

    white = max. distance

    ⇒ 2 clusters



- **VAT** (Visual Assessment of [clustering] Tendency)

1. $K = \{1, 2, \ldots, N\}, I \leftarrow \emptyset, J \leftarrow \emptyset, O = [0, \ldots, 0]$
2. $(i, j) = \arg\max_{p \in K, q \in K} \{\wp_{pq}\}$

   $I \leftarrow \{i\}, J \leftarrow K - \{i\}, O[1] = i$
3. $\underline{\textbf{for}}\ r = 2, \ldots, N$

   $$(i, j) = \arg\min_{p \in I, q \in J} \{\wp_{pq}\}$$

   $$I \leftarrow I \cup \{j\}, J \leftarrow J - \{j\}, O[r] = j$$

   $\underline{\textbf{end}}$
4. Reorder the proximity matrix $\mathcal{P}$ using the reordering array O as:

$$\widetilde{\wp}_{ij} = \wp_{O[i]O[j]}, \quad \forall i, j$$

- **VAT** (Visual Assessment of [clustering] Tendency)
  - Example:

1) I = x2, J = {x1, x3, x4, x5}
2) I = {x2, x4}, J = {x1, x3, x5}
3) I = {x2, x4, x3}, J = {x1, x5}
4) I = {x2, x4, x3, x1} J = {x5}
5) I = {x2, x4, x3, x1, x5}

$\Rightarrow$ O = [2, 4, 3, 1, 5]

1)	**x1**	**x2**	**x3**	**x4**	**x5**
x1	0	0.73	0.19	0.71	0.16
x2	0.73	0	0.59	0.12	0.78
x3	0.19	0.59	0	0.55	0.19
x4	0.71	0.12	0.55	0	0.74
x5	0.16	0.78	0.19	0.74	0

3)	**x1**	**x2**	**x3**	**x4**	**x5**
x1	0	0.73	0.19	0.71	0.16
x2	0.73	0	0.59	0.12	0.78
x3	0.19	0.59	0	0.55	0.19
x4	0.71	0.12	0.55	0	0.74
x5	0.16	0.78	0.19	0.74	0

2)	**x1**	**x2**	**x3**	**x4**	**x5**
x1	0	0.73	0.19	0.71	0.16
x2	0.73	0	0.59	0.12	0.78
x3	0.19	0.59	0	0.55	0.19
x4	0.71	0.12	0.55	0	0.74
x5	0.16	0.78	0.19	0.74	0

4)	**x1**	**x2**	**x3**	**x4**	**x5**
x1	0	0.73	0.19	0.71	0.16
x2	0.73	0	0.59	0.12	0.78
x3	0.19	0.59	0	0.55	0.19
x4	0.71	0.12	0.55	0	0.74
x5	0.16	0.78	0.19	0.74	0

Is there structure in the data?

- **VAT** (Visual Assessment of [clustering] Tendency)

	x1	x2	x3	x4	x5
x1	0	0.73	0.19	0.71	0.16
x2	0.73	0	0.59	0.12	0.78
x3	0.19	0.59	0	0.55	0.19
x4	0.71	0.12	0.55	0	0.74
x5	0.16	0.78	0.19	0.74	0

	x2	x4	x3	x1	x5
x2	0	0.12	0.59	0.73	0.78
x4		0	0.55	0.71	0.74
x3			0	0.19	0.19
x1				0	0.16
X5					0

1) I = x2, J = {x1, x3, x4, x5}
2) I = {x2, x4}, J = {x1, x3, x5}
3) I = {x2, x4, x3}, J = {x1, x5}
4) I = {x2, x4, x3, x1} J = {x5}
5) I = {x2, x4, x3, x1, x5}

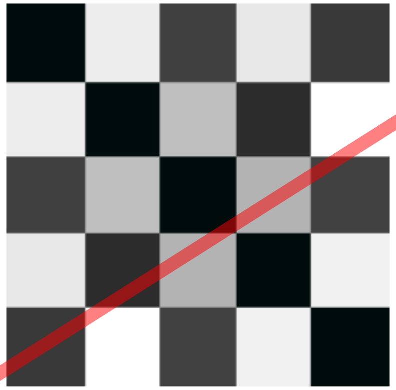
$\Rightarrow$ O = [2, 4, 3, 1, 5]

	x2	x4	x3	x1	x5
x2	0	0.12	0.59	0.73	0.78
x4	0.12	0	0.55	0.71	0.74
x3	0.59	0.55	0	0.19	0.19
x1	0.73	0.71	0.19	0	0.16
X5	0.78	0.74	0.19	0.16	0

Is there structure in the data?

- **VAT** (Visual Assessment of [clustering] Tendency)

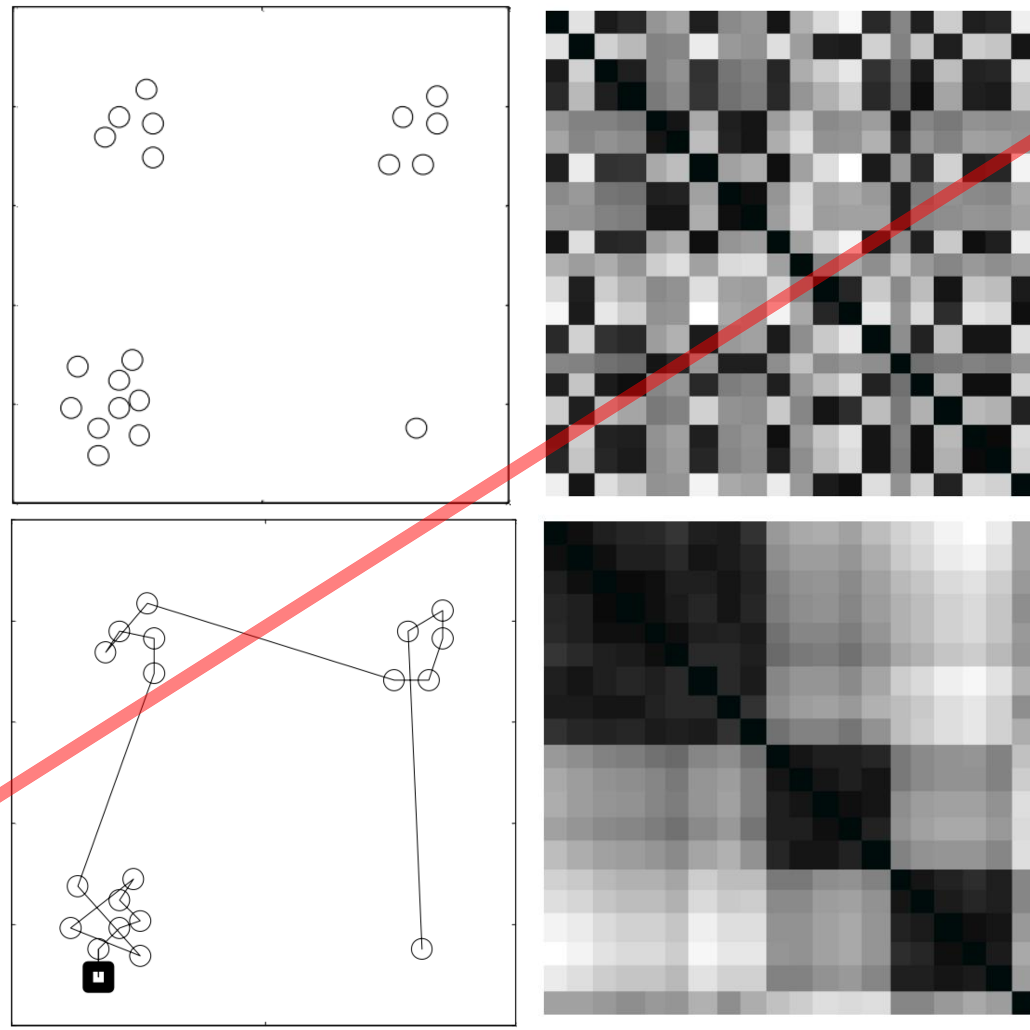
	x1	**x2**	**x3**	**x4**	**x5**
x1	0	0.73	0.19	0.71	0.16
x2	0.73	0	0.59	0.12	0.78
x3	0.19	0.59	0	0.55	0.19
x4	0.71	0.12	0.55	0	0.74
x5	0.16	0.78	0.19	0.74	0



	x2	**x4**	**x3**	**x1**	**x5**
x2	0	0.12	0.59	0.73	0.78
x4	0.12	0	0.55	0.71	0.74
x3	0.59	0.55	0	0.19	0.19
x1	0.73	0.71	0.19	0	0.16
X5	0.78	0.74	0.19	0.16	0

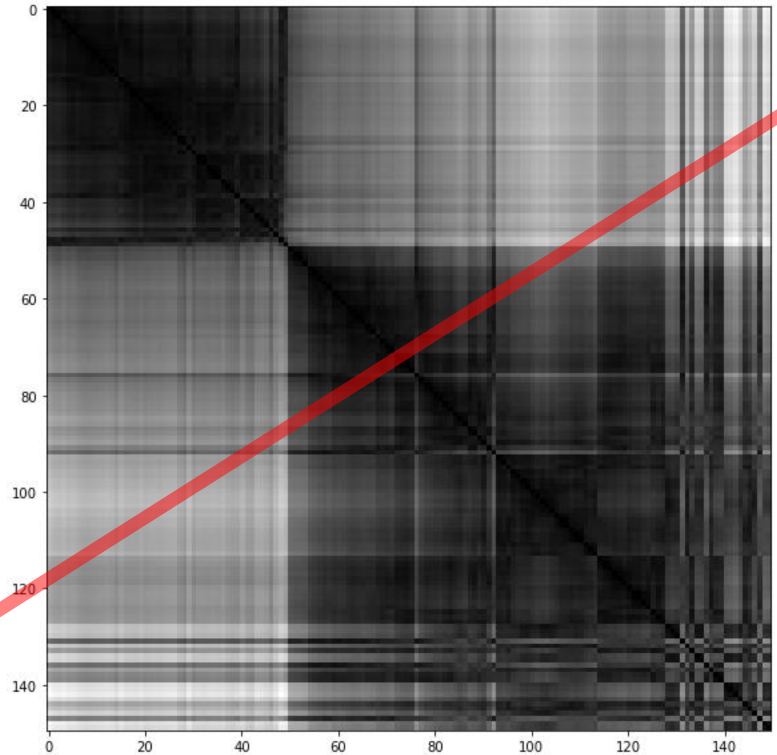
Is there structure in the data?

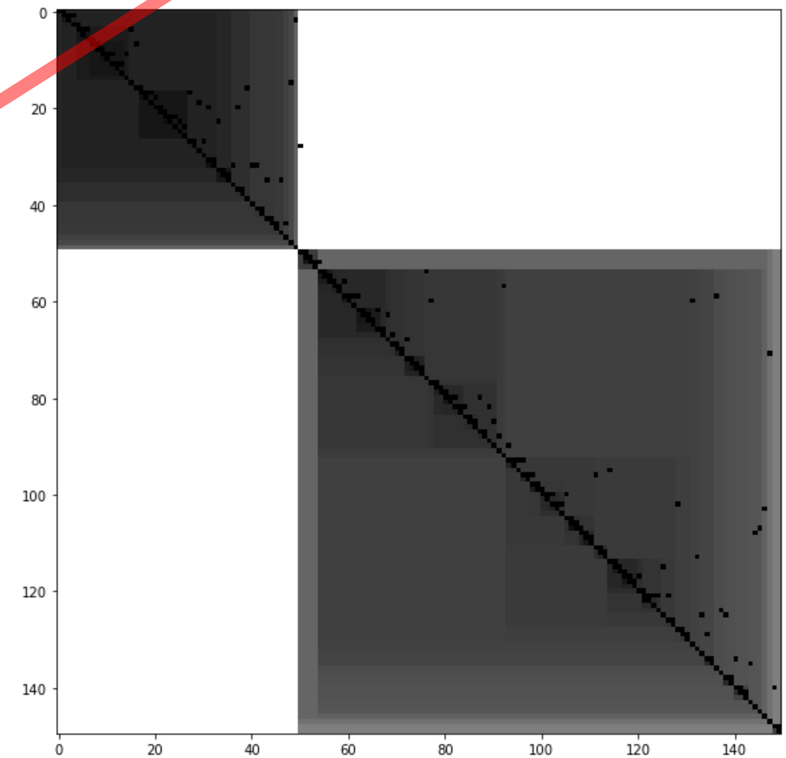
- **VAT** (Visual Assessment of [clustering] Tendency)



- **VAT** (Visual Assessment of [clustering] Tendency)
  - Example (Iris dataset):

```
from sklearn import datasets
from pyclustertend import vat, ivat
from sklearn.preprocessing import scale
X = scale(datasets.load_iris().data)
print(vat(X), ivat(X))
```





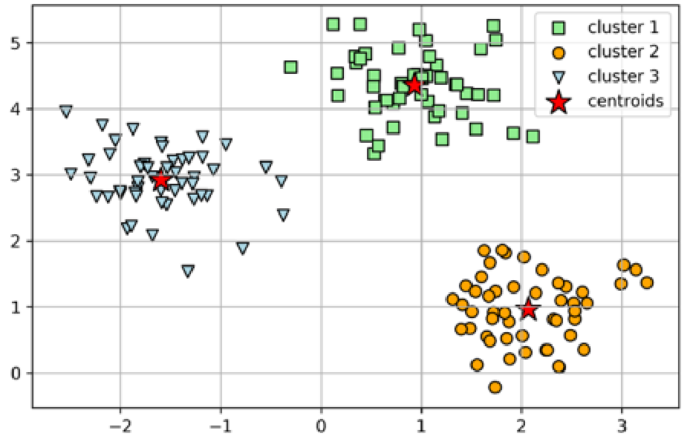
Contents

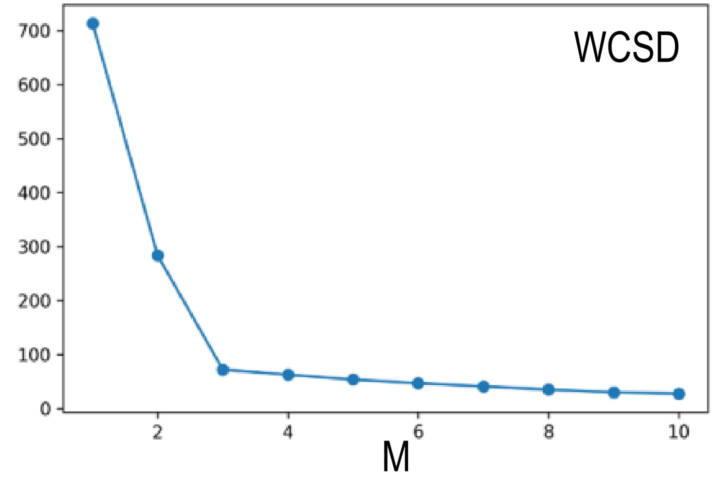
The elbow method and the silhouette coefficient

- The **elbow method** analyzes how clusters compactness varies as **the number of clusters M increases**, and selects the minimum M^* for which clusters compactness stops increasing
- **Compactness** is measured as the *within-cluster-sum of distances* (WCSD) for different values of M:

$$\mathrm{WCSD}(M) = \sum_{j=1}^{M} \sum_{x_i \in C_j} \wp(x_i, C_j)$$

- <u>Example</u>:

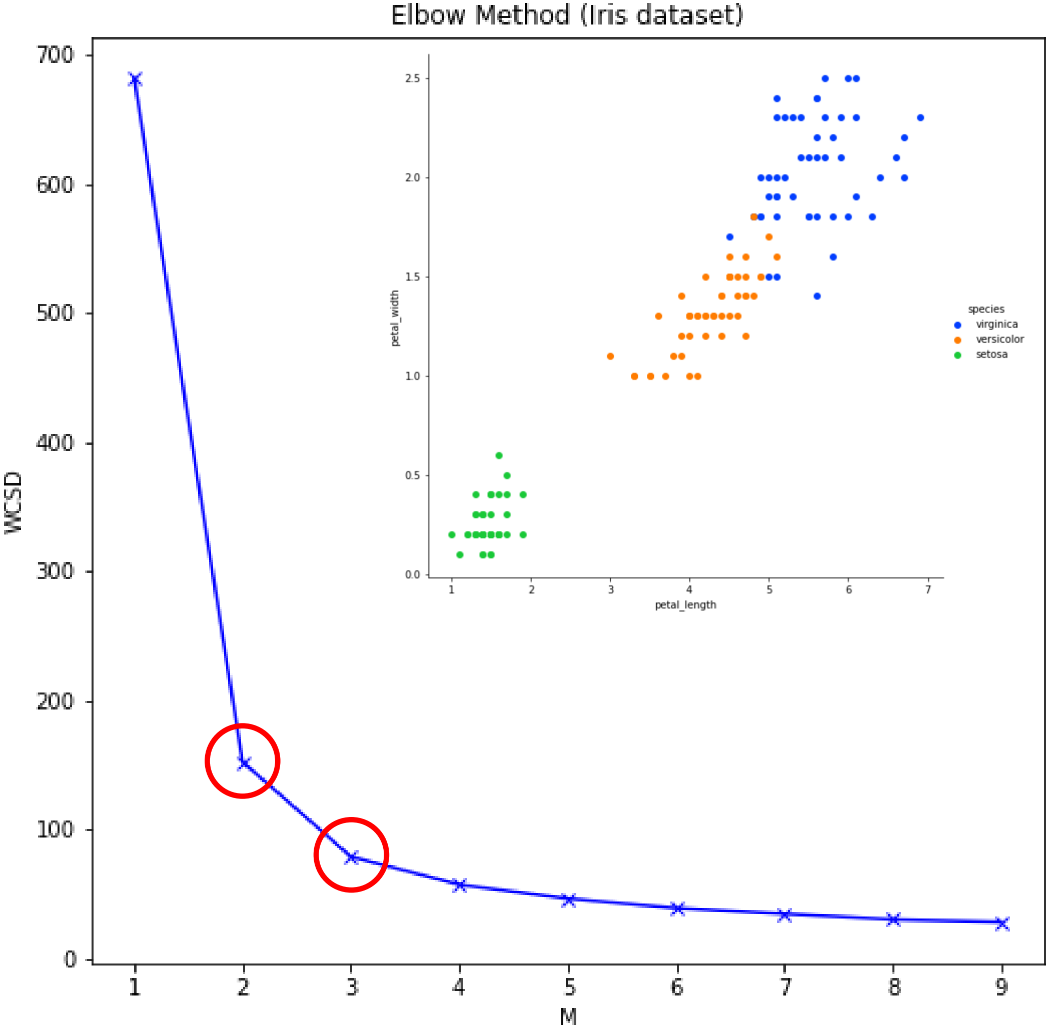




  – As expected for this example, WCSD decreases most for $M = 2$ and 3, while the rate of decrease gets almost 0 from $M = 3$. The plot looks as an arm and the critical point as an **elbow** (at $M = 3$).

The elbow method and the silhouette coefficient

- **<u>Example</u>: Elbow method**, k-means and Iris dataset




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets

iris = datasets.load_iris()
df = pd.DataFrame(iris['data'])

wcsd = []
M = range(1,10)
for j in M:
    kmeansModel = KMeans(n_clusters=j)
    kmeansModel.fit(df)
    wcsd.append(kmeansModel.inertia_)

plt.figure(figsize=(8,8))
plt.plot(M, wcsd, 'bx-')
plt.show()
```

The elbow method and the silhouette coefficient

- Unfortunately, we do not always have such clearly clustered data
  - This means that the elbow may not be that clear and sharp for each case
- In more ambiguous cases, we may use the **Silhouette index / coefficient**:

$$\text{given } x_i \in C_r:$$

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, +1] \quad [s(i) = 0 \text{ if } n_r = 1]$$

$$a(i) = \frac{1}{n_r - 1} \sum_{x_j \in C_r, i \neq j} \wp(x_i, x_j) \quad \text{(compactness)}$$

$$b(i) = \min_{s \neq r} \left\{ \frac{1}{n_s} \sum_{x_j \in C_s} \wp(x_i, x_j) \right\} \quad \text{(separation)}$$

  - $a(i)$ can be interpreted as a measure of **how well x_i is assigned to its cluster**
    - The smaller $a(i)$, the better is the assignment of x_i to its cluster ($\wp$ is DM)
  - $b(i)$ is the smallest mean distance of x_i to all points in any other cluster, of which x_i is not a member
    - The cluster with this smallest mean dissimilarity is said to be the **neighboring cluster** of x_i because it is the next best fit cluster for sample x_i
    - The larger $b(i)$, the better is the assignment of x_i to its cluster ($\wp$ is DM)

The elbow method and the silhouette coefficient

$$s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$$

- A $s(i)$ close to $+1$ means that the data is appropriately clustered:
  - A small value of $a(i)$ means x_i is similar to its own cluster and hence well clustered.
  - A large $b(i)$ means x_i is dissimilar to its neighbouring cluster.
- A $s(i)$ close to -1 indicates that x_i should be rather clustered in its neighbouring cluster.
- A $s(i)$ near zero means the sample is at the border of two natural clusters.

- The **mean of $s(i)$ over all points of a cluster** is a measure of the cluster compactness:

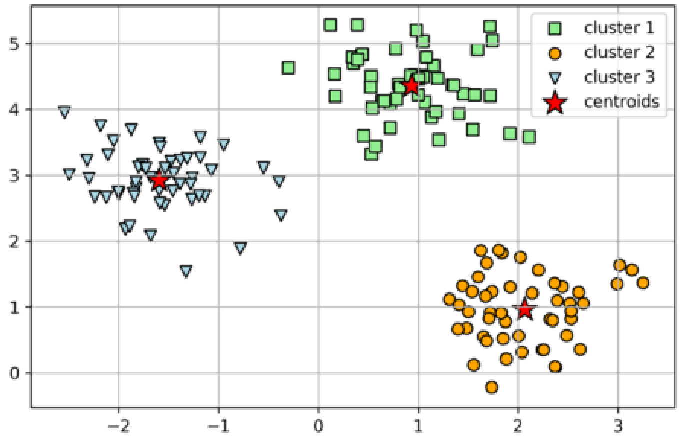
$$\mathrm{AVS}(k) = \frac{1}{n_k} \sum_{x_i \in C_k} s(i)$$

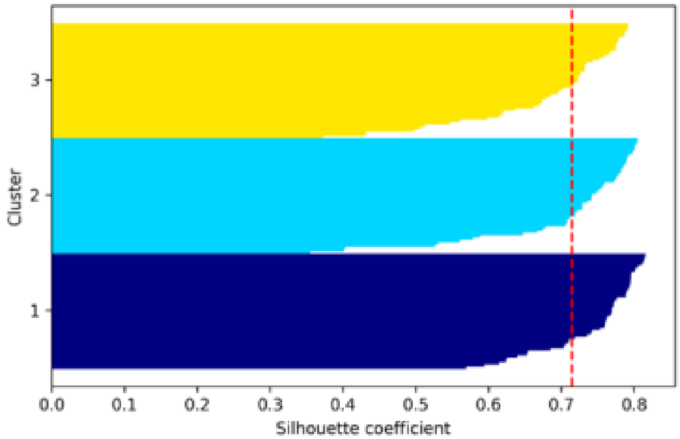
  - The closer to $+1$, the better
- The **mean of $s(i)$ over all data** of the entire dataset is a measure of how appropriately the data have been clustered:

$$\mathrm{AVS} = \frac{1}{M} \sum_{k}^{M} \mathrm{AVS}(k)$$

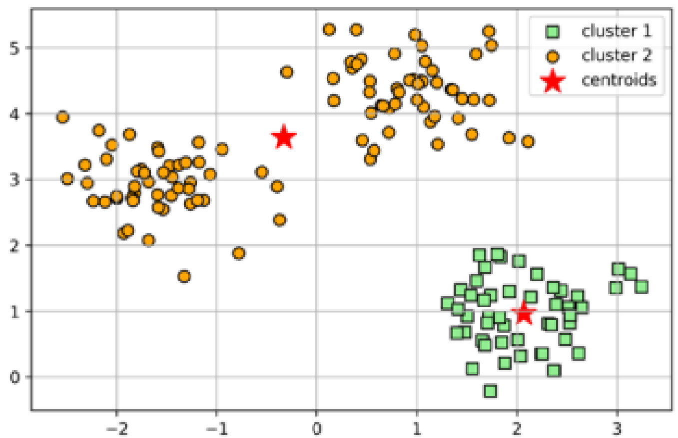
  - The closer to $+1$, the better

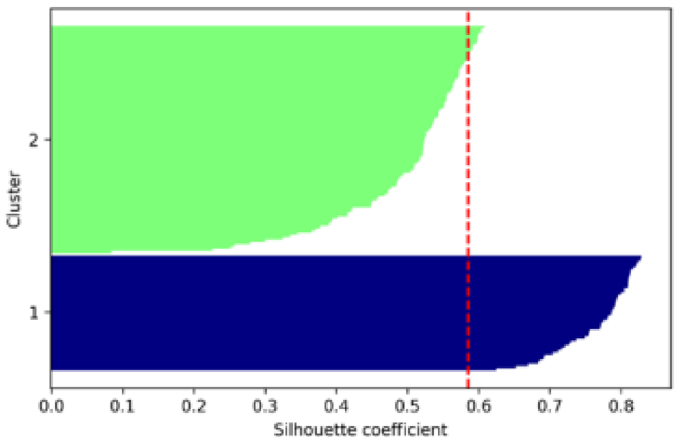
The elbow method and the silhouette coefficient





- If there are too many or too few clusters, as may occur for a poor choice of M, some of the clusters will typically display much narrower silhouettes than the rest.

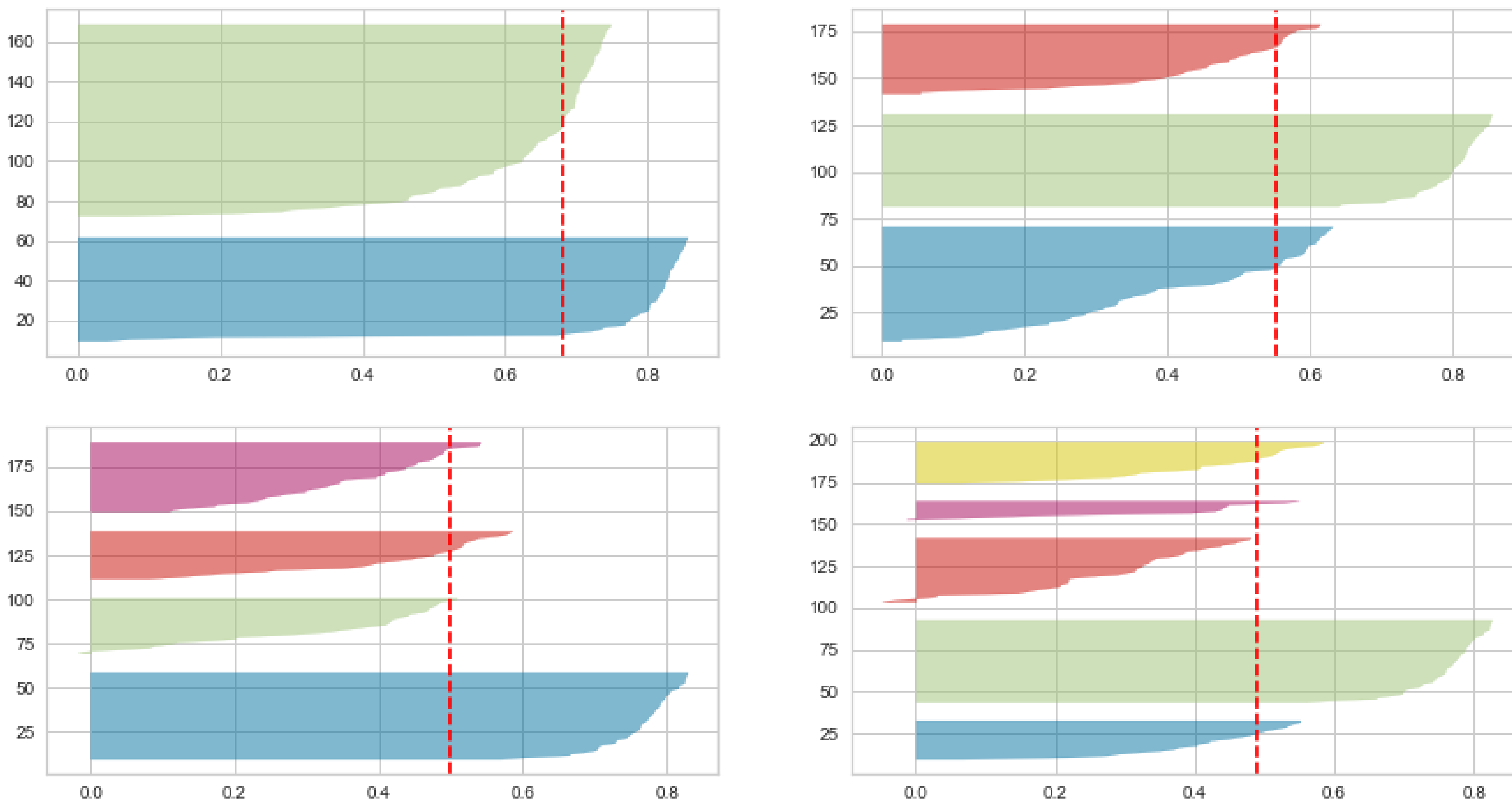




- Silhouette plots and averages can thus be used to determine the natural number of clusters within a dataset.

The elbow method and the silhouette coefficient

- **<u>Example</u>**: **Silhouette index**, k-means and Iris dataset



The elbow method and the silhouette coefficient

- **Example**: **Silhouette index**, k-means and Iris dataset

```
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from yellowbrick.cluster import SilhouetteVisualizer
iris = datasets.load_iris()
X = iris.data
y = iris.target
fig, ax = plt.subplots(2, 2, figsize=(15,8))
for i in [2, 3, 4, 5]:
    km = KMeans(n_clusters=i, init='k-means++', n_init=10, max_iter=100)
    q, mod = divmod(i, 2)
    visualizer = SilhouetteVisualizer(km, colors='yellowbrick', ax=ax[q-1][mod])
    visualizer.fit(X)

km = KMeans(n_clusters=3, random_state=42)
score = silhouette_score(X, km.labels_, metric='euclidean')
km.fit_predict(X)
print('Silhouette coefficient: %.3f' % score)
>>> Silhouette coefficient: 0.553
```

Contents

Dunn and Davies-Bouldin indices

- Cluster the dataset for different values of the number of clusters M and select the M^* that optimizes a certain expression involving the resulting clusters

  – **Davies-Bouldin index**:

$$\mathrm{DB}(M) = \frac{1}{M}\sum_{i=1}^{M} \max_{j \neq i}\left\{\frac{S_i + S_j}{\|\mu_i - \mu_j\|}\right\}$$

$$S_i = \sqrt{\frac{1}{n_i}\sum_{x \in C_i}\|x - \mu_i\|^2}$$

  - S_i^2 = intra-cluster variance (it is assumed the use of the Euclidean distance for measuring dissimilarity)
  - Compact and well-separated clusters $\Rightarrow DB \downarrow\downarrow$
  - Take the M^* that **minimizes** $DB(M)$

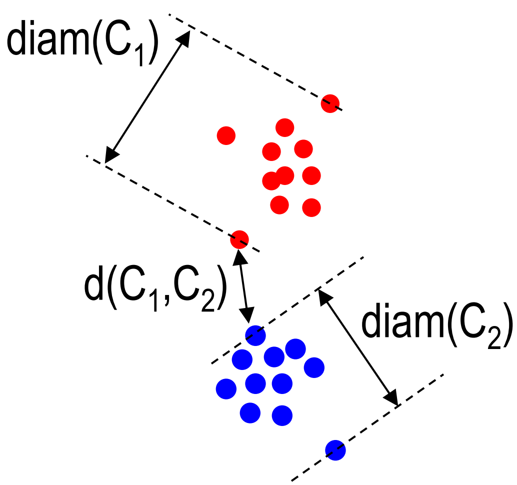
p.e. $$DB(2) = \frac{1}{2}\left(\overbrace{\max\left\{\frac{S_1+S_2}{\|\mu_1-\mu_2\|}\right\}}^{i=1} + \overbrace{\max\left\{\frac{S_2+S_1}{\|\mu_2-\mu_1\|}\right\}}^{i=2}\right)$$

p.e. $$DB(3) = \frac{1}{3}\left(\overbrace{\max\left\{\frac{S_1+S_2}{\|\mu_1-\mu_2\|}, \frac{S_1+S_3}{\|\mu_1-\mu_3\|}\right\}}^{i=1} + \overbrace{\max\left\{\frac{S_2+S_1}{\|\mu_2-\mu_1\|}, \frac{S_2+S_3}{\|\mu_2-\mu_3\|}\right\}}^{i=2} + \overbrace{\max\left\{\frac{S_3+S_1}{\|\mu_3-\mu_1\|}, \frac{S_3+S_2}{\|\mu_3-\mu_2\|}\right\}}^{i=3}\right)$$

Dunn and Davies-Bouldin indices

- Cluster the dataset for different values of the number of clusters M and select the M^* that optimizes a certain expression involving the resulting clusters

  – **Dunn index**:



$$\mathrm{DI}(M) = \min_{i=1,\ldots,M;j>i} \left\{ \frac{d(C_i, C_j)}{\max_{k=1,\ldots,M} \{\mathrm{diam}(C_k)\}} \right\}$$

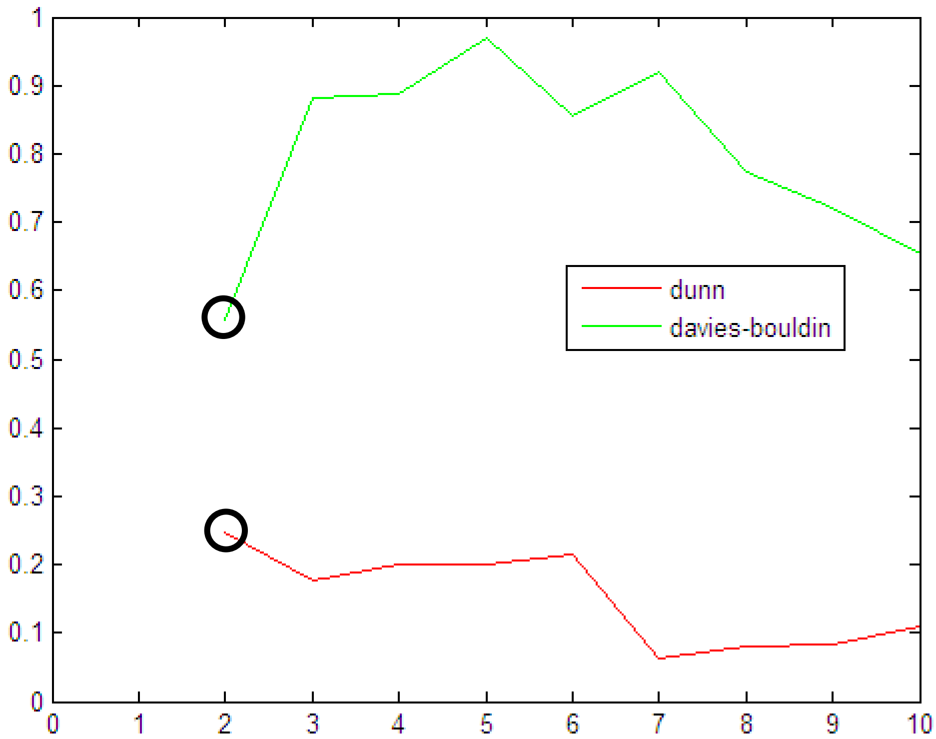
$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$$

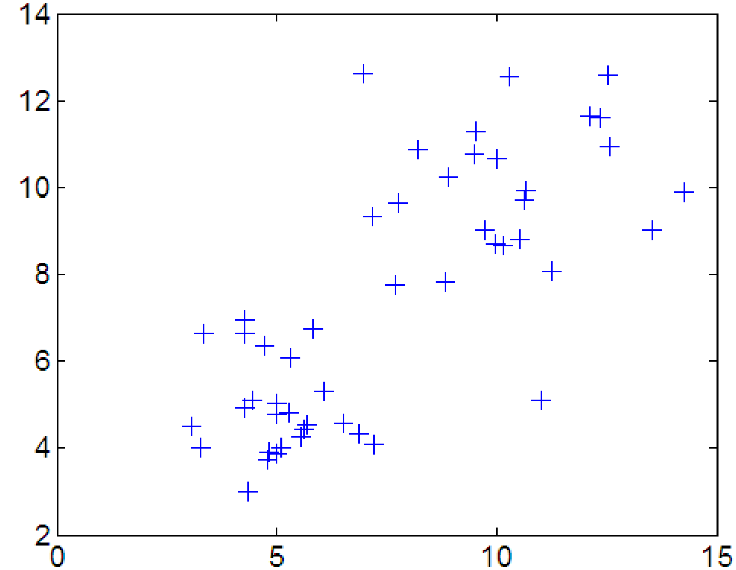
$$\mathrm{diam}(C_k) = \max_{x,y \in C_k} d(x, y)$$

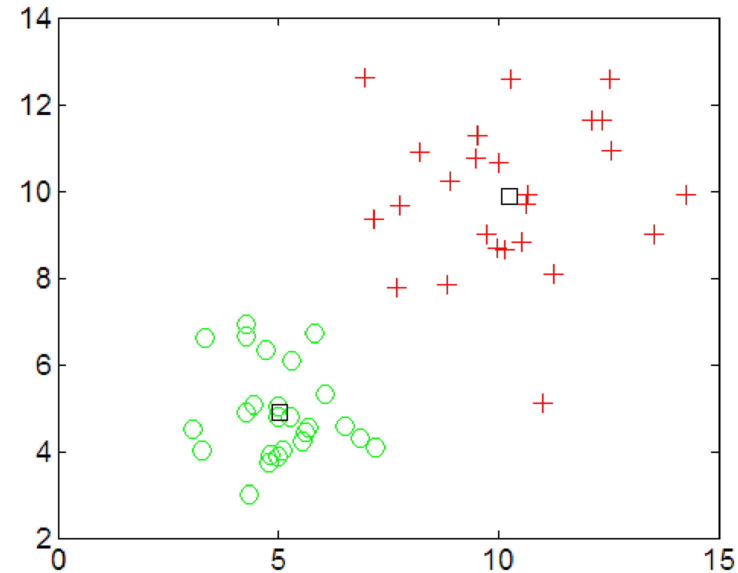
- compact and separated clusters $\Rightarrow DI \uparrow\uparrow$
- expressed for a **generic dissimilarity** d

- Choose M^* that maximizes **DI(M)**

- **Example 1**

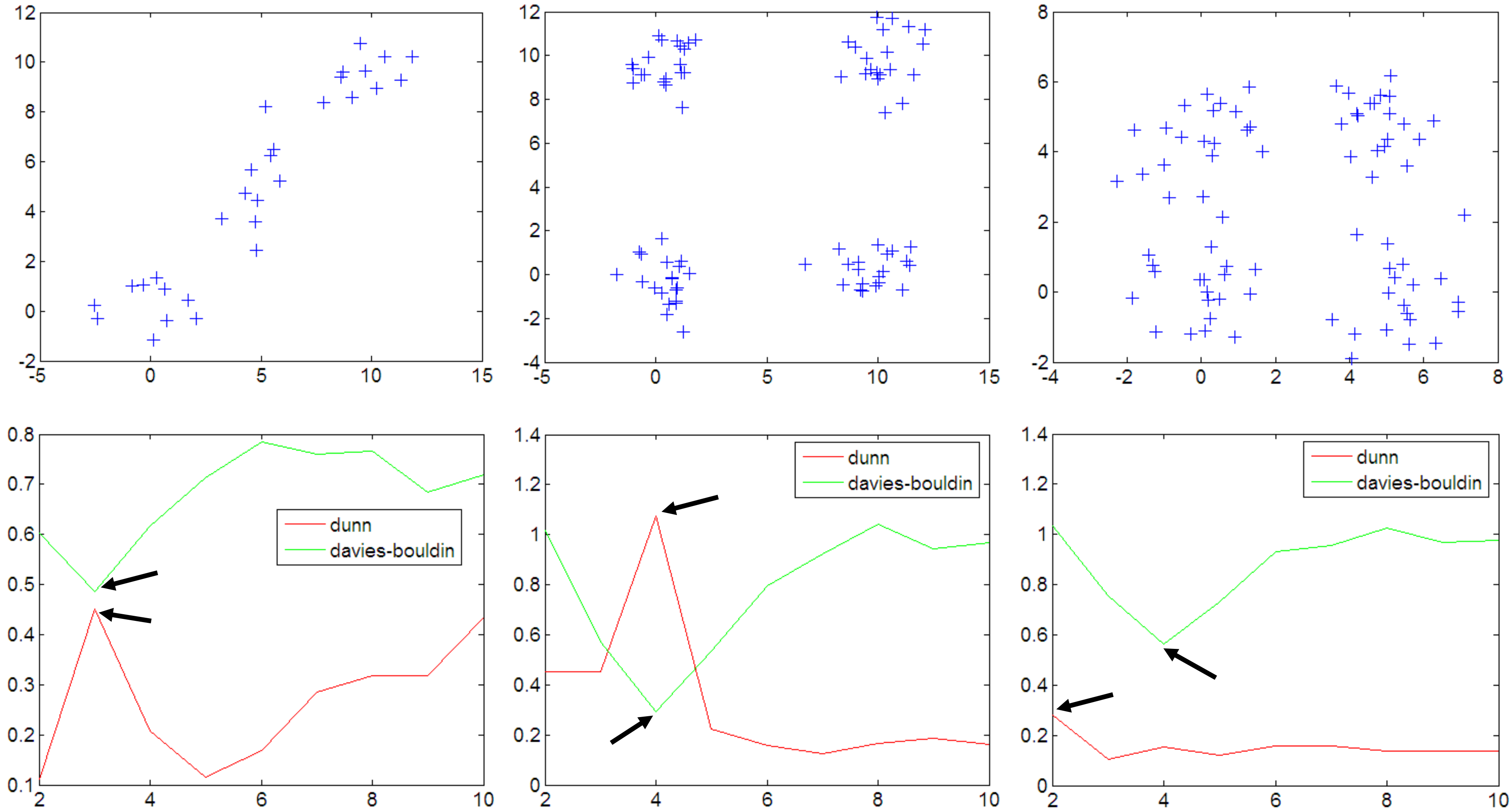

dunn
davies-bouldin
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0 1 2 3 4 5 6 7 8 9 10


2 4 6 8 10 12 14
0 5 10 15


2 4 6 8 10 12 14
0 5 10 15

Dunn and Davies-Bouldin indices

- **Example 2**



Dunn and Davies-Bouldin indices

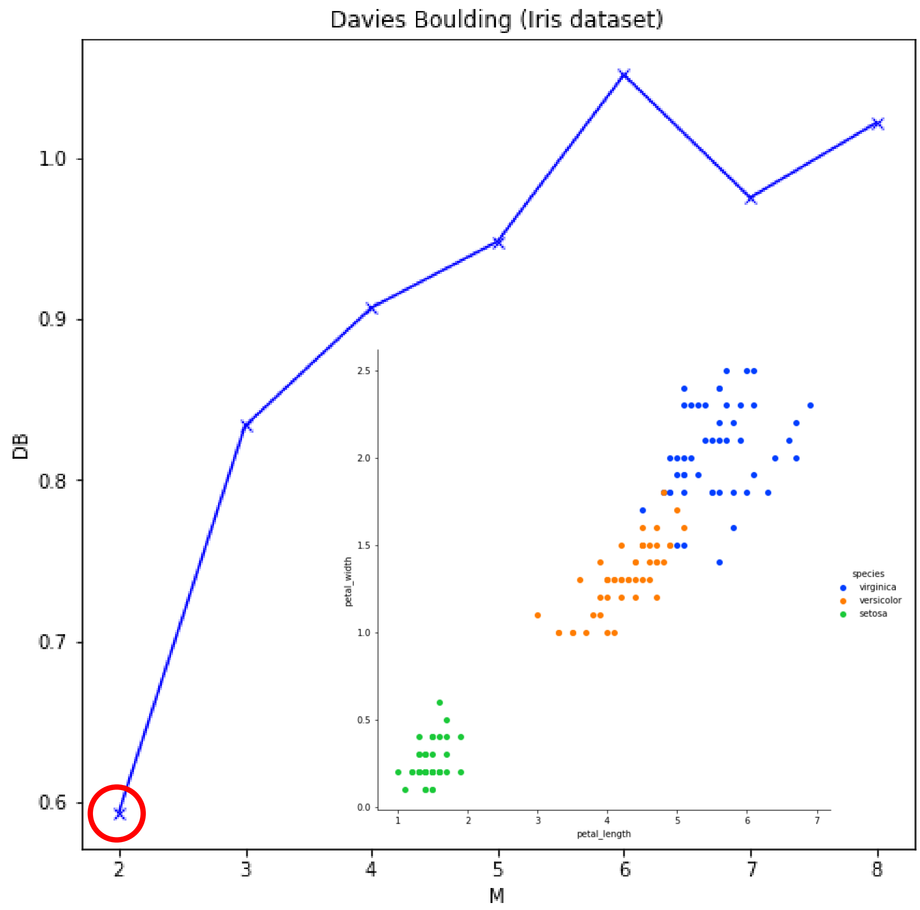
- **Example 3**: **Davis-Bouldin** index, k-means and Iris dataset

```
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale

iris = datasets.load_iris()
X = scale(iris.data)
y = iris.target

db = []
M = [2, 3, 4, 5, 6, 7, 8]
for j in M:
    km = KMeans(n_clusters=j, init='k-means++',
          n_init=10, max_iter=100)
    labels = km.fit_predict(X)
    db.append(davies_bouldin_score(X, labels))

plt.figure(figsize=(8,8))
plt.plot(M, db, 'bx-')
plt.show()
```



Contents

Homogeneity, completeness and V-measure

- The V-measure is the weighted harmonic mean of the **homogeneity** h and the **completeness** c of a clustering:

$$V_\beta = \frac{(1+\beta)h\,c}{\beta h + c}\,, \qquad \text{if } \beta = 1 \Rightarrow V = \frac{2\,h\,c}{h+c}$$

  - The V-measures has been proved to be equivalent to another metric, the so-called **Normalized Mutual Information** (NMI)
  - Homogeneity and completeness are defined on the basis of a clustering C and the true classes G, from the so-called **contingency table** $\rightarrow$

clustering C

true classes G	$\mathbf{C_1}$	$\mathbf{C_2}$	$\dots$	$\mathbf{C_K}$	
$\mathbf{G_1}$	$a_{1,1}$	$a_{1,2}$	$\dots$	$a_{1,K}$	$\leftarrow c$
$\mathbf{G_2}$	$a_{2,1}$	$a_{2,2}$	$\dots$	$a_{2,K}$	$\leftarrow c$
$\vdots$			$\ddots$		
$\mathbf{G_M}$	$a_{M,1}$	$a_{M,2}$	$\dots$	$a_{M,K}$	$\leftarrow c$
	$\uparrow h$	$\uparrow h$		$\uparrow h$	

    - the **homogeneity** h is maximized when each cluster contains elements of as few different classes as possible, ideally one single class $\rightarrow h = 1$
    - the **completeness** c is maximized when elements of each class lie in as few different clusters as possible, ideally one single cluster $\rightarrow c = 1$
    - V-measure for the ideal case is $v = 1$

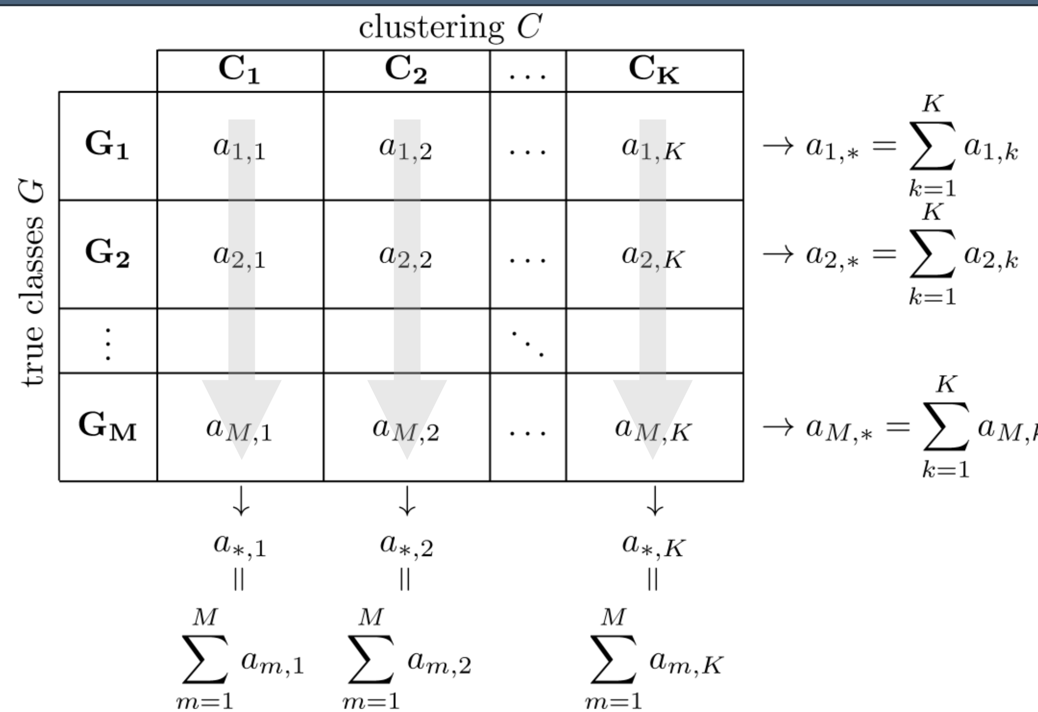
homogeneity

$$h = \begin{cases} 1 & \text{if } H(G) = 0 \\ 1 - \frac{H(G|C)}{H(G)} & \text{otherwise} \end{cases}$$

$$H(G|C) = -\sum_{k=1}^{K} \sum_{m=1}^{M} \frac{a_{m,k}}{N} \log \frac{a_{m,k}}{a_{*,k}}$$

$$H(G) = -\sum_{m=1}^{M} \frac{a_{m,*}}{N} \log \frac{a_{m,*}}{N}$$

$$\forall k, \exists m \;\Big|\; \frac{a_{m,k}}{a_{*,k}} = 1 \Rightarrow h = 1$$



entropy and conditional entropy

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

$$H(X|Y) = -\sum_{i=1,j=1}^{n,m} p(x_i, y_j) \log_2 \frac{p(x_i, y_j)}{p(y_j)}$$

Homogeneity, completeness and V-measure

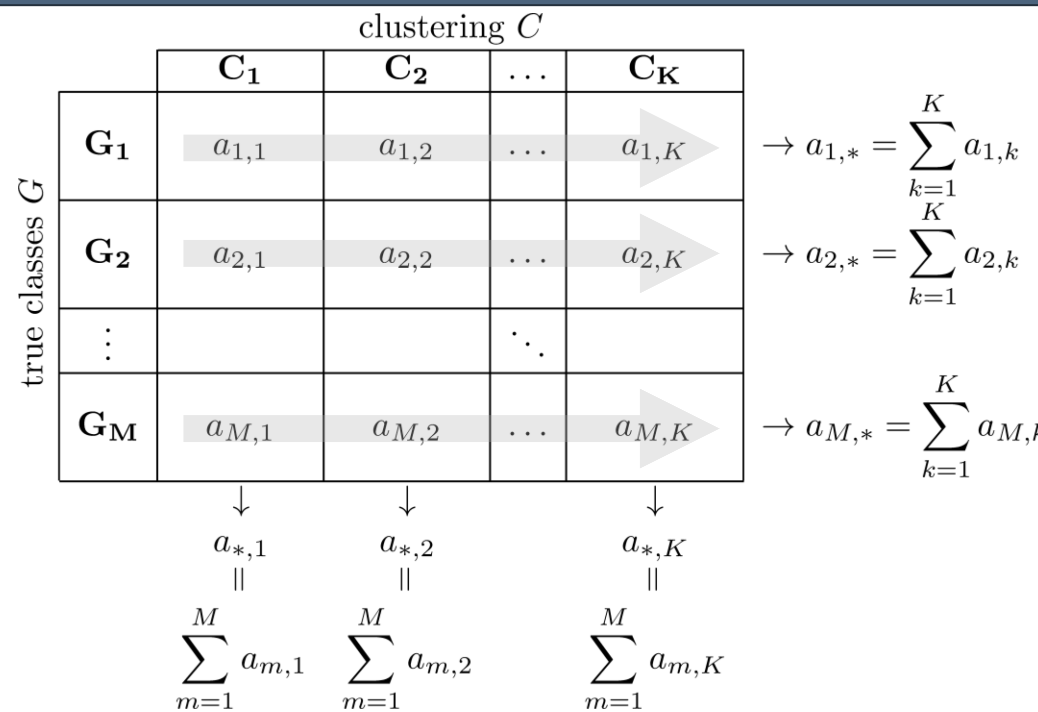
completeness

$$c = \begin{cases} 1 & \text{if } H(C) = 0 \\ 1 - \frac{H(C|G)}{H(C)} & \text{otherwise} \end{cases}$$

$$H(C|G) = -\sum_{k=1}^{K} \sum_{m=1}^{M} \frac{a_{m,k}}{N} \log \frac{a_{m,k}}{a_{m,*}}$$

$$H(C) = -\sum_{k=1}^{K} \frac{a_{*,k}}{N} \log \frac{a_{*,k}}{N}$$

$$\forall m, \exists k \,\middle|\, \frac{a_{m,k}}{a_{m,*}} = 1 \Rightarrow c = 1$$



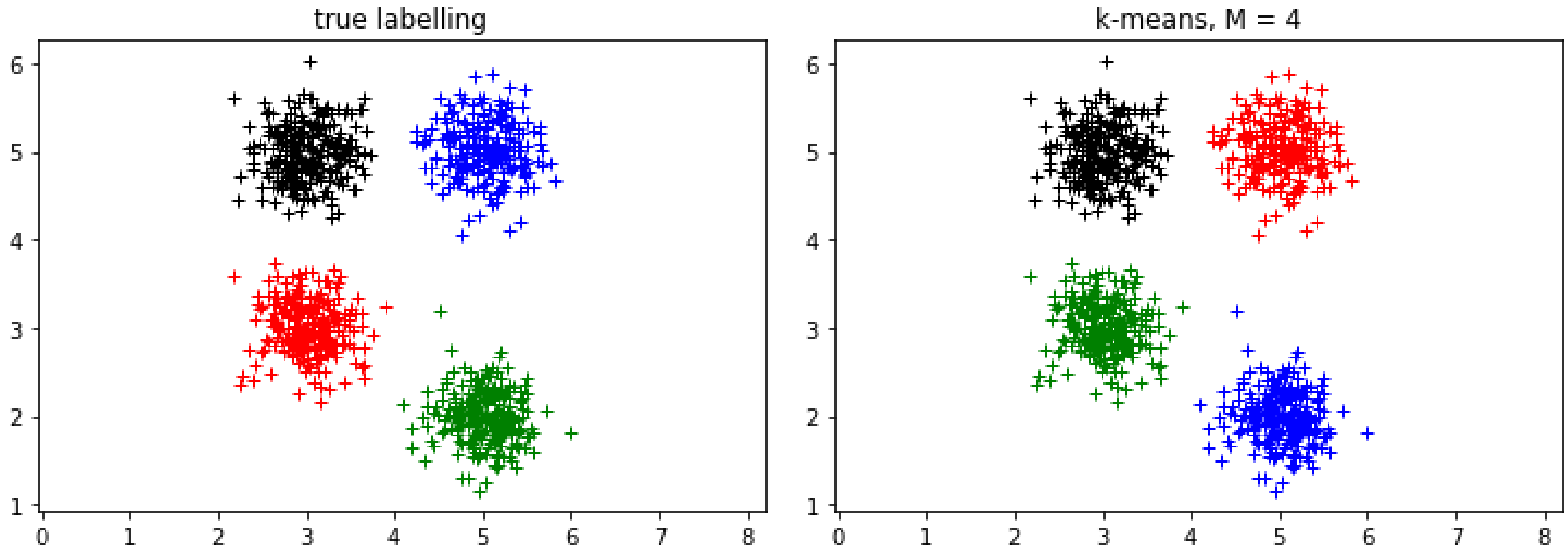
entropy and conditional entropy

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

$$H(X|Y) = -\sum_{i=1,j=1}^{n,m} p(x_i, y_j) \log_2 \frac{p(x_i, y_j)}{p(y_j)}$$

Homogeneity, completeness and V-measure

- <u>Example</u>: 4 classes, 250 samples/class




```
km = KMeans(n_clusters=4, init='k-means++', n_init=10, max_iter=100)
km.fit_predict(X)
cm = contingency_matrix(y, km.labels_)
print(cm)
s = homogeneity_completeness_v_measure(y, km.labels_, beta=1.0)
print('h = ', s[0], ', c = ', s[1], ', v = ', s[2])
```

(perform proper imports!)

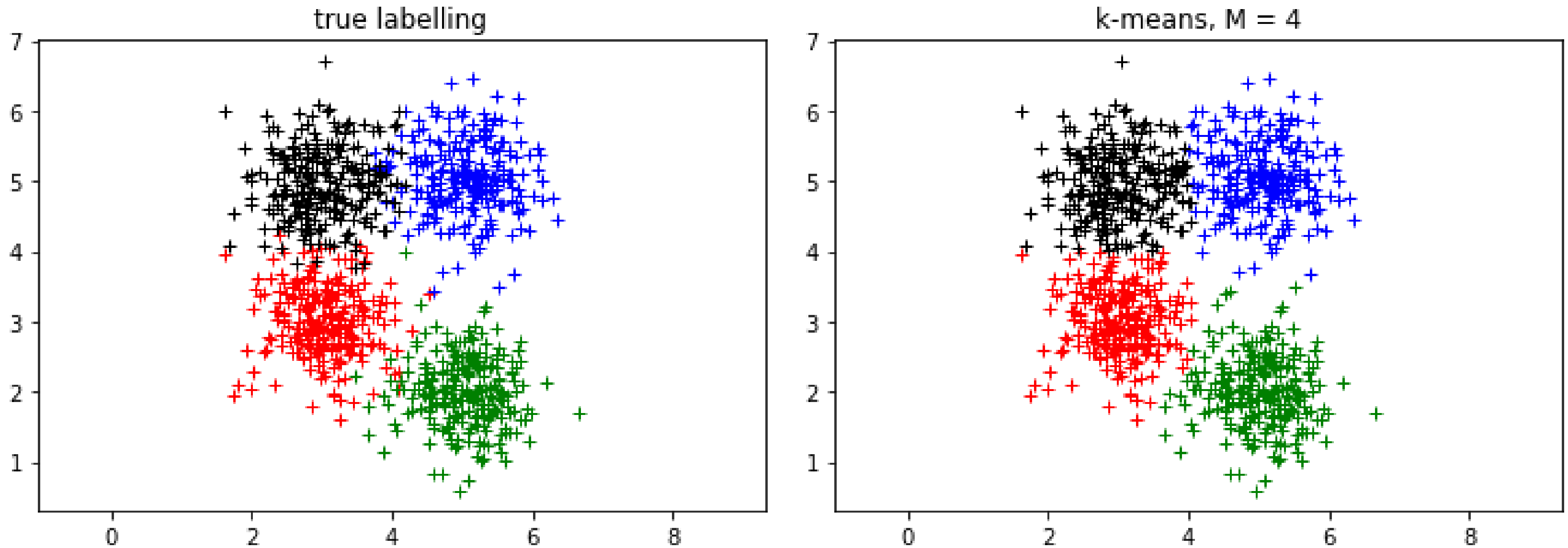
<u>results</u>:

```
[[  0 250   0   0]
 [  0   0 250   0]
 [250   0   0   0]
 [  0   0   0 250]]

h =  1.0,
c =  1.0,
v =  1.0
```

Homogeneity, completeness and V-measure

- <u>Example</u>: 4 classes, 250 samples/class




```
km = KMeans(n_clusters=4, init='k-means++', n_init=10, max_iter=100)
km.fit_predict(X)
cm = contingency_matrix(y, km.labels_)
print(cm)
s = homogeneity_completeness_v_measure(y, km.labels_, beta=1.0)
print('h = ', s[0], ', c = ', s[1], ', v = ', s[2])
```

(perform proper imports!)

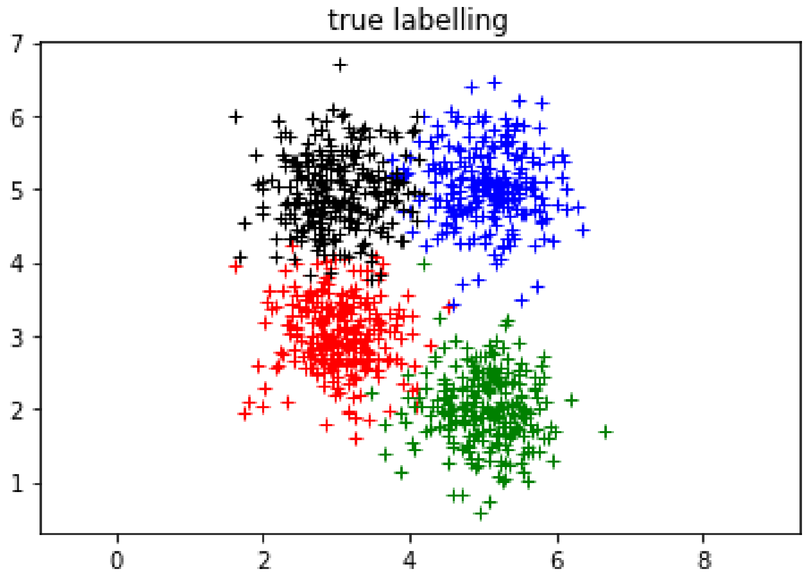
<u>results</u>:

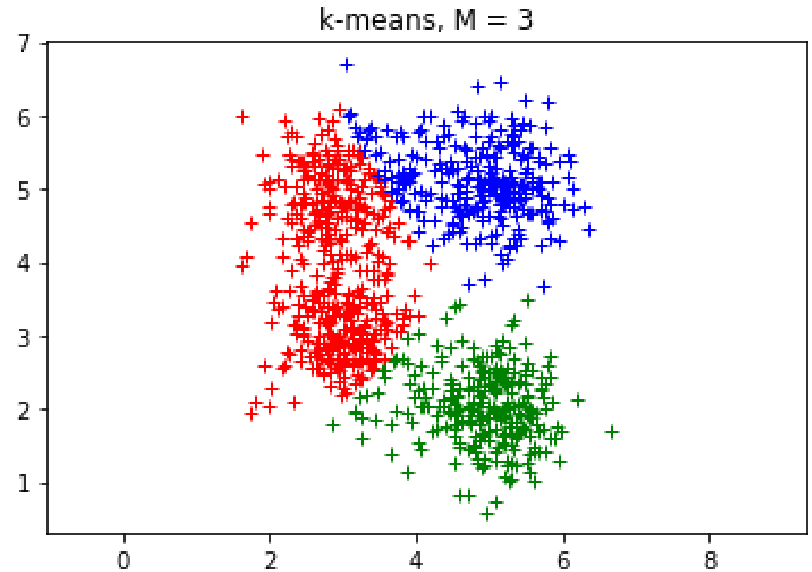
```
[[238   7   0   5]
 [  2 247   1   0]
 [  0   2 239   9]
 [  4   0   7 239]]

h =  0.8721128057576535,
c =  0.8722260670609913,
v =  0.8721694327322493
```

Homogeneity, completeness and V-measure

- <u>Example</u>: 4 classes, 250 samples/class






```
km = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=100)
km.fit_predict(X)
cm = contingency_matrix(y, km.labels_)
print(cm)
s = homogeneity_completeness_v_measure(y, km.labels_, beta=1.0)
print('h = ', s[0], ', c = ', s[1], ', v = ', s[2])
```

(perform proper imports!)

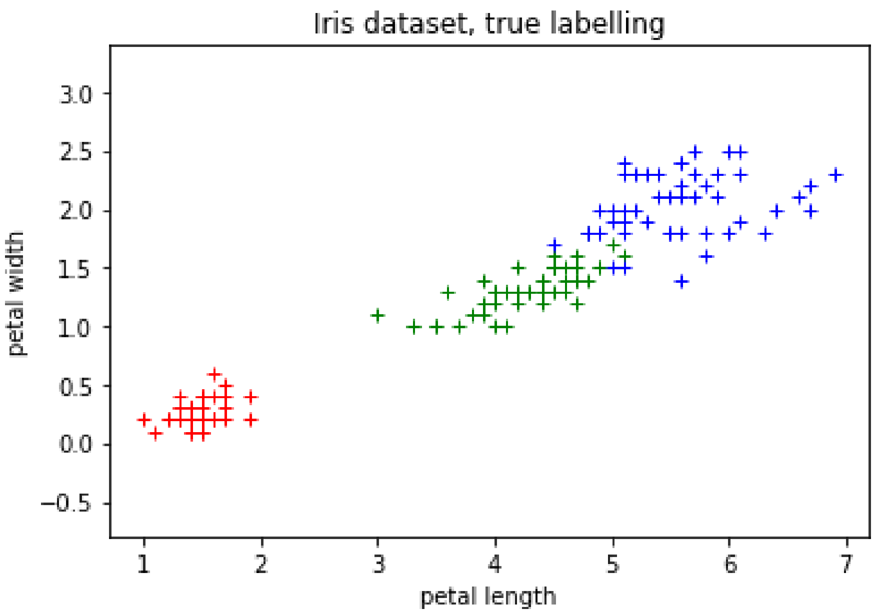
<u>results</u>:

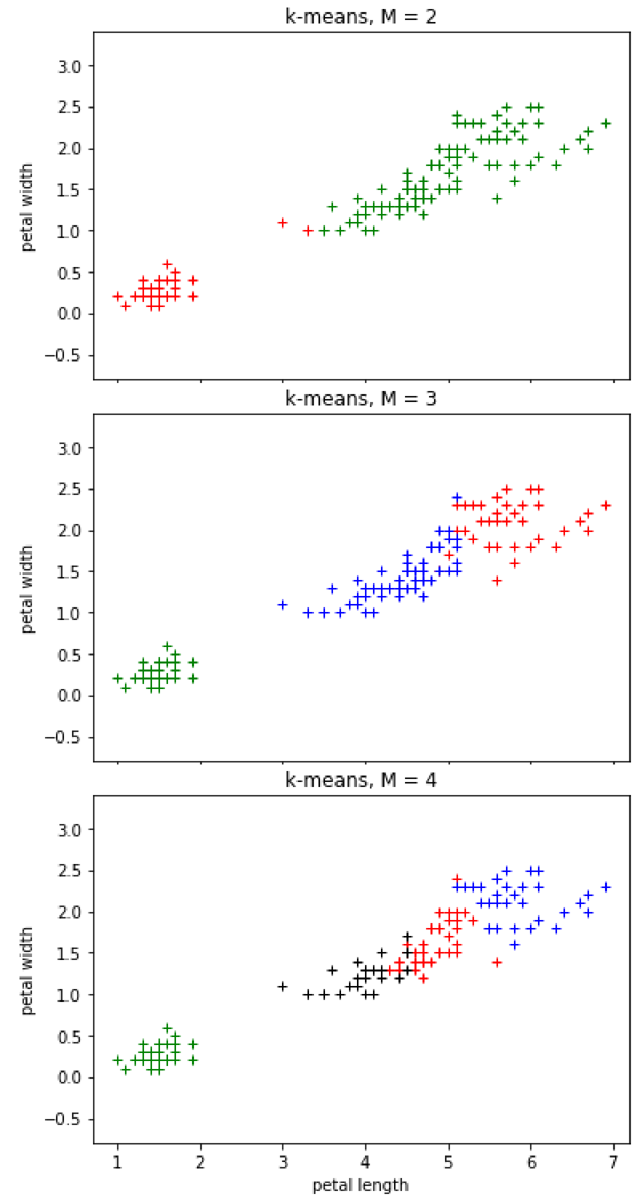
```
[[226   0  24]
 [  1   0 249]
 [  0 248   2]
 [202  48   0]]

h =  0.6195907856538674,
c =  0.7964668735209744,
v =  0.6969822629958449
```

Homogeneity, completeness and V-measure

- <u>Example</u> (Iris dataset):





results (M = 2)	**results** (M = 3)	**results** (M = 4)
[[50 0] [3 47] [0 50]]	[[0 50 0] [2 0 48] [36 0 14]]	[[0 50 0 0] [23 0 0 27] [17 0 32 1]]
h = 0.5223, c = 0.8835, v = 0.6565	h = 0.7515, c = 0.7650, v = 0.7582	h = 0.8083, c = 0.6522, v = 0.7219

Unsupervised Learning: Clustering validity

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ