# MATH60026/MATH70026
# **Methods for Data Science**
# Lecture 7

Barbara Bravi, Imperial College London

Department of Mathematics, Academic year 2024-2025

IMPERIAL

# Supervised vs Unsupervised

Data available:

$$\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$$

Multiple-feature data + **labels**

$$\mathbf{x}^{(i)} \in \mathbb{R}^p \quad \text{or} \quad \mathbf{x}^{(i)} \in A_1 \times A_2 \cdots \times A_p$$

Continuous or categorical

$$\begin{cases} y^{(i)} \in \mathbb{R} & \text{(regression)} \\ \text{or} & \\ y^{(i)} \in \{c_1, \ldots, c_Q\} & \text{(classification)} \end{cases}$$

# Supervised vs Unsupervised

Data available:

$$\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$$

Multiple-feature data + **labels**

$$\mathbf{x}^{(i)} \in \mathbb{R}^p \quad \text{or} \quad \mathbf{x}^{(i)} \in A_1 \times A_2 \cdots \times A_p$$

Continuous or categorical

$$\begin{cases} y^{(i)} \in \mathbb{R} & \text{(regression)} \\ \text{or} & \\ y^{(i)} \in \{c_1, \ldots, c_Q\} & \text{(classification)} \end{cases}$$

We are interested in: making a prediction

$$\text{Given } \mathbf{x}^{\text{in}}, \quad \text{predict } \widehat{y} = \begin{cases} f_{\boldsymbol{\theta}}(\mathbf{x}^{\text{in}}) \in \mathbb{R} & \text{(regression)} \\ \text{or} & \\ f_{\boldsymbol{\theta}}(\mathbf{x}^{\text{in}}) \in \{c_1, \ldots, c_Q\} & \text{(classification)} \end{cases}$$

learning the optimal parameters for the prediction task

# Supervised vs Unsupervised

Data available:

$$\{\mathbf{x}^{(i)}\}_{i=1}^{N}$$ Multiple-feature data

$$\mathbf{x}^{(i)} \in \mathbb{R}^p \quad \text{or} \quad \mathbf{x}^{(i)} \in A_1 \times A_2 \cdots \times A_p$$

Continuous or categorical

We are interested in: learning intrinsic data properties

# Supervised vs Unsupervised

Data available:

$$\{\mathbf{x}^{(i)}\}_{i=1}^{N}$$

Multiple-feature data

$$\mathbf{x}^{(i)} \in \mathbb{R}^p \quad \text{or} \quad \mathbf{x}^{(i)} \in A_1 \times A_2 \cdots \times A_p$$

Continuous or categorical

We are interested in: learning intrinsic data properties

How?

Finding **informative similarities** between data points,

Finding **convenient representations** for inspection, etc.

# Supervised vs Unsupervised

Data available:

$$\{\mathbf{x}^{(i)}\}_{i=1}^{N}$$ Multiple-feature data

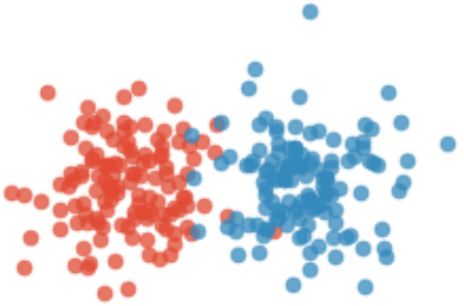
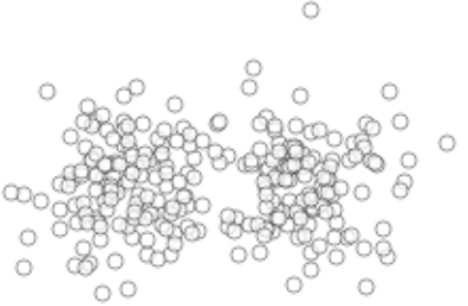
$$\mathbf{x}^{(i)} \in \mathbb{R}^p \quad \text{or} \quad \mathbf{x}^{(i)} \in A_1 \times A_2 \cdots \times A_p$$

Continuous or categorical

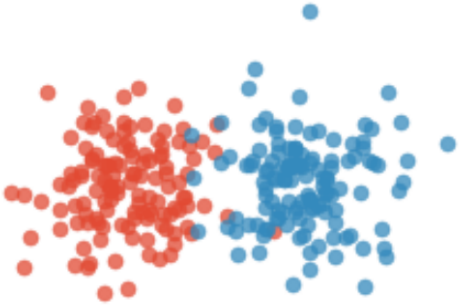
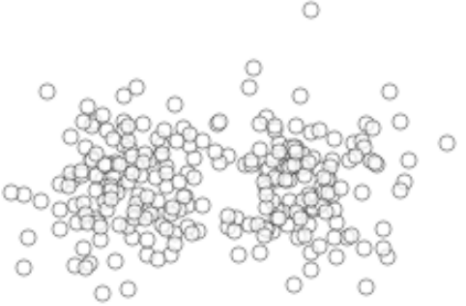
We are interested in: learning intrinsic data properties
How?
Finding **informative similarities** between data points,
Finding **convenient representations** for inspection, etc.

Key: a posteriori step of **interpretation & inspection**

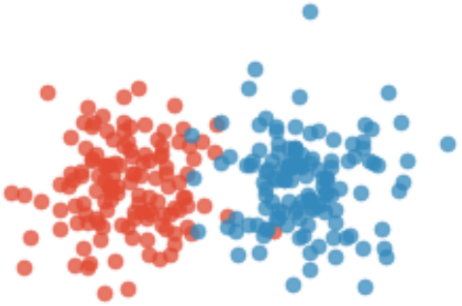
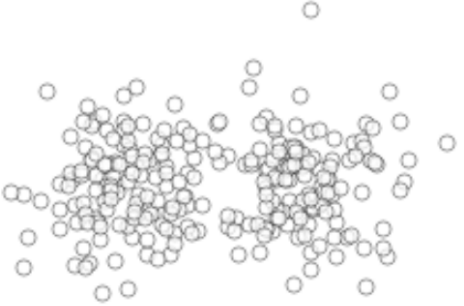
# Learning paradigms

| | Supervised | Unsupervised |
|---|---|---|
| Data | $\mathbf{X}, \boldsymbol{y}$  | $\mathbf{X}$  |
| Aim | Prediction $\boldsymbol{y} = f(\mathbf{X}; \boldsymbol{\theta})$ | Representation $f(\mathbf{X}; \boldsymbol{\theta})$ |
| Tasks | Classification<br>Regression | |

# Learning paradigms

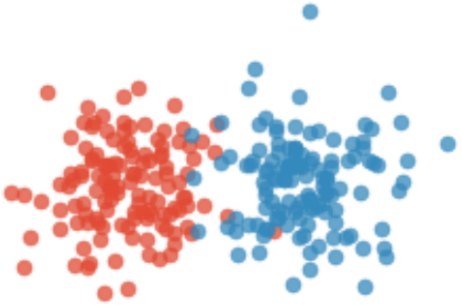
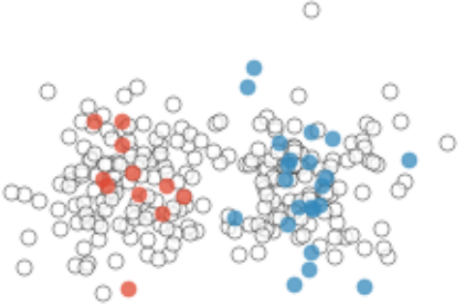
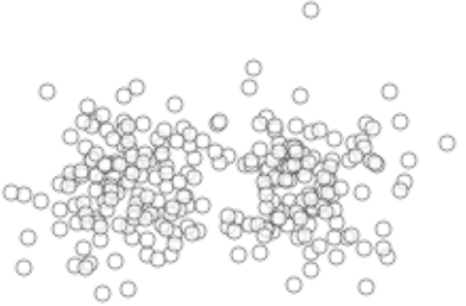
| | Supervised | Unsupervised |
|---|---|---|
| Data | $\mathbf{X}, \boldsymbol{y}$  | $\mathbf{X}$  |
| Aim | Prediction $\boldsymbol{y} = f(\mathbf{X}; \boldsymbol{\theta})$ | Representation $f(\mathbf{X}; \boldsymbol{\theta})$ |
| Tasks | Classification<br>Regression | Clustering<br>Dimensionality reduction<br>Density estimation |

# Learning paradigms

| | Supervised | Unsupervised |
|---|---|---|
| Data | $\mathbf{X}, \boldsymbol{y}$  | $\mathbf{X}$  |
| Aim | Prediction $\boldsymbol{y} = f(\mathbf{X}; \boldsymbol{\theta})$ | Representation $f(\mathbf{X}; \boldsymbol{\theta})$ |
| Tasks | Classification<br>Regression | Clustering<br>Dimensionality reduction<br>Density estimation |

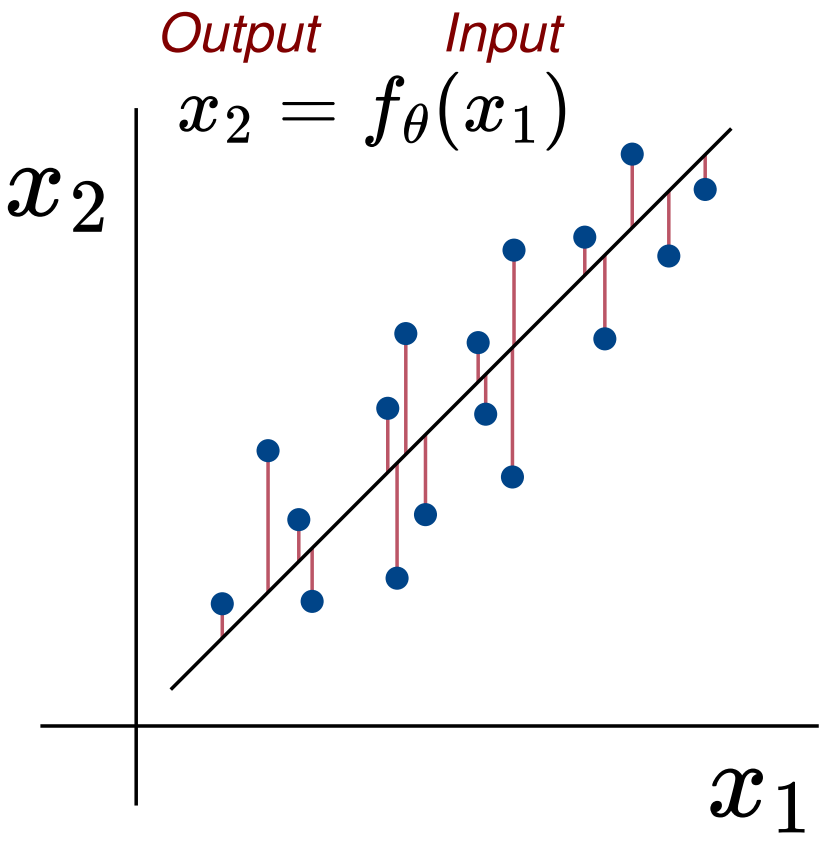
*Graph-based techniques helping further!*

# Learning paradigms

| | Supervised | Semi-supervised | Unsupervised |
|---|---|---|---|
| Data | $\mathbf{X}, \boldsymbol{y}$ | $\mathbf{X}, \boldsymbol{y}, \mathbf{X}_u$ | $\mathbf{X}$ |
| |  |  |  |
| Aim | Prediction<br>$\boldsymbol{y} = f(\mathbf{X}; \boldsymbol{\theta})$ | $\longleftrightarrow$ | Representation<br>$f(\mathbf{X}; \boldsymbol{\theta})$ |
| Tasks | Classification<br>Regression | $\longleftrightarrow$ | Clustering<br>Dimensionality reduction<br>Density estimation |

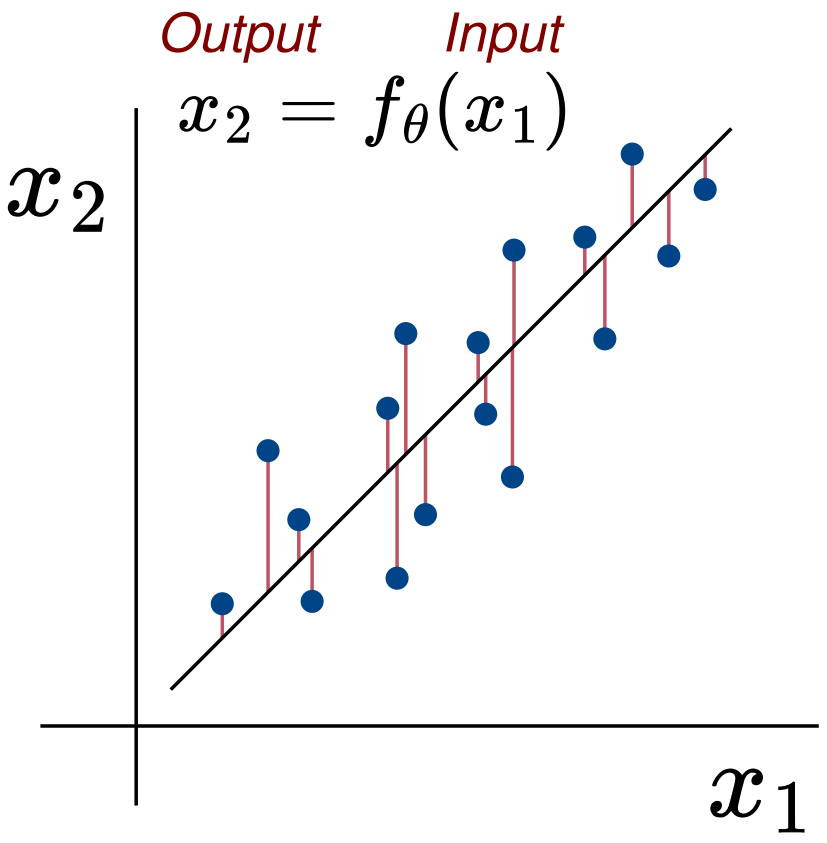
*Graph-based techniques helping further!*

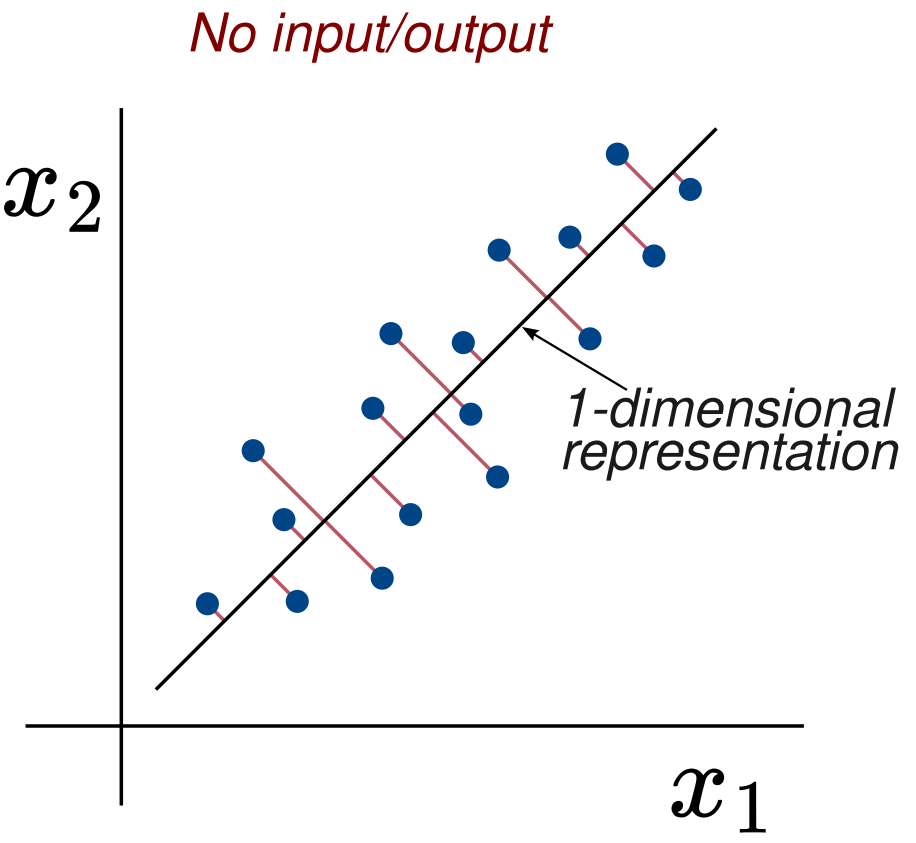
# Regression vs Dimensionality Reduction




e.g. linear regression: find a regression line **minimising** the squared prediction error.

# Regression vs Dimensionality Reduction

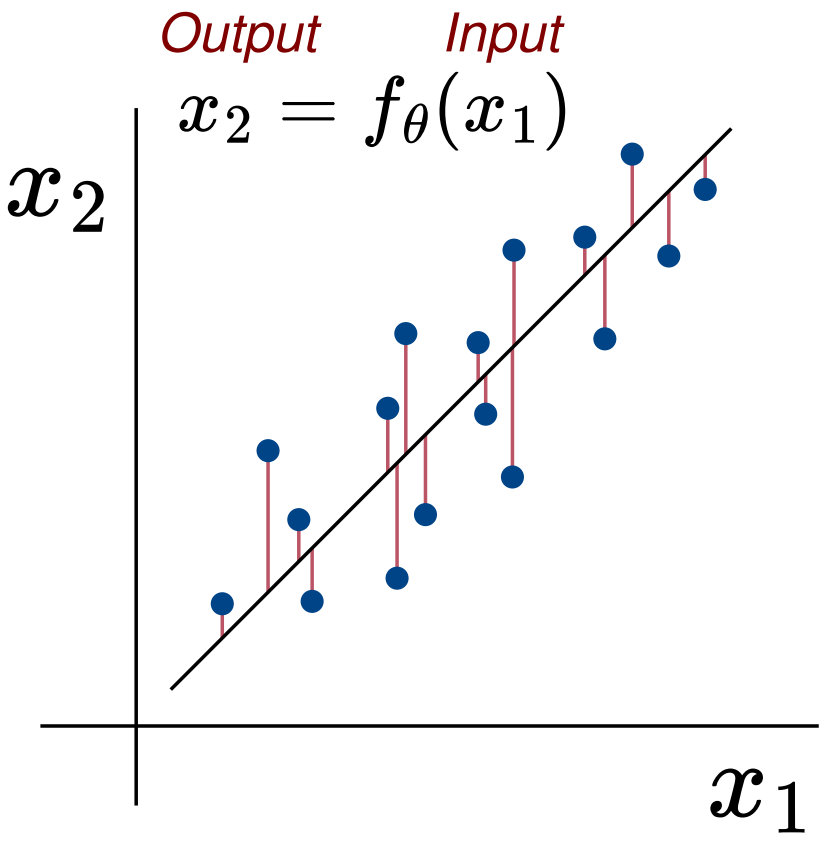


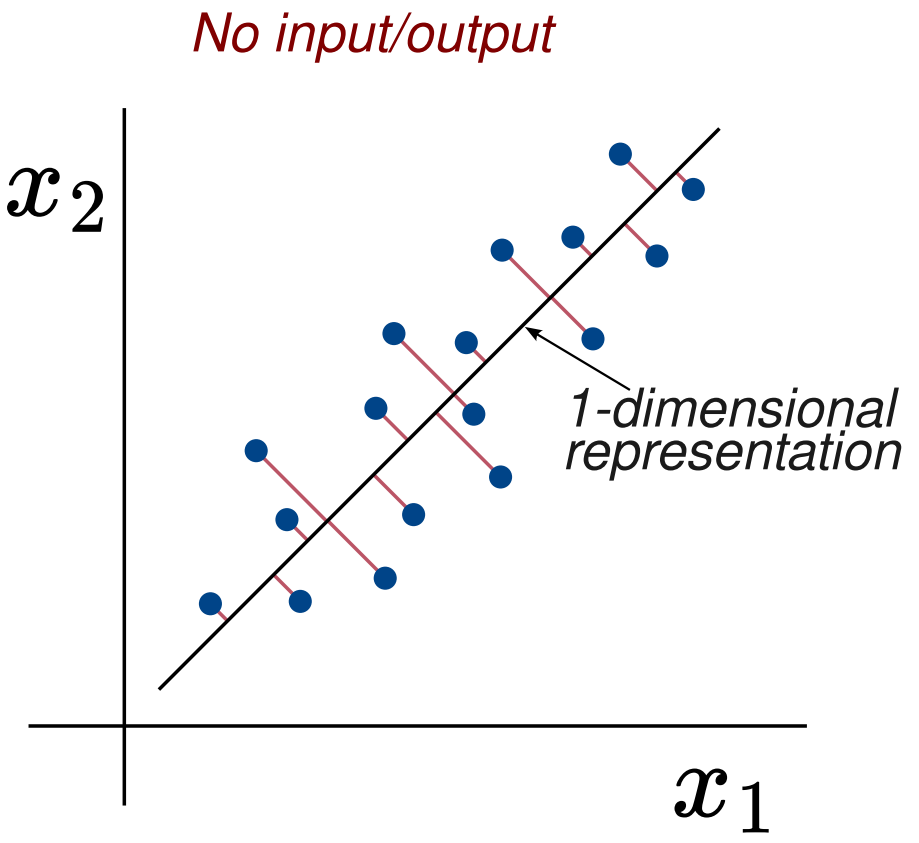




e.g. linear regression: find a regression line **minimising** the squared prediction error.

e.g. PCA (linear dim. red.): find the one-dimensional representation that **maximises** the data variance.

# Regression vs Dimensionality Reduction



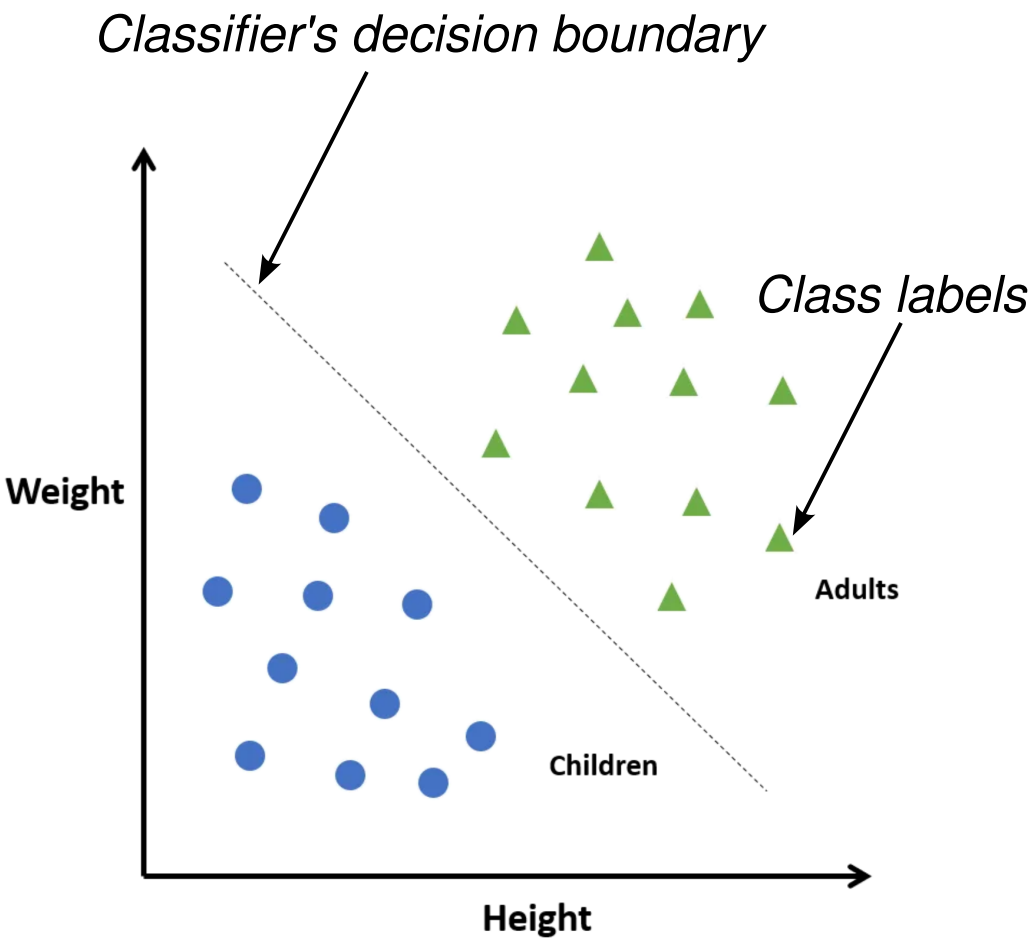




e.g. linear regression: find a regression line **minimising** the squared prediction error.

e.g. PCA (linear dim. red.): find the one-dimensional representation that **maximises** the data variance.

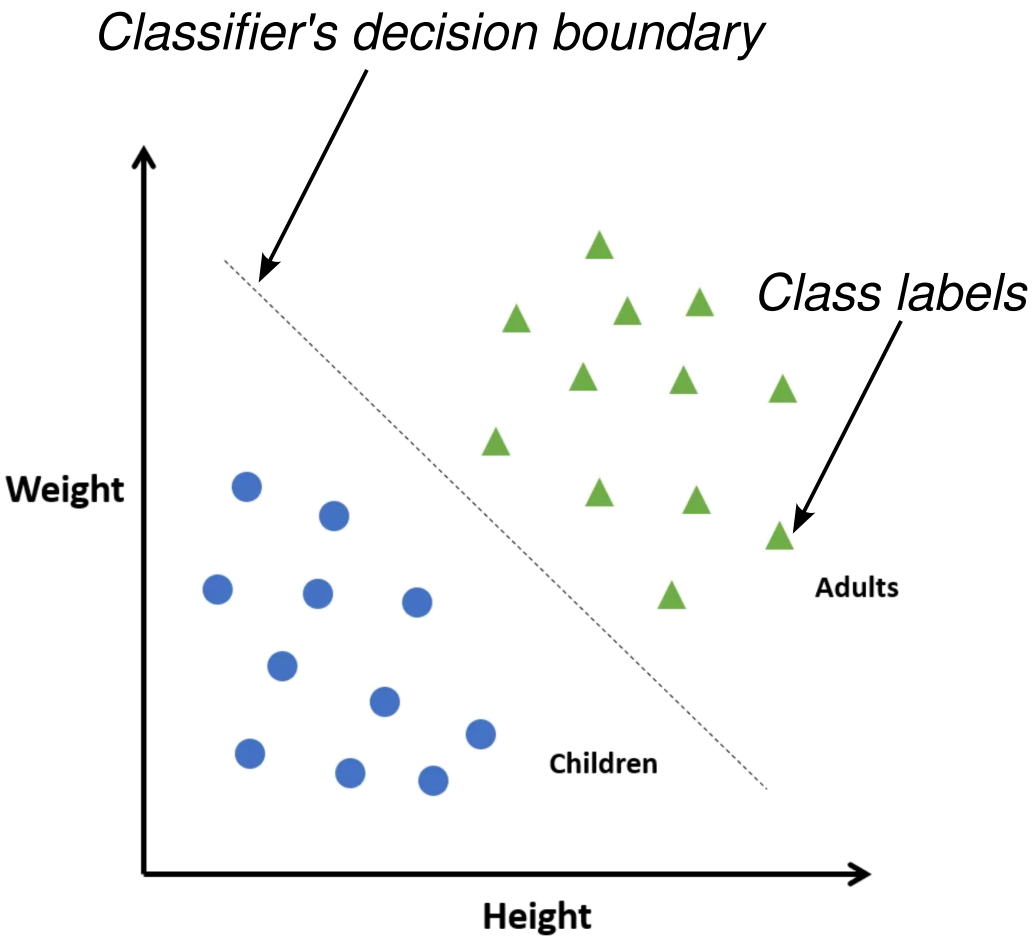
**Optimisation remains crucial!**

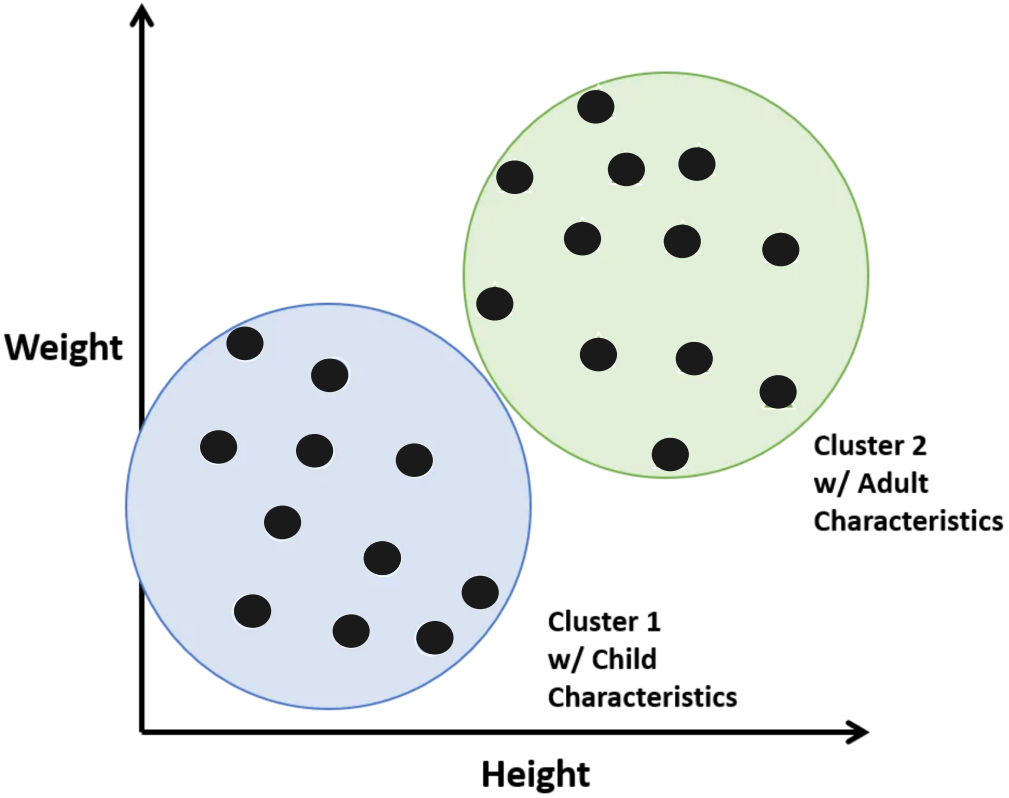
# Classification vs clustering




If class labels are given,
one can train a classifier

# Classification vs clustering

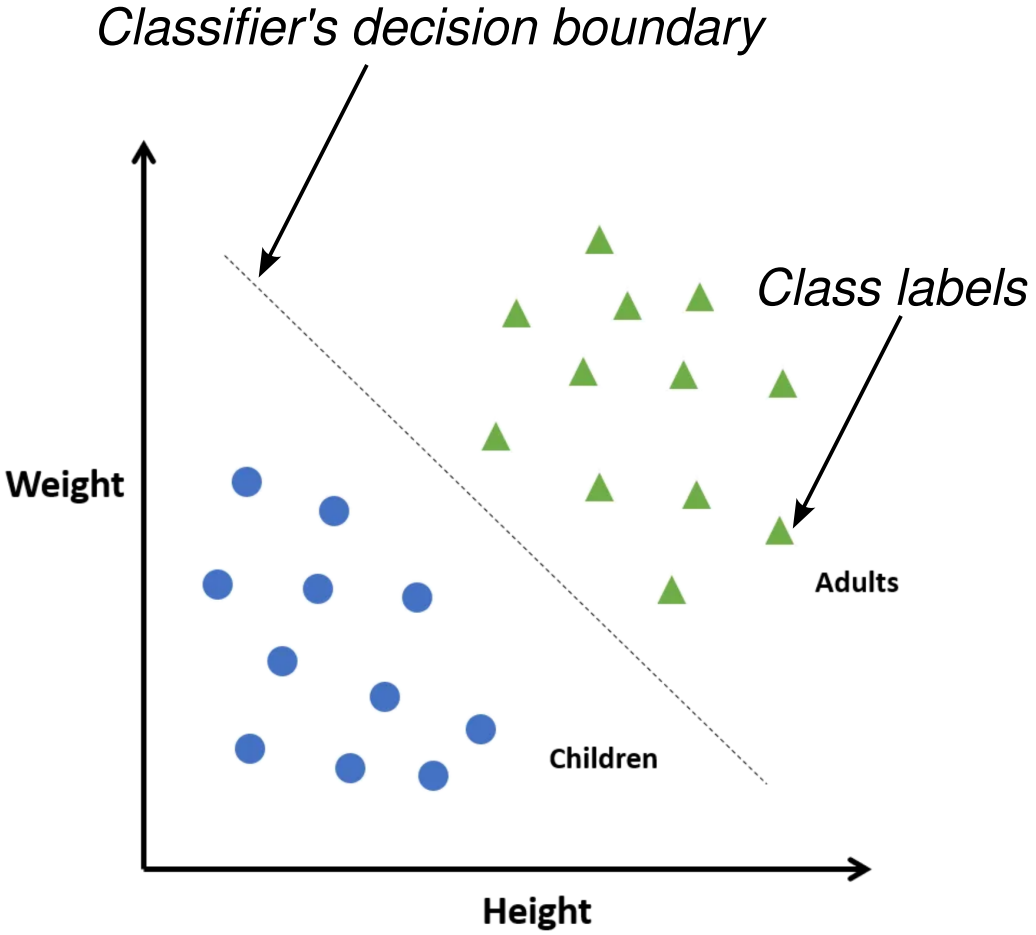


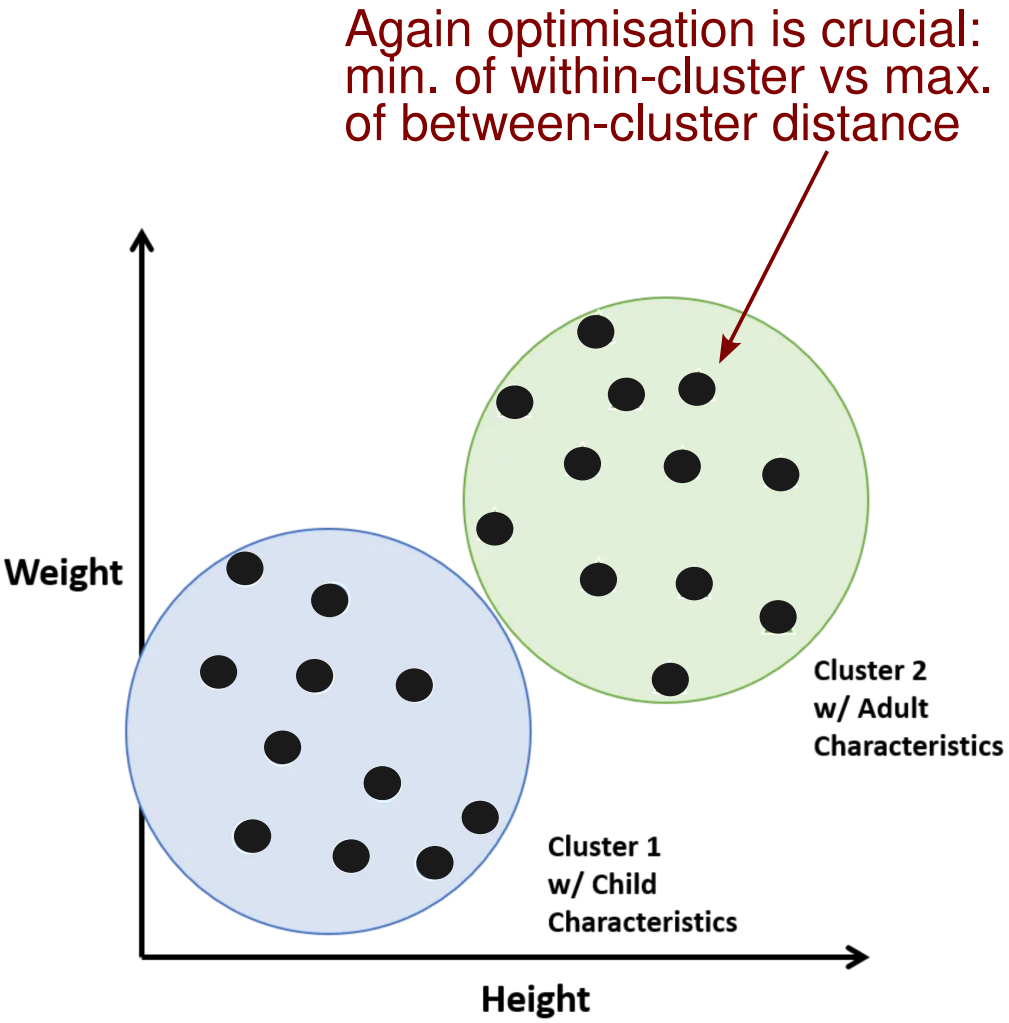




If class labels are given, one can train a classifier

If **no** class labels are given, similarities in data features allow us to discover groups

# Classification vs clustering




If class labels are given, one can train a classifier

If **no** class labels are given, similarities in data features allow us to discover groups

# Clustering:

**Hard clustering** = each data point is assigned to a single cluster

**Soft clustering** = each data point is assigned to different clusters with a certain probability

# Clustering:

**Hard clustering** = each data point is assigned to a single cluster

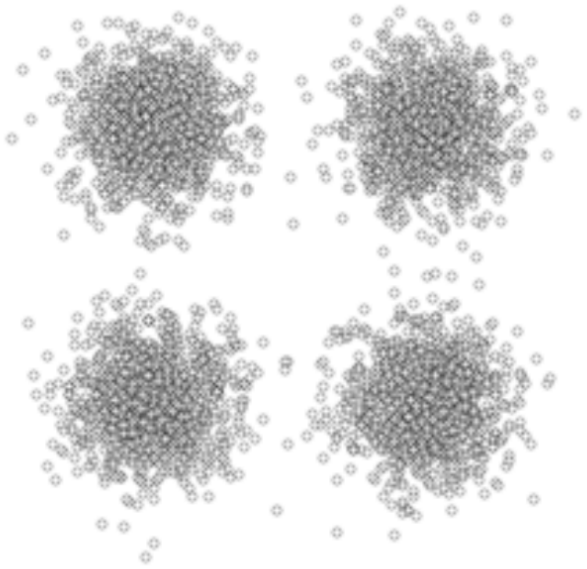
*K*-means & Hierarchical Clustering

**Soft clustering** = each data point is assigned to different clusters with a certain probability

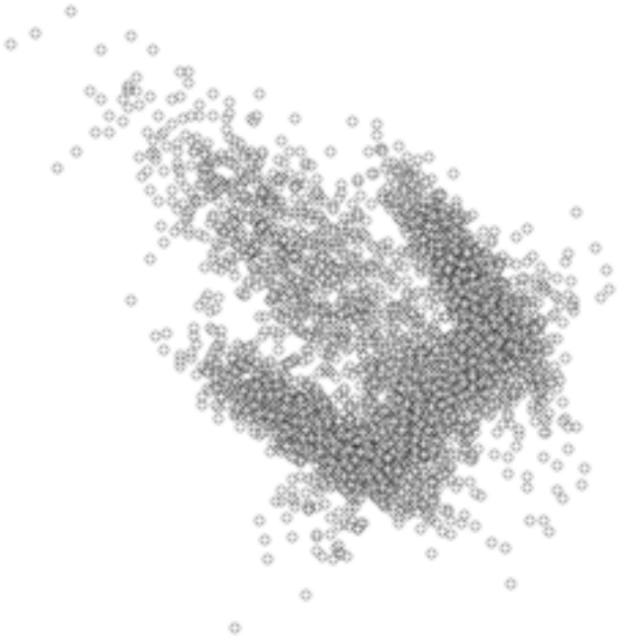
Gaussian Mixture Models

# What is a *cluster*?

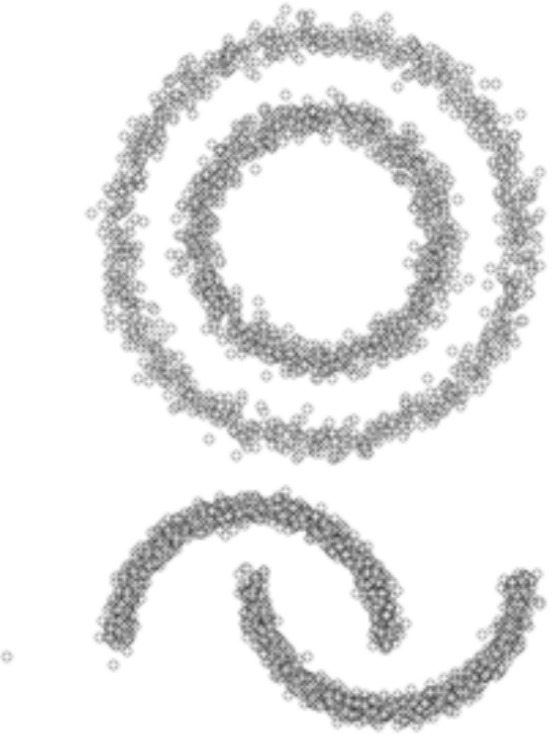
Dataset 1



Dataset 2

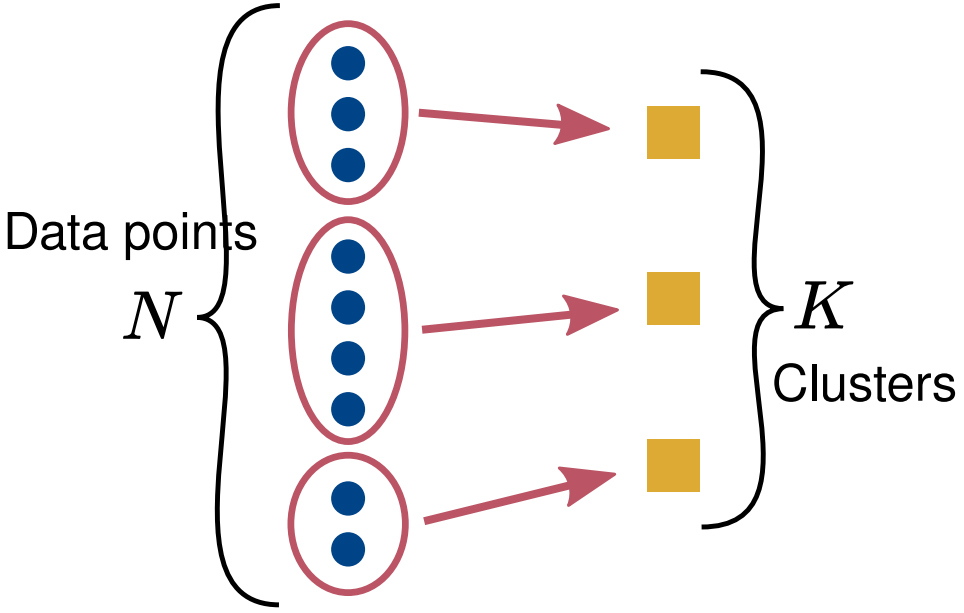


Dataset 3

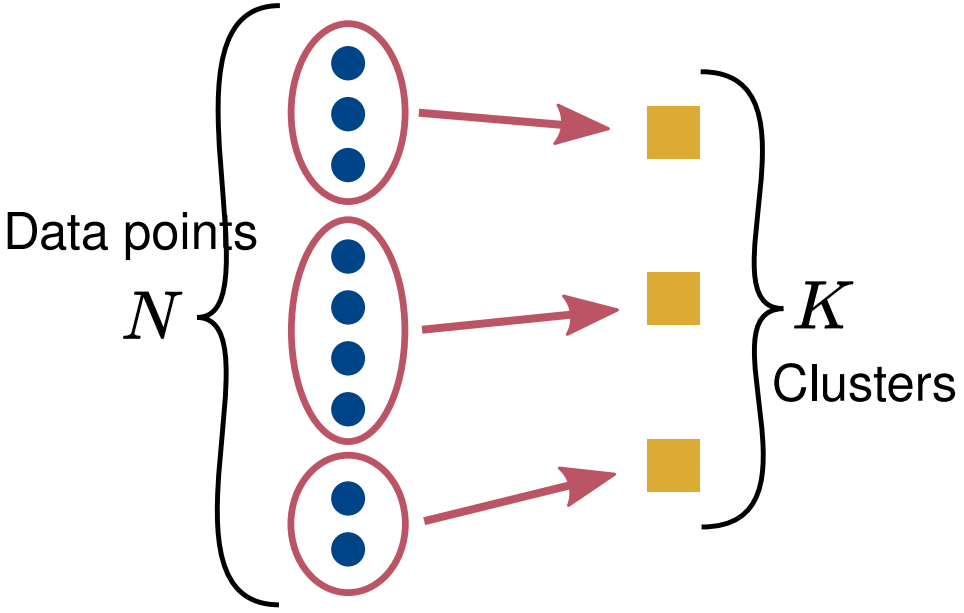


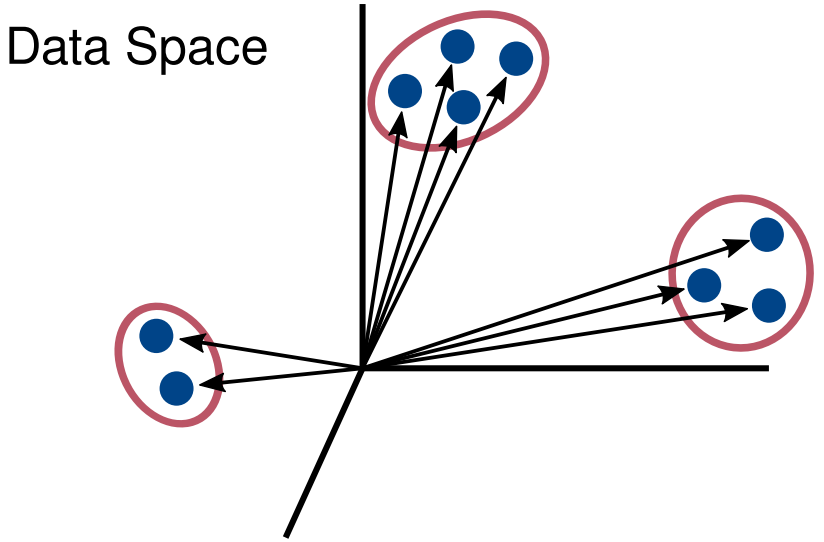
We require a notion of **distance** or **distribution**

# The clustering problem, mathematically



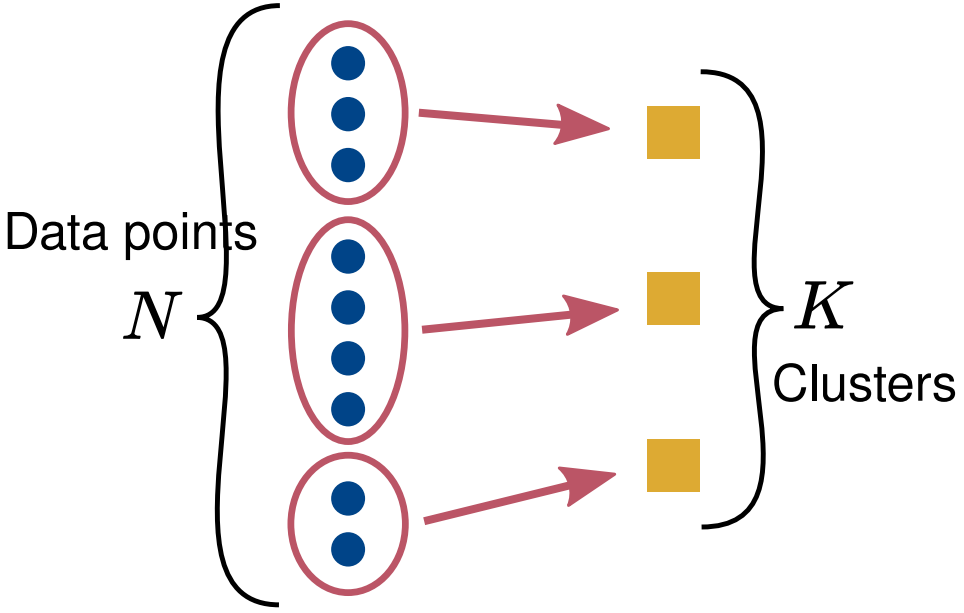
# The clustering problem, mathematically

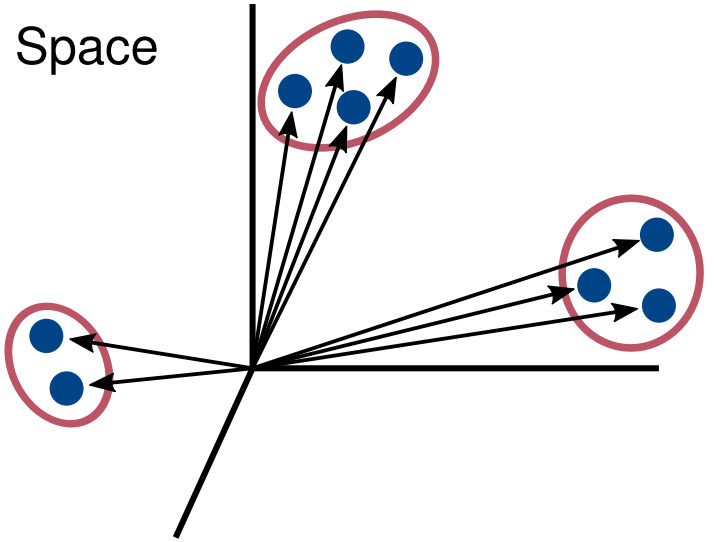



**Desiderata:** low distance within-cluster
high distance between-cluster

# The clustering problem, mathematically




**Desiderata:** low distance within-cluster
high distance between-cluster

Key notion 1: **Distance**, for example:

**squared Euclidean dist.** (continuous)

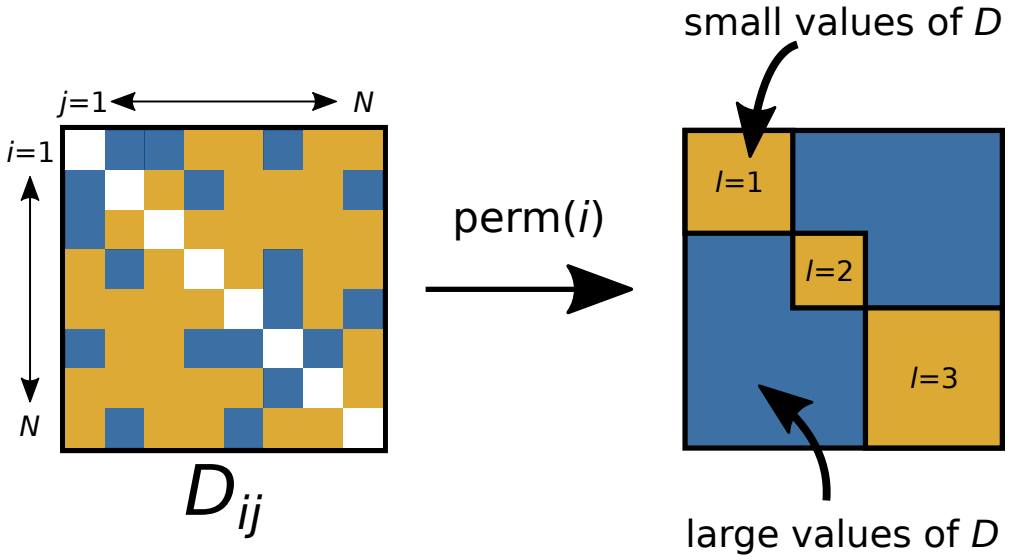
$$D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2$$

**Hamming distance** (discrete)

$$D_H(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_{m=1}^{p} I(x_m^{(i)} \neq x_m^{(j)})$$

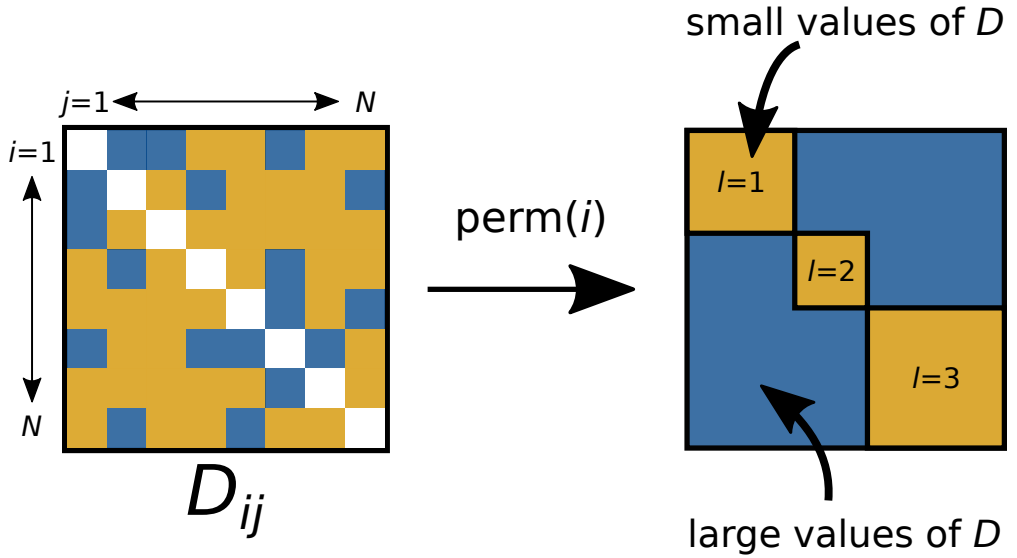
# The clustering problem, mathematically

In other words, clustering = producing a rearrangement of the matrix of distances that is **maximally block-diagonal** (1 block = 1 cluster)



# The clustering problem, mathematically

In other words, clustering = producing a rearrangement of the matrix of distances that is **maximally block-diagonal** (1 block = 1 cluster)




We look for a *i,j* permutation that solves a *combinatorial optimisation*:

$$W(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\in c_k} D(\mathbf{x}^{(i)},\mathbf{x}^{(j)})$$

minimise within-cluster distance,

$$B(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in c_k}\sum_{j\notin c_k} D(\mathbf{x}^{(i)},\mathbf{x}^{(j)})$$

or maximise between-cluster one.

# *K*-means: hard, flat clustering

Continuous data, squared Euclidean distance $D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2$

Find the clustering $\mathcal{C} = \{c_k\}_{k=1}^{K}$ minimising the normalised within-cluster distance:

$$W(\mathcal{C}) = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{|c_k|} \sum_{i,j \in c_k} \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2$$

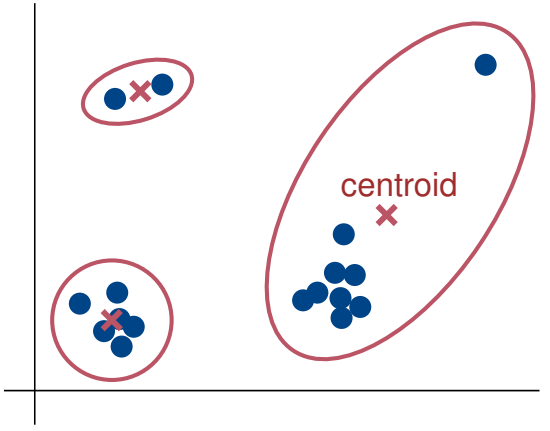
# *K*-means: hard, flat clustering

Continuous data, squared Euclidean distance $D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2$

Find the clustering $\mathcal{C} = \{c_k\}_{k=1}^K$ minimising the normalised within-cluster distance:

$$W(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\frac{1}{|c_k|}\sum_{i,j\in c_k}\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 = \sum_{k=1}^{}\sum_{i\in c_k}\|\mathbf{x}^{(i)} - \mathbf{m}_k\|^2$$

Using the centroid of a cluster



# *K*-means: hard, flat clustering

Continuous data, squared Euclidean distance $D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2$

Find the clustering $\mathcal{C} = \{c_k\}_{k=1}^K$ minimising the normalised within-cluster distance:

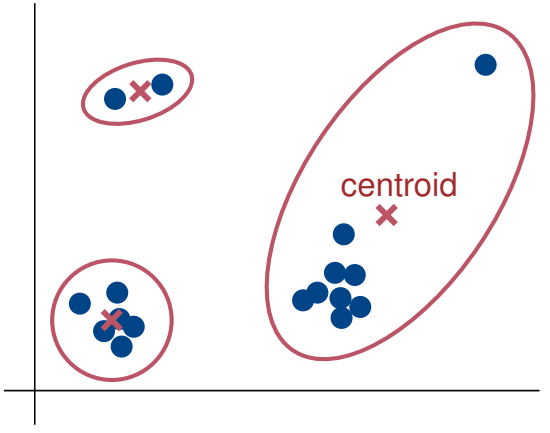
$$W(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\frac{1}{|c_k|}\sum_{i,j\in c_k}\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 = \sum_{k=1}^{}\sum_{i\in c_k}\|\mathbf{x}^{(i)} - \mathbf{m}_k\|^2$$

Using the centroid of a cluster

*K*-means algorithm:

Iterative algorithm that leads to a decreasing $W(\mathcal{C})$

Guarantees convergence only to a **local minimum**
(repeat for a few initialisations and take the best)

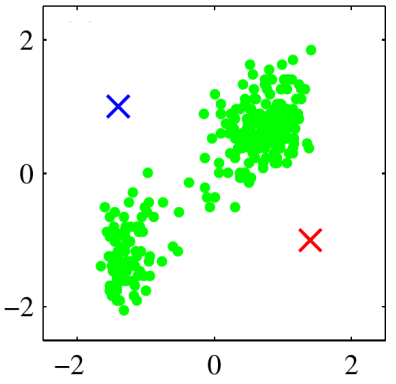


# Running the *K*-means algorithm:

**Step 0:** Given a number of clusters $K$, assign every sample to one of the $K$ clusters at random.

**Step 1:** Compute the *centroid* of each of the $K$ clusters:

$$\mathbf{m}_k = \frac{1}{|c_k|} \sum_{i \in c_k} \mathbf{x}^{(i)}, \quad k = 1, \ldots, K$$



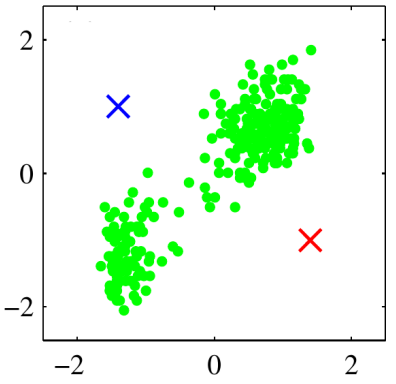

Figures adapted from Bishop, *Pattern Recognition and Machine Learning,* chap. 9

# Running the *K*-means algorithm:

**Step 0:** Given a number of clusters $K$, assign every sample to one of the $K$ clusters at random.

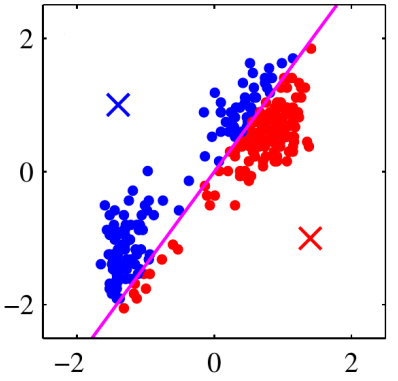
**Step 1:** Compute the *centroid* of each of the $K$ clusters:

$$\mathbf{m}_k = \frac{1}{|c_k|} \sum_{i \in c_k} \mathbf{x}^{(i)}, \quad k = 1, \ldots, K$$




**Step 2:** Reassign each $\mathbf{x}^{(i)}$ to the closest centroid. Mathematically speaking, we consider the cluster assigned to $\mathbf{x}^{(i)}$ at iteration $t$, which is denoted by $k_i^{(t)}$, and evaluate the following:

$$k_i^{(t+1)} = \underset{k}{\operatorname{argmin}} \|\mathbf{x}^{(i)} - \mathbf{m}_k\|^2$$



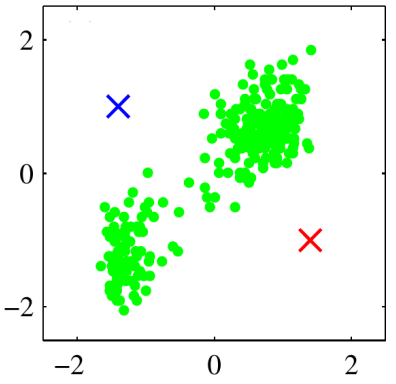

Figures adapted from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

# Running the *K*-means algorithm:

**Step 0:** Given a number of clusters $K$, assign every sample to one of the $K$ clusters at random.

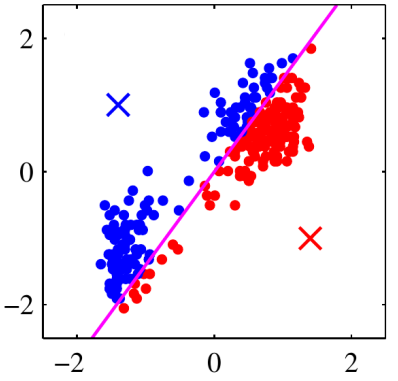
**Step 1:** Compute the *centroid* of each of the $K$ clusters:

$$\mathbf{m}_k = \frac{1}{|c_k|} \sum_{i \in c_k} \mathbf{x}^{(i)}, \quad k = 1, \ldots, K$$

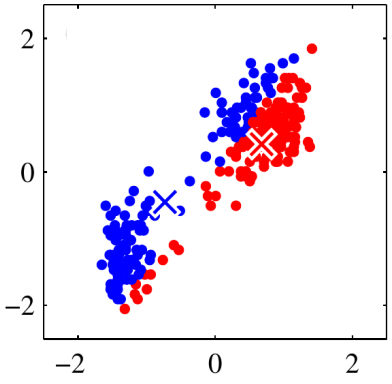


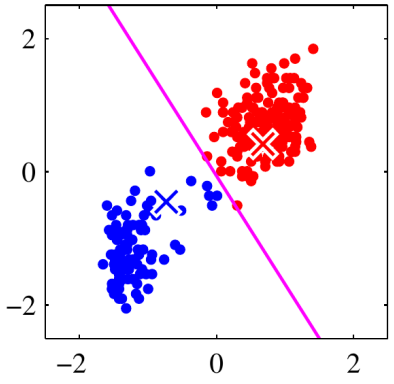

**Step 2:** Reassign each $\mathbf{x}^{(i)}$ to the closest centroid. Mathematically speaking, we consider the cluster assigned to $\mathbf{x}^{(i)}$ at iteration $t$, which is denoted by $k_i^{(t)}$, and evaluate the following:

$$k_i^{(t+1)} = \underset{k}{\operatorname{argmin}} \|\mathbf{x}^{(i)} - \mathbf{m}_k\|^2$$




**Iterate** steps 1 and 2 until convergence, i.e. $W(\mathcal{C})$ does not improve much (see below) or the assignments do not change.



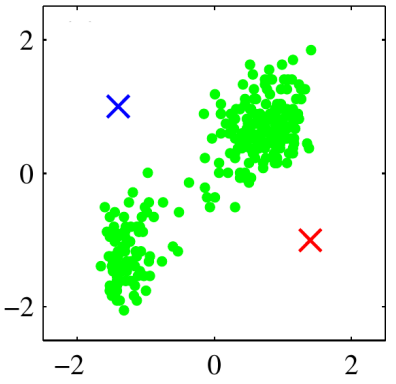

Figures adapted from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

# Running the *K*-means algorithm:

**Step 0:** Given a number of clusters $K$, assign every sample to one of the $K$ clusters at random.

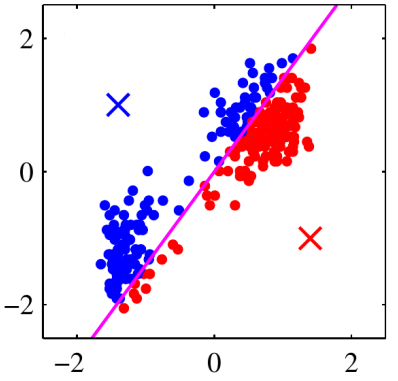
**Step 1:** Compute the *centroid* of each of the $K$ clusters:

$$\mathbf{m}_k = \frac{1}{|c_k|} \sum_{i \in c_k} \mathbf{x}^{(i)}, \quad k = 1, \ldots, K$$



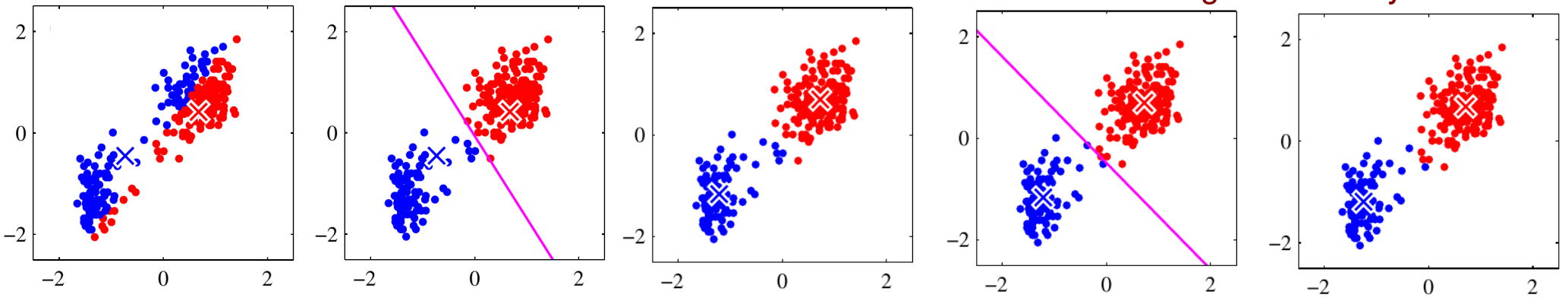

**Step 2:** Reassign each $\mathbf{x}^{(i)}$ to the closest centroid. Mathematically speaking, we consider the cluster assigned to $\mathbf{x}^{(i)}$ at iteration $t$, which is denoted by $k_i^{(t)}$, and evaluate the following:

$$k_i^{(t+1)} = \underset{k}{\operatorname{argmin}} \|\mathbf{x}^{(i)} - \mathbf{m}_k\|^2$$




**Iterate** steps 1 and 2 until convergence, i.e. $W(\mathcal{C})$ does not improve much (see below) or the assignments do not change.

Cluster Assignments stay stable




Figures adapted from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

# Example: segment and compress images


**Image segmentation:** breaking up the image into meaningful or perceptually similar regions

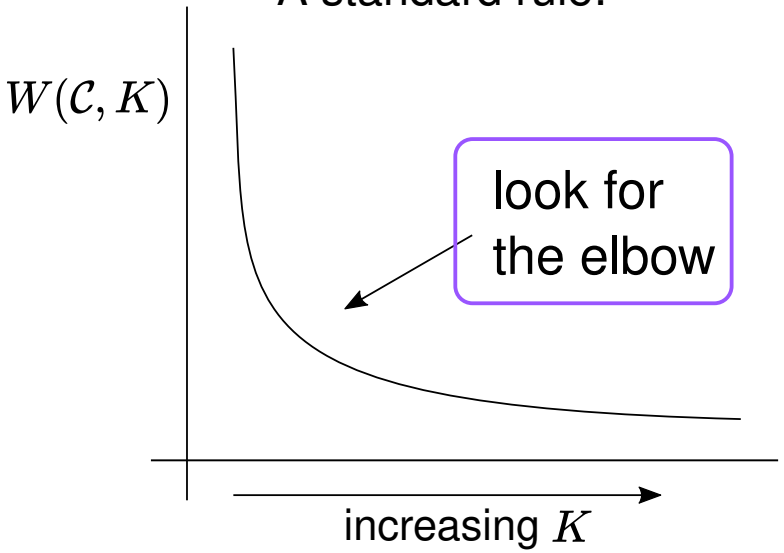
**(Lossy) data compression:** for each pixel, we store only its cluster and the cluster center

Figure from Bishop, *Pattern Recognition and Machine Learning,* chap. 9

# *K*-means: a few considerations

**1. Number of clusters K? Key, but hard choice.**

A standard rule:

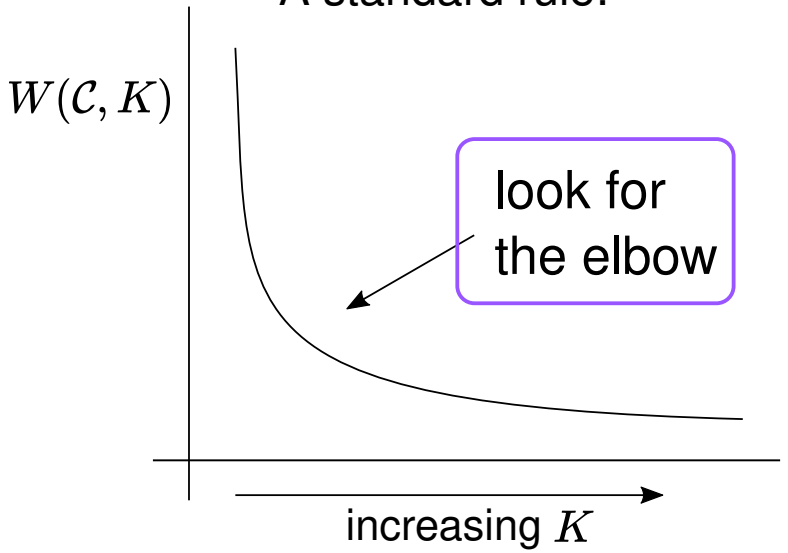



Take *K* after which no significant improvement

# *K*-means: a few considerations

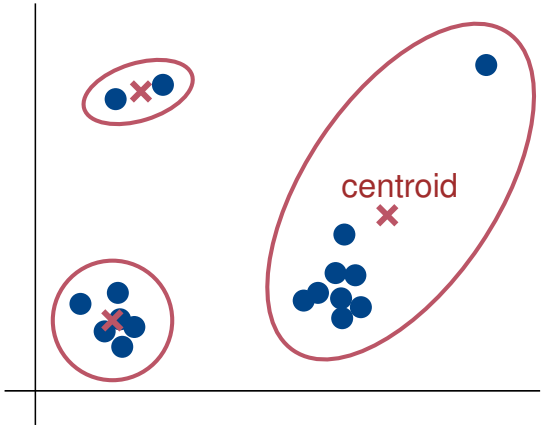
## 1. Number of clusters K? Key, but hard choice.

A standard rule:




Take *K* after which no significant improvement

## 2. Sensitivity to outliers:

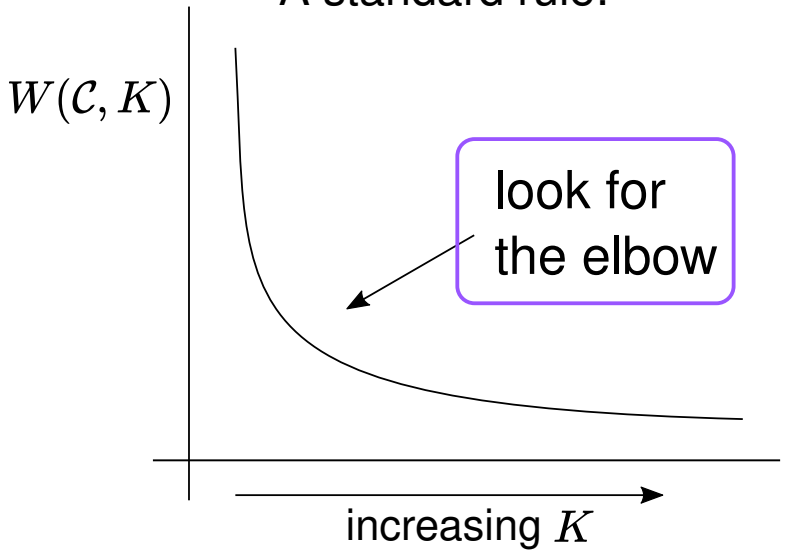



Alternative: *K*-medoid, where the centroid is a data point

# *K*-means: a few considerations

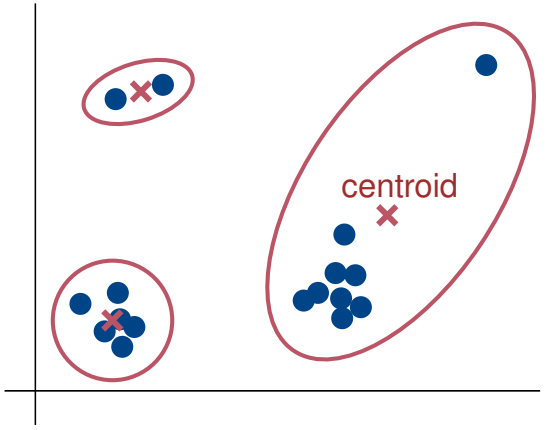
**1. Number of clusters K? Key, but hard choice.**

A standard rule:




Take *K* after which no significant improvement

**2. Sensitivity to outliers:**



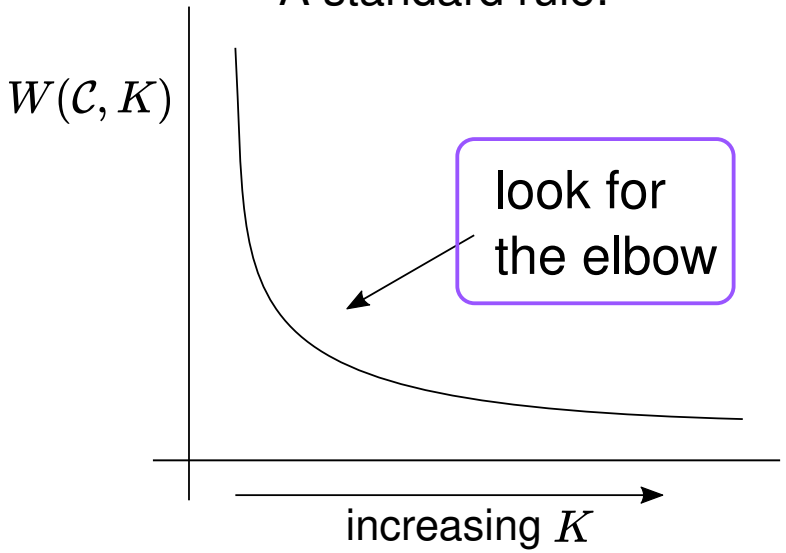

Alternative: *K*-medoid, where the centroid is a data point

**3. Ignores differences in variance:** clusters of non-uniform size?
**Ignores structure of covariance:** elliptical clusters?

# *K*-means: a few considerations

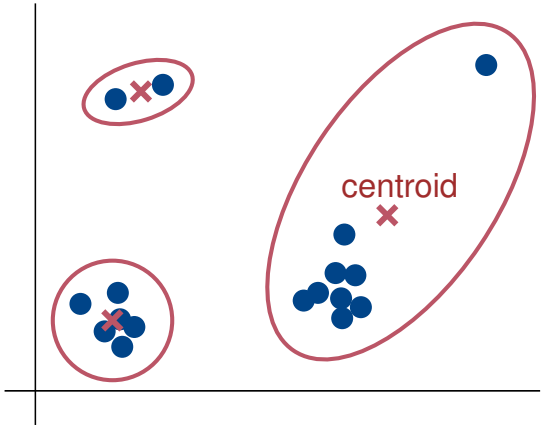
**1. Number of clusters K? Key, but hard choice.**

A standard rule:




Take *K* after which no significant improvement

**2. Sensitivity to outliers:**




Alternative: *K*-medoid, where the centroid is a data point

**3. Ignores differences in variance:** clusters of non-uniform size?
**Ignores structure of covariance:** elliptical clusters?

**4. Hard clustering:** ambigous samples in borderline regions?

# Probabilistic clustering

Each point is assigned to a cluster with a certain *probability* (soft).

# Probabilistic clustering

Each point is assigned to a cluster with a certain *probability* (soft).

These methods build upon the **Key notion 2: distribution** (to learn). For clustering, it is convenient to learn **Mixture Models:**

$$P(\boldsymbol{x} = \mathbf{x}^{(i)}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}^{(i)}|\boldsymbol{\theta})$$

Data are modelled by a mixture of distributions (one per cluster)

Generative model = sample new data

# Probabilistic clustering

Each point is assigned to a cluster with a certain *probability* (soft).

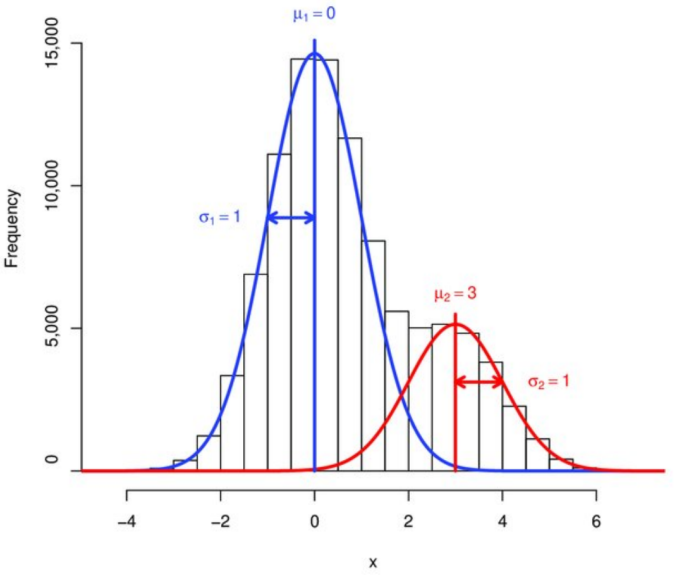
These methods build upon the **Key notion 2: distribution** (to learn). For clustering, it is convenient to learn **Mixture Models:**

$$P(\boldsymbol{x} = \mathbf{x}^{(i)}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}^{(i)}|\boldsymbol{\theta})$$

Data are modelled by a mixture of distributions (one per cluster)

Generative model = sample new data

**Here: Gaussian Mixture Model (GMM)**



# GMMs as latent variable models

We introduce a latent variable $z$
to model the cluster assignment,
i.e. taking $K$ discrete values.
We model the joint probability:

$$P(\boldsymbol{x}, z)$$

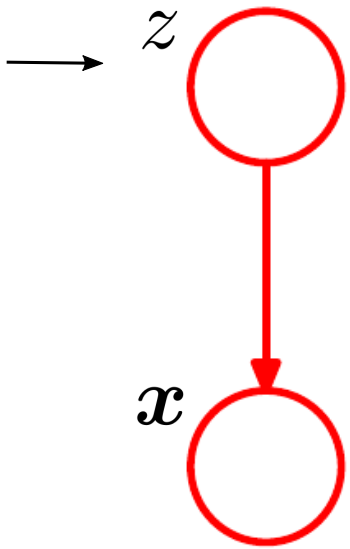
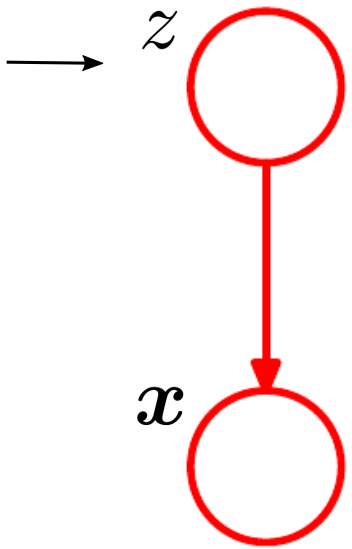



Figure from Bishop, *Pattern Recognition and Machine Learning,* chap. 9

# GMMs as latent variable models

We introduce a latent variable $z$
to model the cluster assignment,
i.e. taking $K$ discrete values.
We model the joint probability:

$$P(\boldsymbol{x}, z)$$




**Mixture components**

Marginal data probability

$$P(\boldsymbol{x} = \mathbf{x}^{(i)}) = \sum_{k=1}^{K} P(z = k) P(\boldsymbol{x} = \mathbf{x}^{(i)} | z = k) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}^{(i)} | \boldsymbol{\theta})$$

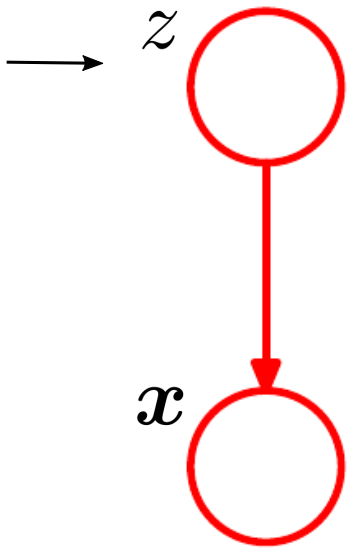
**Mixture weights**

Figure from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

# GMMs as latent variable models

We introduce a latent variable *z*
to model the cluster assignment,
i.e. taking *K* discrete values.
We model the joint probability:

$$P(\boldsymbol{x}, z)$$




**Mixture components**

Marginal data probability

$$P(\boldsymbol{x} = \mathbf{x}^{(i)}) = \sum_{k=1}^{K} P(z = k) P(\boldsymbol{x} = \mathbf{x}^{(i)} | z = k) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}^{(i)} | \boldsymbol{\theta})$$

**Mixture weights**

**The cluster probability:**

$$r_{ik}(\boldsymbol{\theta}) = P(z = k | \boldsymbol{x} = \mathbf{x}^{(i)}, \boldsymbol{\theta}) = \frac{P(z = k, \boldsymbol{x} = \mathbf{x}^{(i)}, \boldsymbol{\theta})}{P(\boldsymbol{x} = \mathbf{x}^{(i)}, \boldsymbol{\theta})} = \frac{\pi_k p_k(\mathbf{x}^{(i)} | \boldsymbol{\theta})}{\sum_{k'=1}^{K} \pi_{k'} p_{k'}(\mathbf{x}^{(i)} | \boldsymbol{\theta})}$$

Probability of belonging to a cluster *k*

Figure from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

# GMMs as latent variable models

**Mixture weights**

To be a valid probability:

$$0 \leq \pi_k \leq 1$$

$$\sum_{k=1}^{K} \pi_k = 1$$

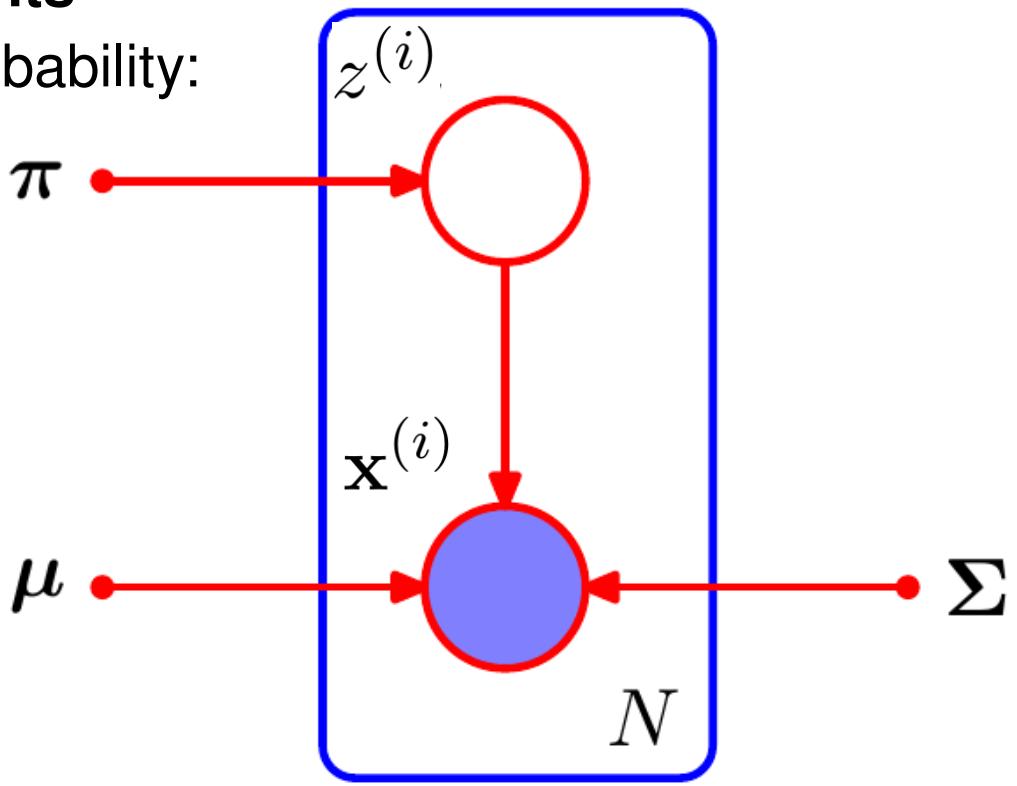



Figure from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

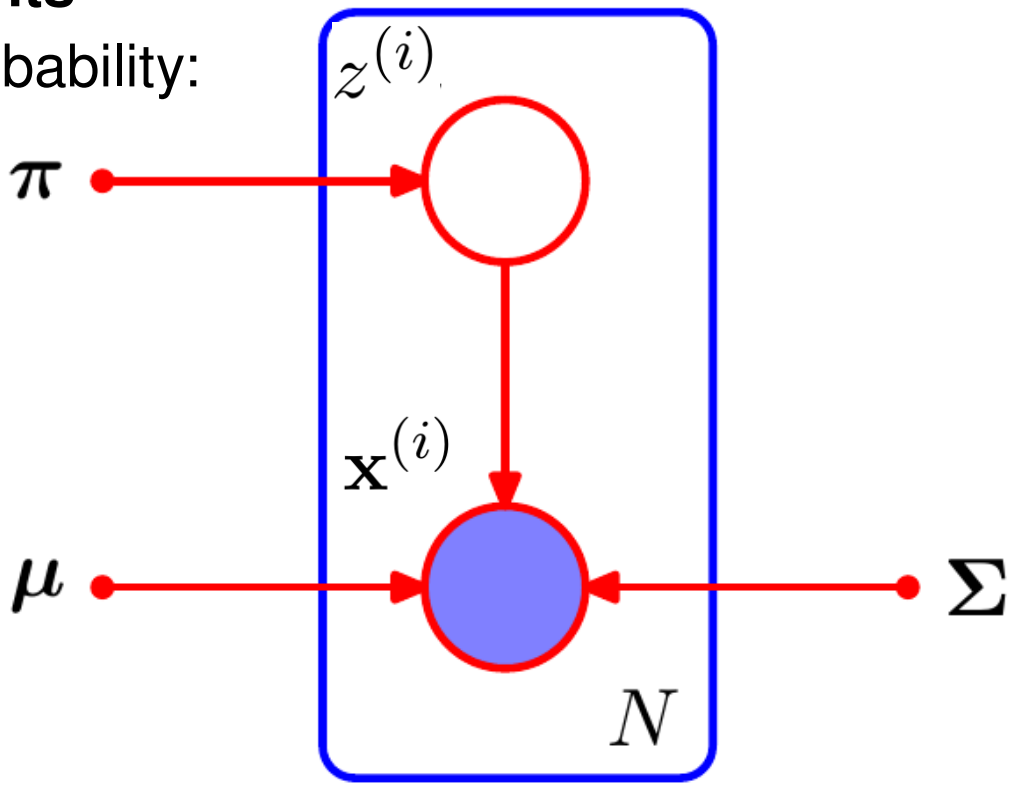
# GMMs as latent variable models

**Mixture weights**

To be a valid probability:

$$0 \leq \pi_k \leq 1$$

$$\sum_{k=1}^{K} \pi_k = 1$$




Each mixture component is a **Gaussian distribution**:

$$p_k(\boldsymbol{x}|\boldsymbol{\theta}) = (2\pi)^{-p/2} \det(\boldsymbol{\Sigma}_k)^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k{}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right)$$

Figure from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

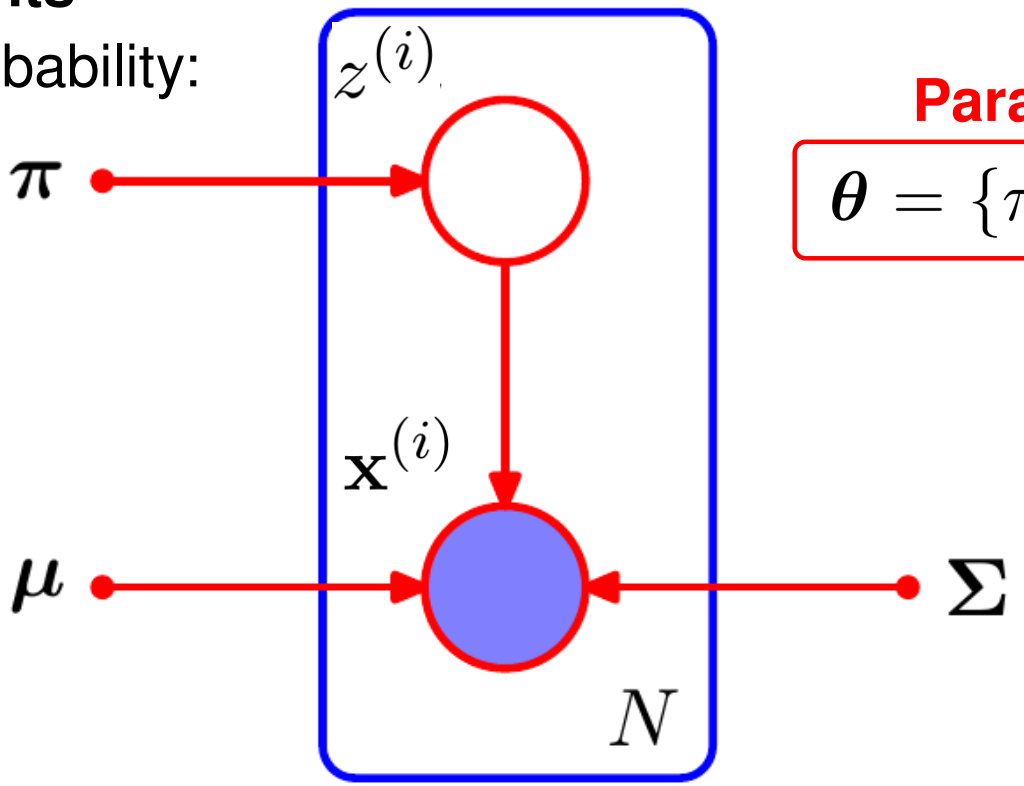
# GMMs as latent variable models

**Mixture weights**

To be a valid probability:

$$0 \leq \pi_k \leq 1$$

$$\sum_{k=1}^{K} \pi_k = 1$$




**Parameters to learn:**

$$\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1,..,K}$$

Each mixture component is a **Gaussian distribution**:

$$p_k(\boldsymbol{x}|\boldsymbol{\theta}) = (2\pi)^{-p/2} \det(\boldsymbol{\Sigma}_k)^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k{}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right)$$

Figure from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

# Expectation Maximisation Algorithm

Since we work with a distribution, we can apply **Maximum Likelihood**:

$$\boldsymbol{\theta}_{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \log \mathcal{L}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{i=1}^{N} \log P(\boldsymbol{x} = \mathbf{x}^{(i)} | \boldsymbol{\theta})$$

$\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1,..,K}$

**Expectation - Maximisation:** iterative algorithm to find a *local* maximum

Initialise $\boldsymbol{\theta}$

For iterations *t = 1,..., T*:

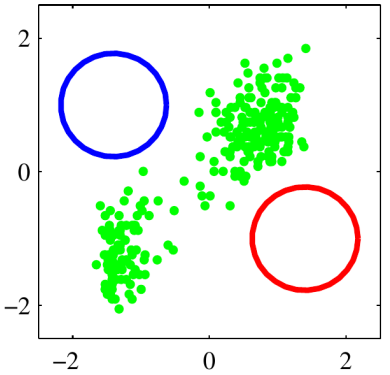
**Expectation Step:** estimate cluster probability

**Maximisation Step:** take the parameters maximising the log-likelihood

until convergence (log-likelihood stops changing)

## The Expectation-Maximisation (EM) algorithm:

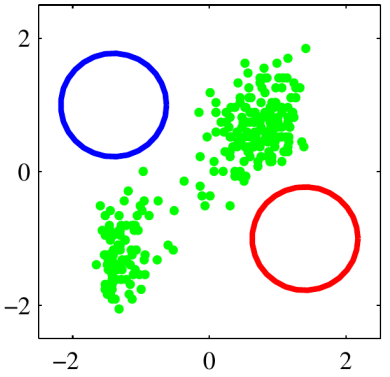
Initialise at random the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1,..,K}$




Figures from Bishop, *Pattern Recognition and Machine Learning,* chap. 9

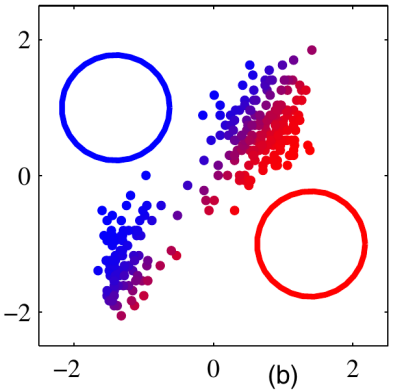
## The Expectation-Maximisation (EM) algorithm:

Initialise at random the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1,..,K}$




**Expectation step (E-step):** uses current parameter values $\boldsymbol{\theta}^t = \{\pi_k^t, \boldsymbol{\mu}_k^t, \boldsymbol{\Sigma}_k^t\}_{k=1,..,K}$ to compute cluster probabilities $r_{ik}(\boldsymbol{\theta}^t)$ and a set of weights:

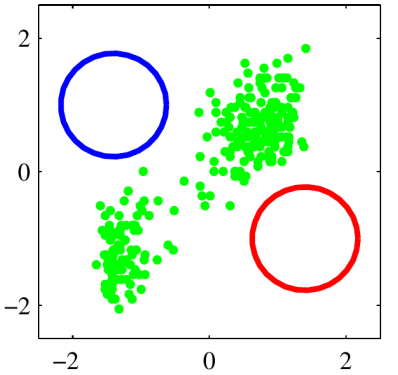
$$w_{ik}(\boldsymbol{\theta}^t) = \frac{r_{ik}(\boldsymbol{\theta}^t)}{\sum_{i'=1}^{N} r_{i'k}(\boldsymbol{\theta}^t)} \tag{9.18}$$




Figures from Bishop, *Pattern Recognition and Machine Learning,* chap. 9

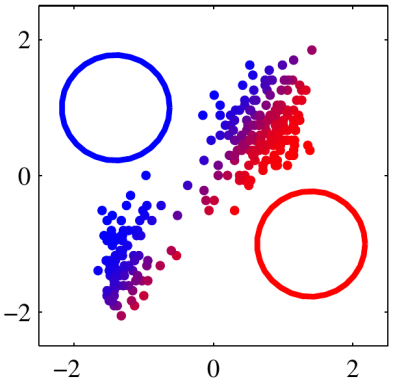
## The Expectation-Maximisation (EM) algorithm:

Initialise at random the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1,..,K}$




**Expectation step (E-step):** uses current parameter values $\boldsymbol{\theta}^t = \{\pi_k^t, \boldsymbol{\mu}_k^t, \boldsymbol{\Sigma}_k^t\}_{k=1,..,K}$ to compute cluster probabilities $r_{ik}(\boldsymbol{\theta}^t)$ and a set of weights:

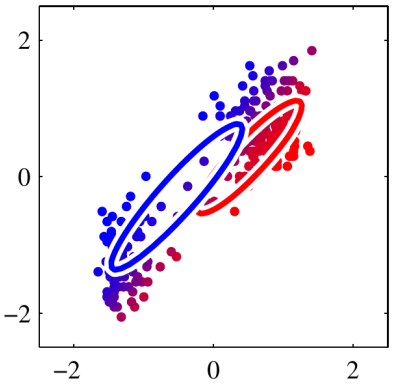
$$w_{ik}(\boldsymbol{\theta}^t) = \frac{r_{ik}(\boldsymbol{\theta}^t)}{\sum_{i'=1}^{N} r_{i'k}(\boldsymbol{\theta}^t)} \tag{9.18}$$




**Maximisation step (M-step):** updates the model's parameters for the next iteration $t+1$ maximising of the log-likelihood. One updates the mixture weight for the $k$-th component via the corresponding data-averaged cluster probability:

$$\pi_k^{t+1} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{z^{(i)}|\mathbf{x}^{(i)},\boldsymbol{\theta}^t}[\delta_{z^{(i)},k}] = \frac{1}{N}\sum_{i=1}^{N} r_{ik}(\boldsymbol{\theta}^t) = \frac{N_k}{N} \tag{9.19}$$

Mixture weights




One updates the parameters of the Gaussian mixture components using $w_{ik}(\boldsymbol{\theta}^t)$:

$$\boldsymbol{\mu}_k^{t+1} = \sum_{i=1}^{N} w_{ik}(\boldsymbol{\theta}^t)\mathbf{x}^{(i)} \tag{9.20}$$

Gaussian means

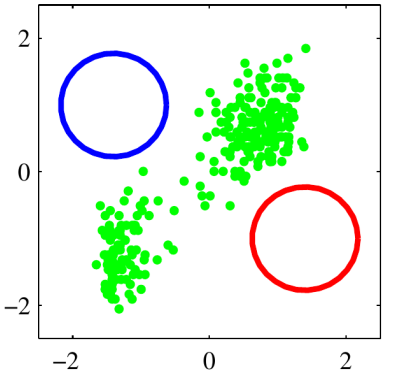
$$\boldsymbol{\Sigma}_k^{t+1} = \sum_{i=1}^{N} w_{ik}(\boldsymbol{\theta}^t)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{t+1})(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{t+1})^T \tag{9.21}$$

Gaussian covariances

Figures from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

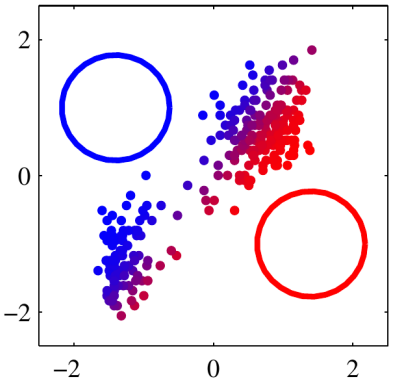
## The Expectation-Maximisation (EM) algorithm:

Initialise at random the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1,..,K}$



**Expectation step (E-step):** uses current parameter values $\boldsymbol{\theta}^t = \{\pi_k^t, \boldsymbol{\mu}_k^t, \boldsymbol{\Sigma}_k^t\}_{k=1,..,K}$ to compute cluster probabilities $r_{ik}(\boldsymbol{\theta}^t)$ and a set of weights:

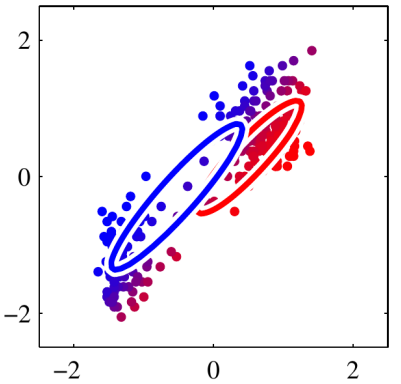
$$w_{ik}(\boldsymbol{\theta}^t) = \frac{r_{ik}(\boldsymbol{\theta}^t)}{\sum_{i'=1}^{N} r_{i'k}(\boldsymbol{\theta}^t)} \tag{9.18}$$



**Maximisation step (M-step):** updates the model's parameters for the next iteration $t+1$ maximising of the log-likelihood. One updates the mixture weight for the $k$-th component via the corresponding data-averaged cluster probability:

$$\pi_k^{t+1} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{z^{(i)}|\mathbf{x}^{(i)},\boldsymbol{\theta}^t}[\delta_{z^{(i)},k}] = \frac{1}{N}\sum_{i=1}^{N} r_{ik}(\boldsymbol{\theta}^t) = \frac{N_k}{N} \tag{9.19}$$

Mixture weights



One updates the parameters of the Gaussian mixture components using $w_{ik}(\boldsymbol{\theta}^t)$:

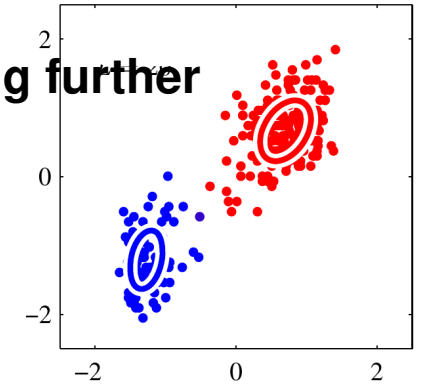
$$\boldsymbol{\mu}_k^{t+1} = \sum_{i=1}^{N} w_{ik}(\boldsymbol{\theta}^t)\mathbf{x}^{(i)} \tag{9.20}$$

Gaussian means

**Iterating further**

$$\boldsymbol{\Sigma}_k^{t+1} = \sum_{i=1}^{N} w_{ik}(\boldsymbol{\theta}^t)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{t+1})(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{t+1})^T \tag{9.21}$$

Gaussian covariances



Figures from Bishop, *Pattern Recognition and Machine Learning*, chap. 9

# GMMs vs *K*-means

**Key differences:**

1. GMM: soft clustering,
   K-means: hard clustering

2. K-means: symmetrical and uniform clusters,
   GMM: ellipsoids to model each cluster

# GMMs vs *K*-means

**Key differences:**

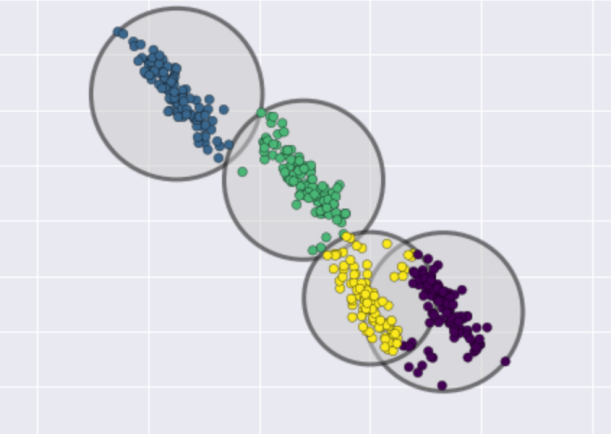
1. GMM: soft clustering,
   K-means: hard clustering

2. K-means: symmetrical and uniform clusters,
   GMM: ellipsoids to model each cluster

**GMM can be seen as a generalisation of *K*-means to soft clustering!**

If we look at the special GMM case:

$$\boldsymbol{\Sigma} = \text{diag}(1) \quad \pi_k = 1/K$$




and if we enforce hard clustering via argmax:

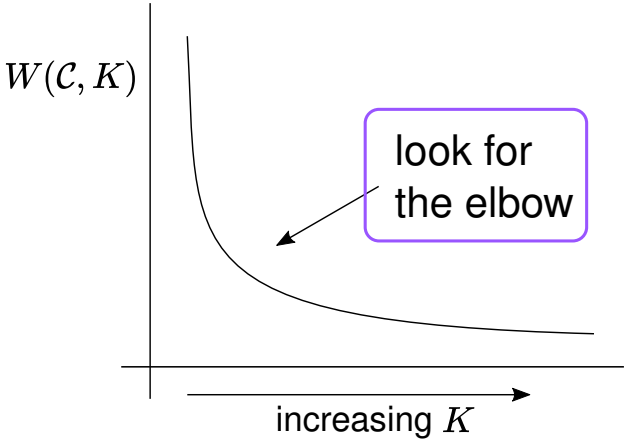
$$r_{ik}(\boldsymbol{\theta}) = \begin{cases} 1 & \text{if } k = \text{argmax}_j \, r_{ij}(\boldsymbol{\theta}) = \text{argmin}_j \, ||\mathbf{x}^{(i)} - \boldsymbol{\mu}_j||^2 \\ 0 & \text{otherwise} \end{cases}$$

We retrieve an assignment based on centroids, i.e., *K*-means!

# GMMs vs *K*-means

**One still needs to choose the number of mixture components *K***

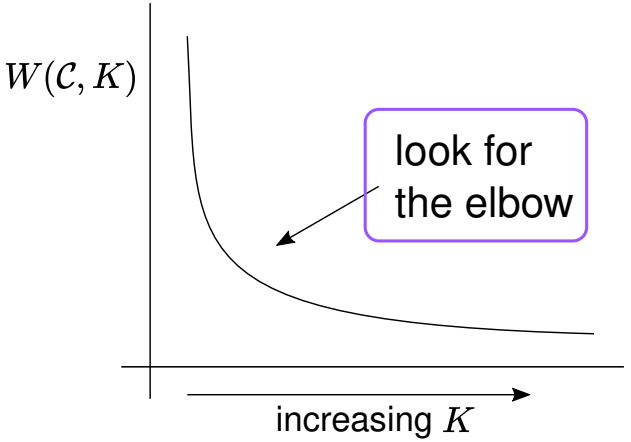
*K*-Means, being distance-based:



# GMMs vs *K*-means

**One still needs to choose the number of mixture components *K***

*K*-Means, being distance-based:




In GMMs (distribution-based) one uses criteria with the log-likelihood:

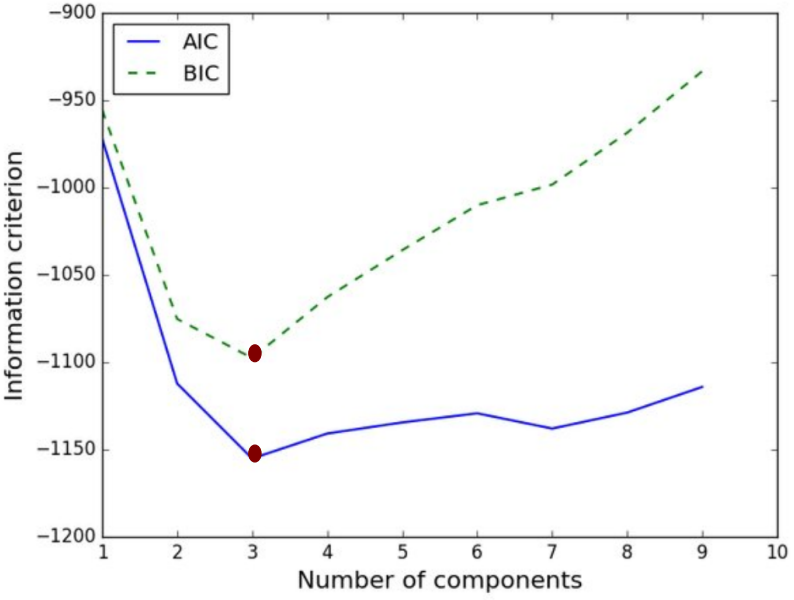
AIC: Akaike Information Criterion

$$AIC = 2N_{\boldsymbol{\theta}} - 2\ln(\mathcal{L}(\boldsymbol{\theta}))$$

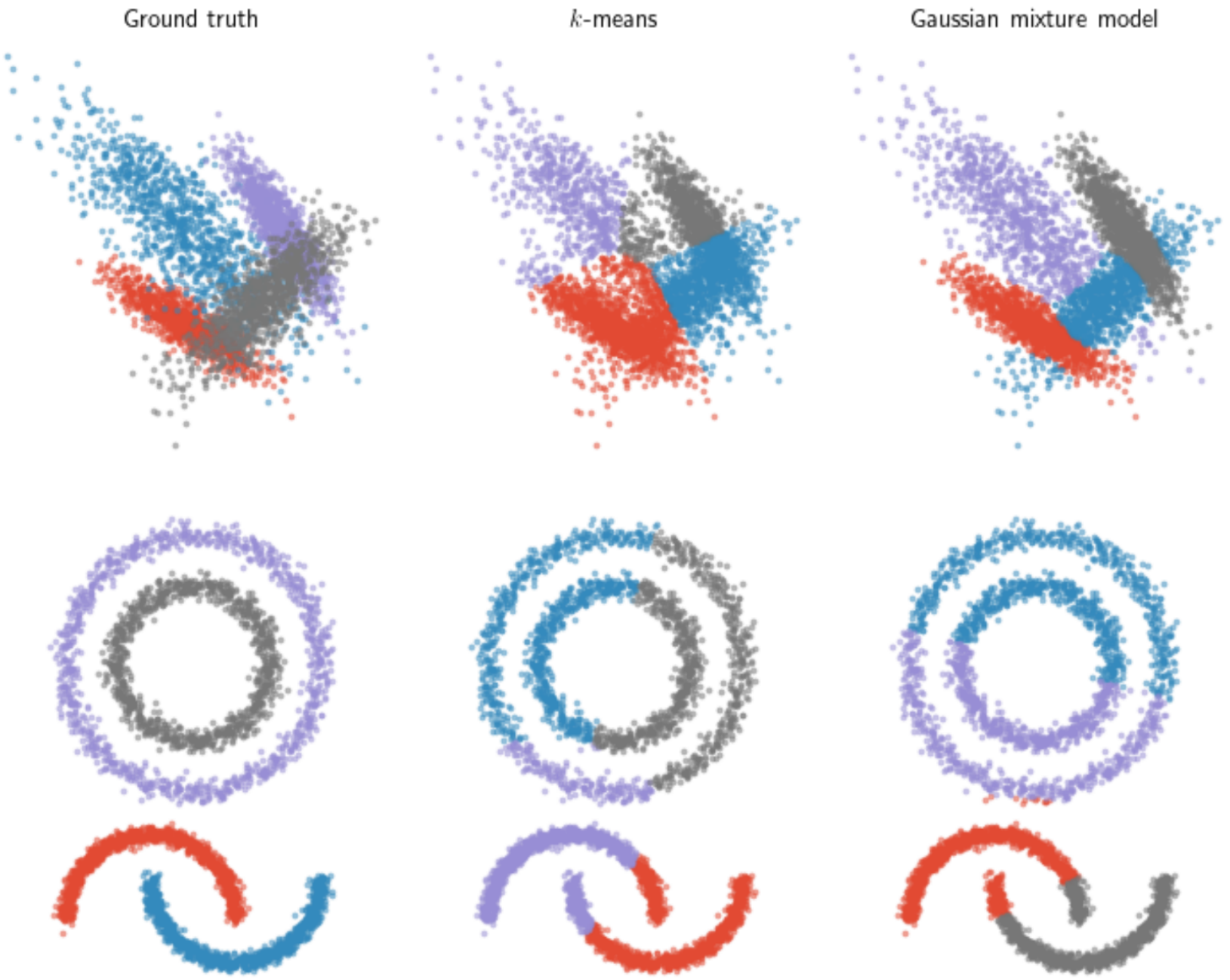
BIC: Bayesian Information Criterion

$$BIC = N_{\boldsymbol{\theta}} lnN - 2\ln(\mathcal{L}(\boldsymbol{\theta}))$$

Balance fitting - model complexity, discouraging overfitting



# GMMs vs *K*-means



# Hierarchical Clustering (HC)

*K*-Means leads to *flat* clustering: i.e., it groups similar objects into a pre-defined number of clusters, without information on the overall data similarity structure.

HC goes beyond this, while still being: for *hard clustering* and *distance-based*

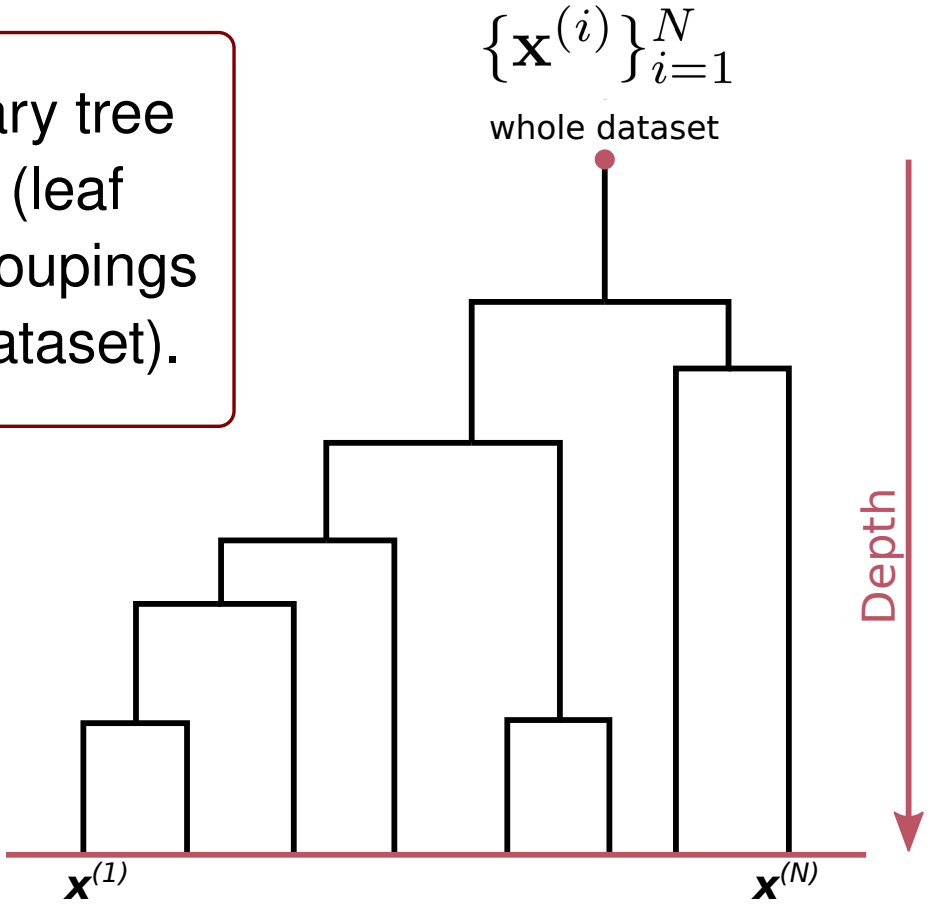
# Hierarchical Clustering (HC)

*K*-Means leads to *flat* clustering: i.e., it groups similar objects into a pre-defined number of clusters, without information on the overall data similarity structure.

HC goes beyond this, while still being: for *hard clustering* and *distance-based*

HC represents whole dataset as a binary tree - **dendrogram** - tracking how samples (leaf nodes) get agglomerated into larger groupings going up to the root node (the whole dataset).

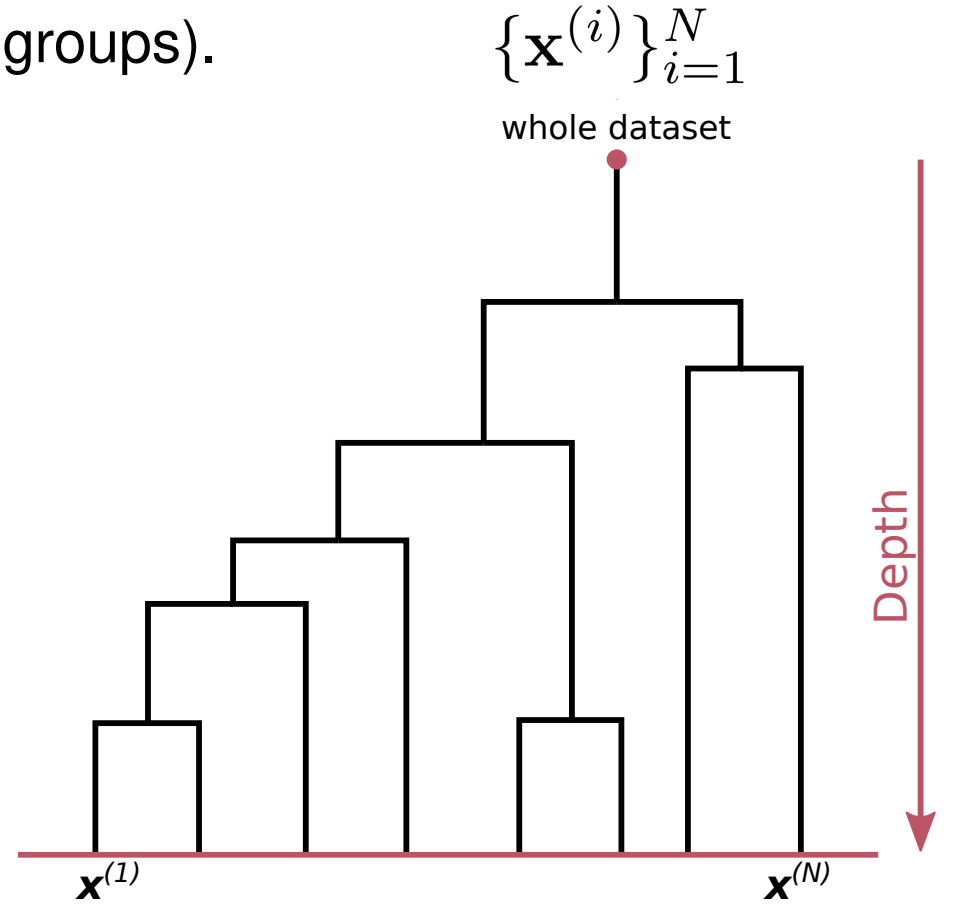
## Dendrogram



# Hierarchical Clustering (HC)

One starts from a matrix of *distances* between samples $D_{ij} := D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

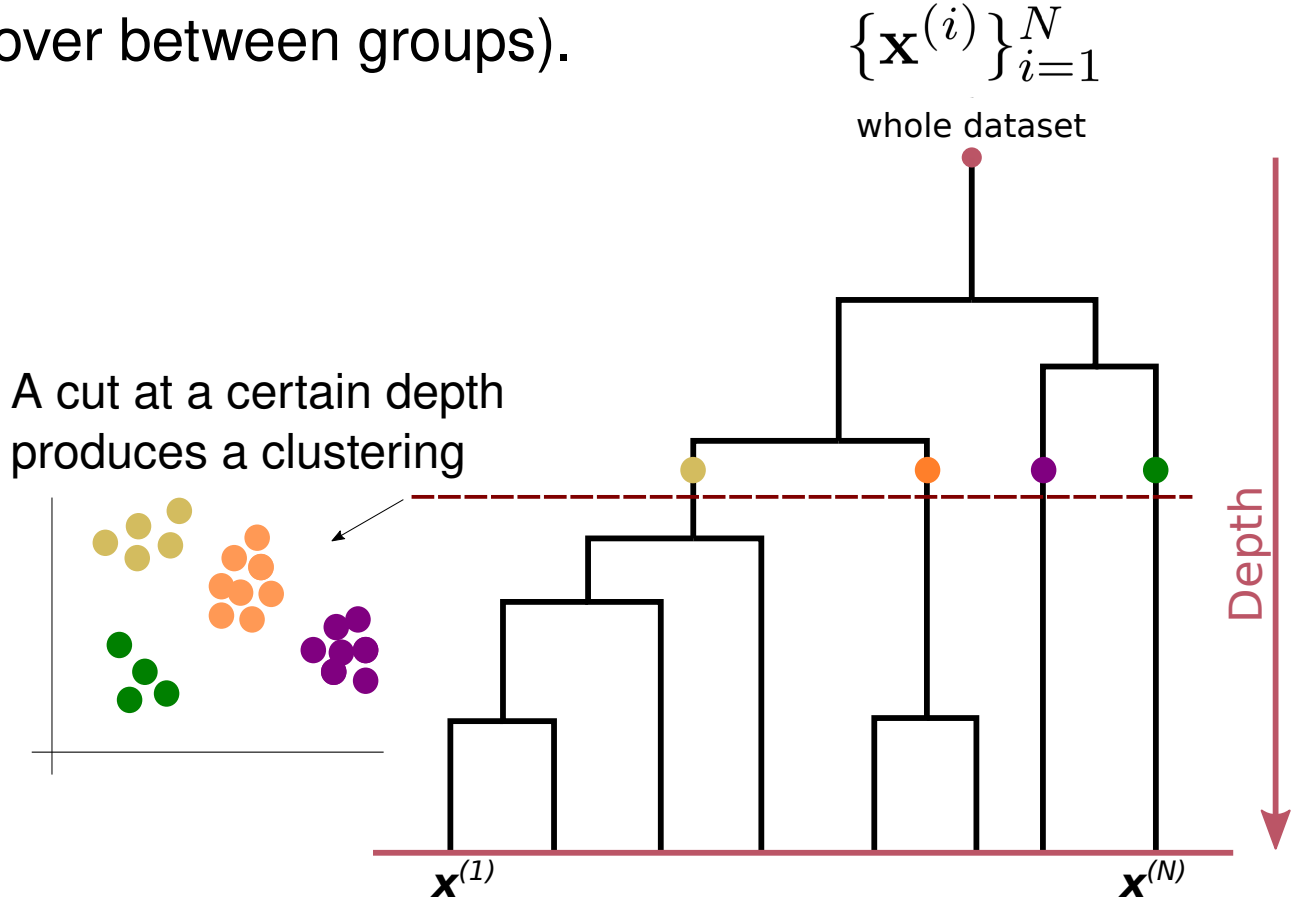
Distances are used for the dendrogram's *repeated merges* of samples' groups:
1 merge = 1 binary split on the dendrogram;
every merge occurs at a specific depth;
strict hierarchy (no cross-over between groups).



# Hierarchical Clustering (HC)

One starts from a matrix of *distances* between samples $D_{ij} := D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

Distances are used for the dendrogram's *repeated merges* of samples' groups:
1 merge = 1 binary split on the dendrogram;
every merge occurs at a specific depth;
strict hierarchy (no cross-over between groups).



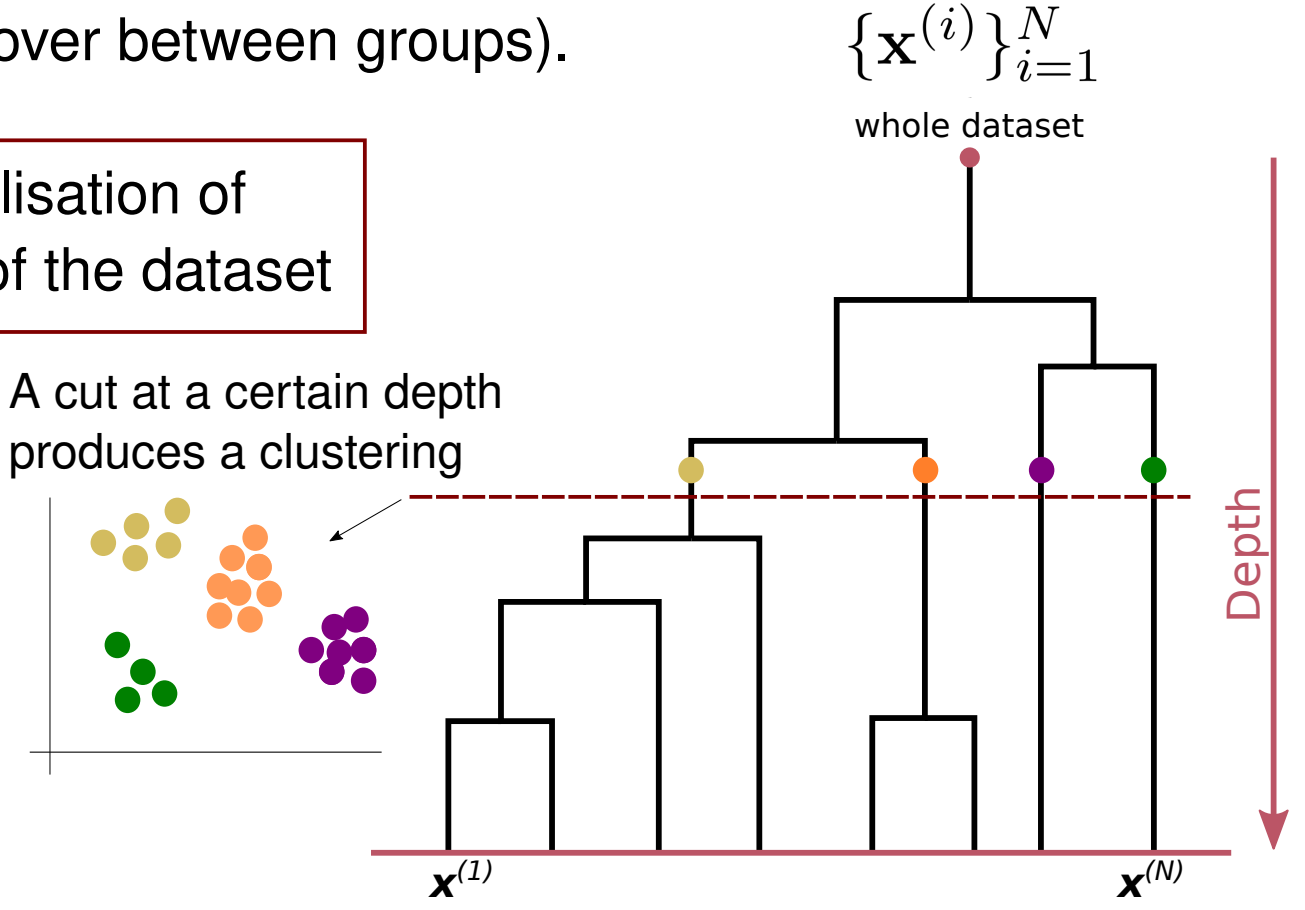

**Result**: a sequence
of clustering assignments

# Hierarchical Clustering (HC)

One starts from a matrix of *distances* between samples $D_{ij} := D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

Distances are used for the dendrogram's *repeated merges* of samples' groups:
1 merge = 1 binary split on the dendrogram;
every merge occurs at a specific depth;
strict hierarchy (no cross-over between groups).

Dendrogram: global visualisation of the *intrinsic organisation* of the dataset




**Result**: a sequence of clustering assignments

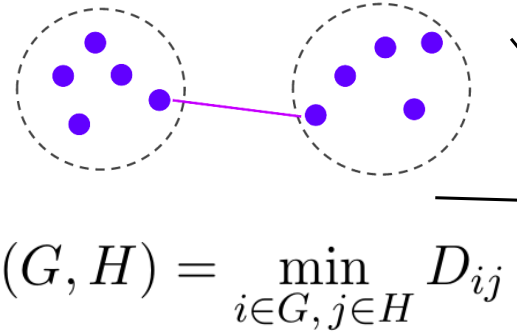
# Hierarchical Clustering (HC)

Distances are used for merges via *agglomerative* or *divisive* schemes.

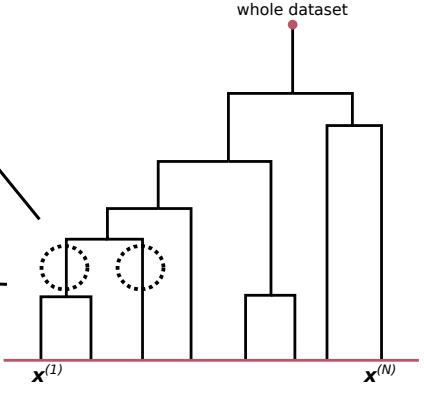
The most established are **agglomerative (bottom-up)** schemes: they proceed from leaves to root by merging groups.

At a given depth, the cluster merge is decided by taking the pair at minimal distance, defined by different linkage criteria:

**1. Single Linkage**



$$d_{\mathrm{SL}}(G, H) = \min_{i \in G,\, j \in H} D_{ij}$$



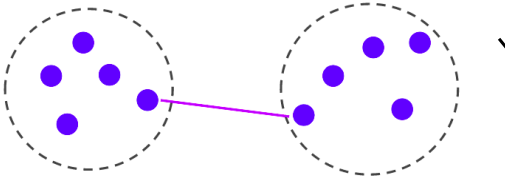
# Hierarchical Clustering (HC)

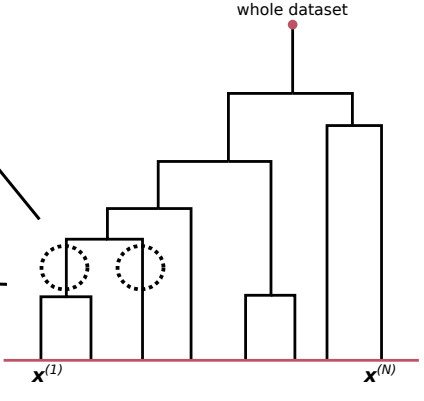
Distances are used for merges via *agglomerative* or *divisive* schemes.

The most established are **agglomerative (bottom-up)** schemes: they proceed from leaves to root by merging groups.

At a given depth, the cluster merge is decided by taking the pair at minimal distance, defined by different linkage criteria:

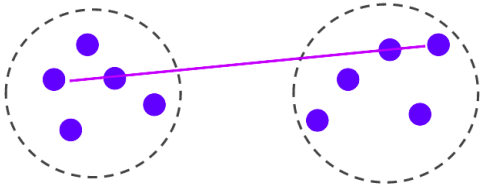
## 1. Single Linkage






$$d_{\mathrm{SL}}(G, H) = \min_{i \in G,\, j \in H} D_{ij}$$

## 2. Complete Linkage



$$d_{\mathrm{CL}}(G, H) = \max_{i \in G,\, j \in H} D_{ij}$$

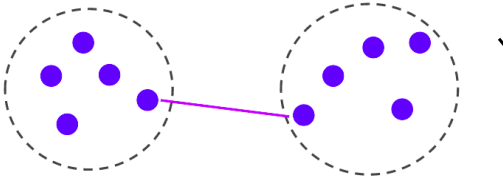
# Hierarchical Clustering (HC)

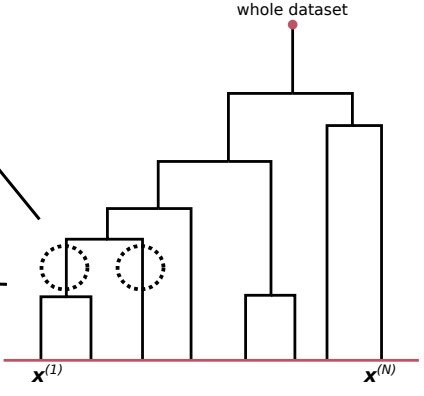
Distances are used for merges via *agglomerative* or *divisive* schemes.

The most established are **agglomerative (bottom-up)** schemes: they proceed from leaves to root by merging groups.

At a given depth, the cluster merge is decided by taking the pair at minimal distance, defined by different linkage criteria:

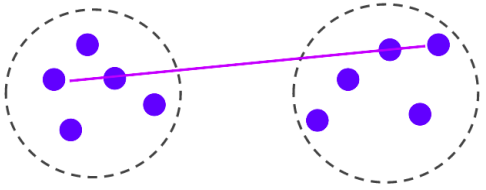
## 1. Single Linkage





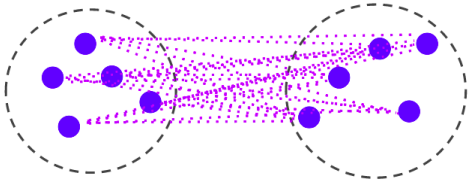

$$d_{\mathrm{SL}}(G, H) = \min_{i \in G,\, j \in H} D_{ij}$$

## 2. Complete Linkage



$$d_{\mathrm{CL}}(G, H) = \max_{i \in G,\, j \in H} D_{ij}$$

## 3. Average Linkage

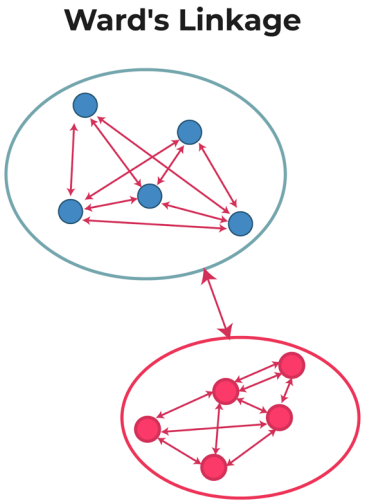


$$d_{\mathrm{GA}}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G, j \in H} D_{ij}$$

Figures from https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/

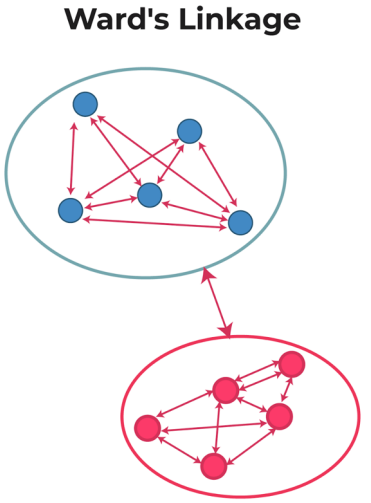
# Hierarchical Clustering (HC)

And other linkage criteria, e.g. *variance*-based: take the merge minimising the within-cluster variance: this is the **Ward's criterion**



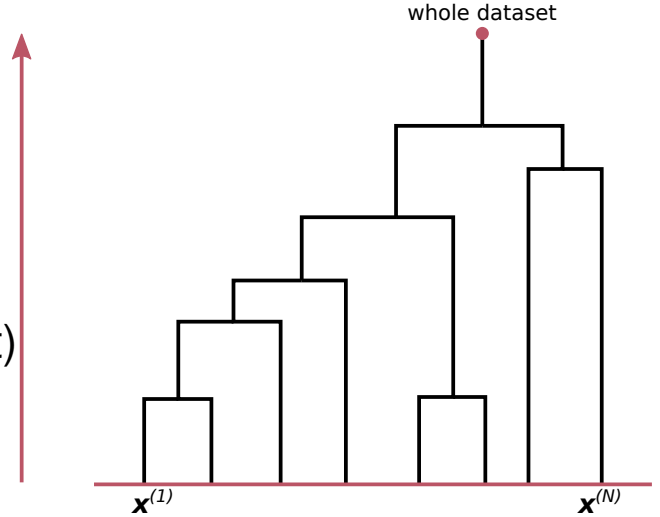
# Hierarchical Clustering (HC)

And other linkage criteria, e.g. *variance*-based: take the merge minimising the within-cluster variance: this is the **Ward's criterion**




**Agglomerative**

(distance minimised at each merge monotonically increases towards the root)




**Divisive**

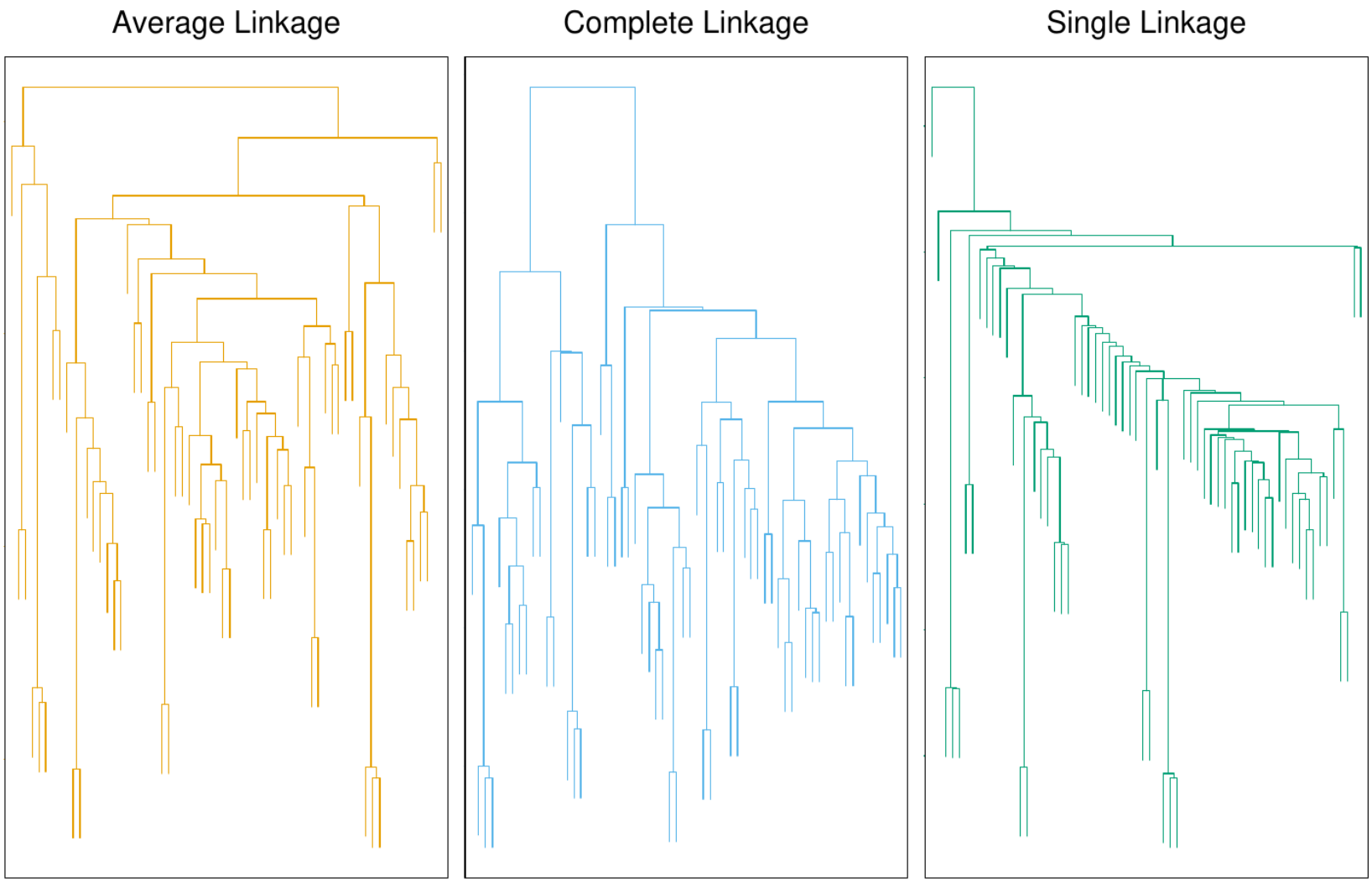
no monotonicity

**Divisive (top-down)** schemes:

by applying recursively *K*-means with (*K*=2), yielding a binary tree.

# Hierarchical Clustering (HC)

HC is sensitive to outliers and the linkage choice:



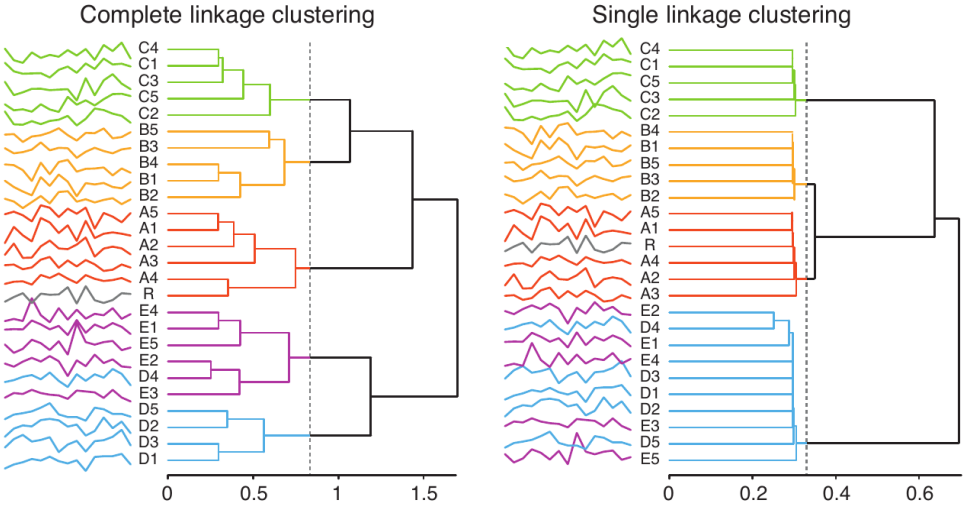

HC Dendrograms for Tumour Microarray Data, Hastie, Tibshirani, Friedman, The Elements of Statistical Learning, Chap. 14

# Hierarchical Clustering (HC)

HC allows for a visualisation of internal substructures of the data across different levels of resolution.

Often used in **bioinformatics**, e.g. genomics

HC dendrograms for gene expression data, from Altman and Krzywinski, Nature Methods 2017 (see 'additional reading')

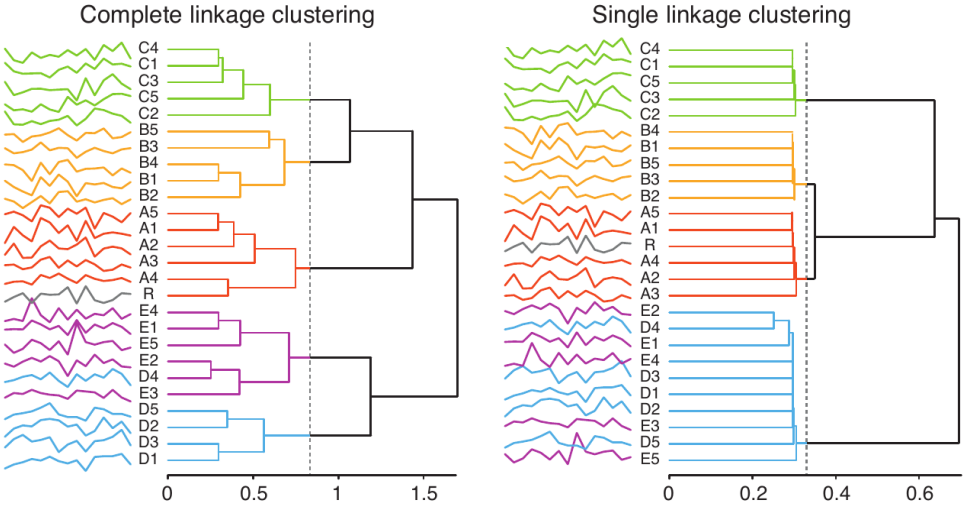


# Hierarchical Clustering (HC)

HC allows for a visualisation of internal substructures of the data across different levels of resolution.

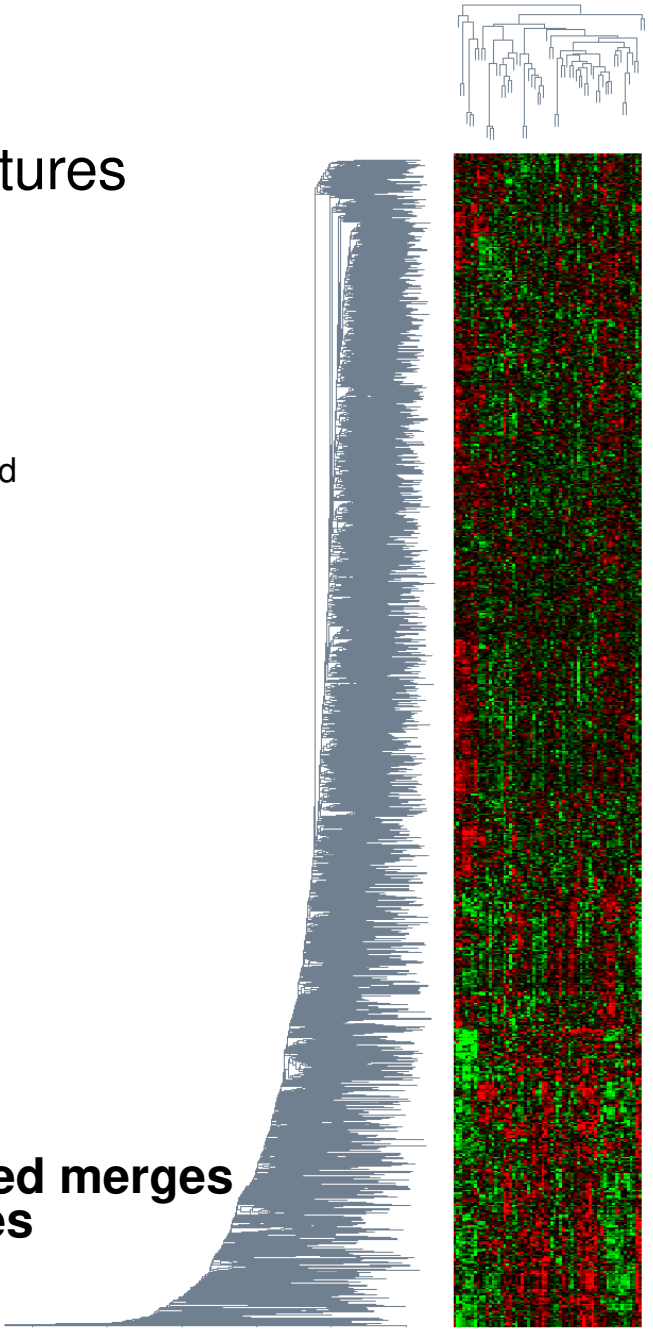
Often used in **bioinformatics**, e.g. genomics

HC dendrograms for gene expression data, from Altman and Krzywinski, Nature Methods 2017 (see 'additional reading')




**Distance-based merges between samples**

**Distance-based merges between genes**



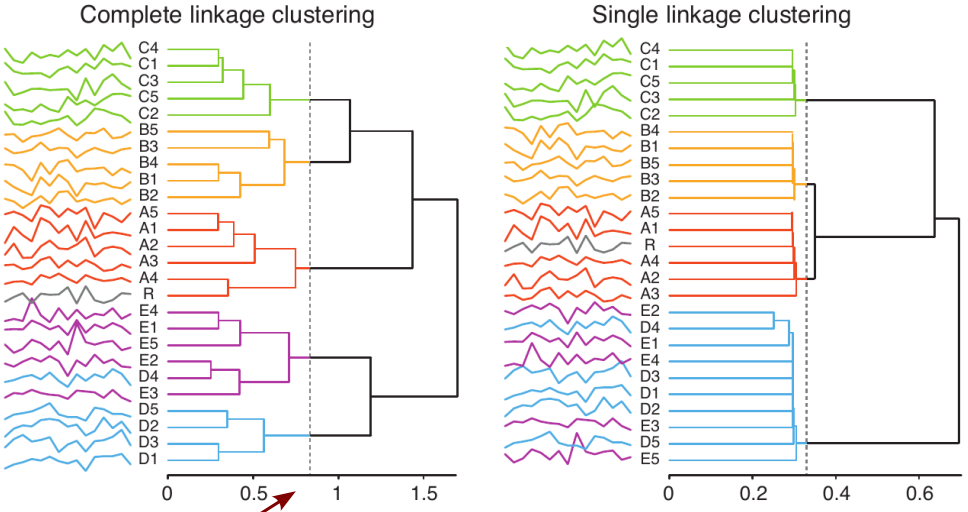
HC Dendrograms for DNA Microarray Data (average linkage), Hastie, Tibshirani, Friedman, The Elements of Statistical Learning}, Chap. 14

# Hierarchical Clustering (HC)

HC allows for a visualisation of internal substructures of the data across different levels of resolution.

Often used in **bioinformatics**, e.g. genomics

HC dendrograms for gene expression data, from Altman and Krzywinski, Nature Methods 2017 (see 'additional reading')



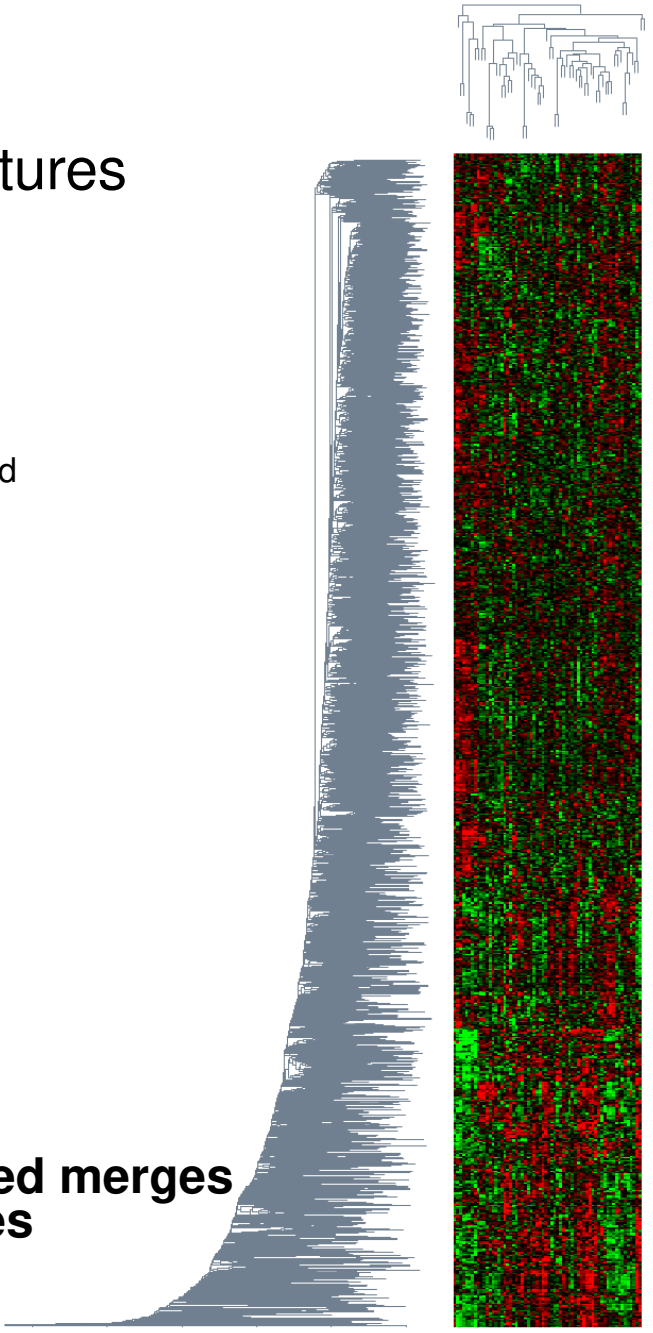

**What level of resolution to pick?**

Stable levels signalled by large jumps in depth

(as depth = function of the between-cluster distance)

**Distance-based merges between samples**

**Distance-based merges between genes**



HC Dendrograms for DNA Microarray Data (average linkage), Hastie, Tibshirani, Friedman, The Elements of Statistical Learning}, Chap. 14

# Evaluating clustering performance

**External validity:** ground truth grouping well reproduced;
**Internal validity:** intra-cluster cohesion, inter-cluster separation

Let's recall:

$$W(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\in c_k} D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

within-cluster distance

$$B(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in c_k}\sum_{j\notin c_k} D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

between-cluster distance

They, and their combinations, are measures of **clustering quality**

# Evaluating clustering performance

**External validity:** ground truth grouping well reproduced;
**Internal validity:** intra-cluster cohesion, inter-cluster separation

Let's recall:

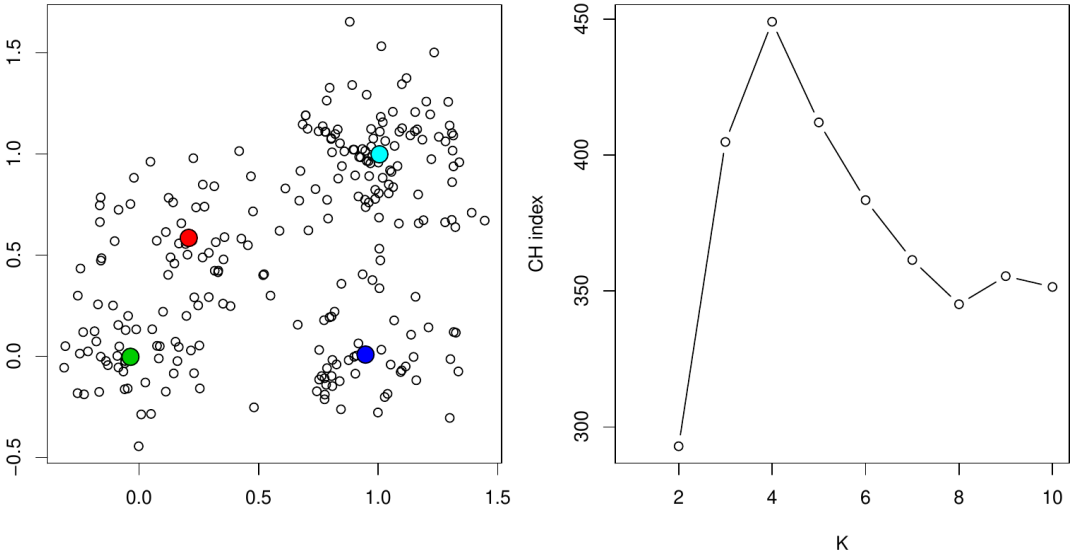
$$W(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\in c_k} D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

within-cluster distance

$$B(\mathcal{C}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in c_k}\sum_{j\notin c_k} D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

between-cluster distance

They, and their combinations, are measures of **clustering quality**




e.g. Calinski-Harabasz (CH) index

$$CH = \frac{B(\mathcal{C})/(K-1)}{W(\mathcal{C})/(N-K)}$$

its maximum helps set optimal *K*

# Comparing clusterings

**Count matrix:** of coincidences in each cluster of 2 clusterings

Clustering 2

| | $Y_1$ | $Y_2$ | $\cdots$ | $Y_{K'}$ |
|---|---|---|---|---|
| $X_1$ | $n_{11}$ | $n_{12}$ | $\cdots$ | $n_{1K'}$ |
| $X_2$ | $n_{21}$ | $n_{22}$ | $\cdots$ | $n_{2K'}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $X_K$ | $n_{K1}$ | $n_{K2}$ | $\cdots$ | $n_{KK'}$ |

Clustering 1

$$n_{ij} = |X_i \cap Y_j|$$

Frequency of coincidences:

$$P_{XY}(i,j) = \frac{|X_i \cap Y_i|}{N}$$

# Comparing clusterings

**Count matrix:** of coincidences in each cluster of 2 clusterings

Clustering 2

| | $Y_1$ | $Y_2$ | $\cdots$ | $Y_{K'}$ |
|---|---|---|---|---|
| $X_1$ | $n_{11}$ | $n_{12}$ | $\cdots$ | $n_{1K'}$ |
| $X_2$ | $n_{21}$ | $n_{22}$ | $\cdots$ | $n_{2K'}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $X_K$ | $n_{K1}$ | $n_{K2}$ | $\cdots$ | $n_{KK'}$ |

Clustering 1

$$n_{ij} = |X_i \cap Y_j|$$

Frequency of coincidences:

$$P_{XY}(i,j) = \frac{|X_i \cap Y_i|}{N}$$

We can define **information-theoretic comparison metrics:**

**Mutual Information:**

$$MI(X,Y) = \sum_{i=1}^{K}\sum_{j=1}^{K'} P_{XY}(i,j)\log\frac{P_{XY}(i,j)}{P_X(i)P_Y(j)}$$

Clustering entropy

$$H(X) = -\sum_{i=1}^{K} P_X(i)\log P_X(i)$$

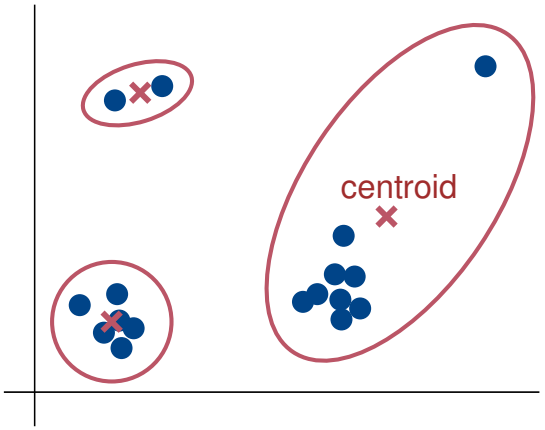
**Normalised Variation of Information:**

$$NVI(X,Y) = \frac{H(X)+H(Y)-2MI(X,Y)}{H(X)+H(Y)-MI(X,Y)}$$

and others, see lecture notes

# Summary: 3 clustering methods

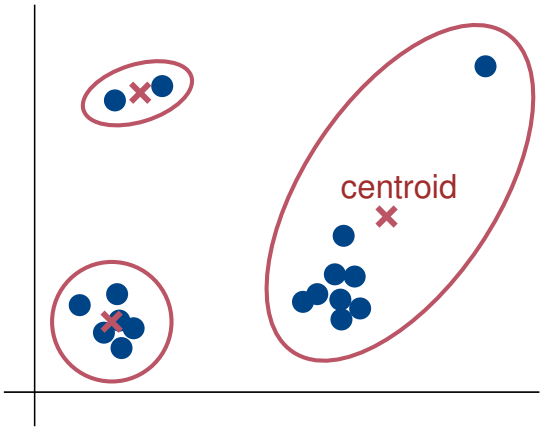
**_K_-Means**



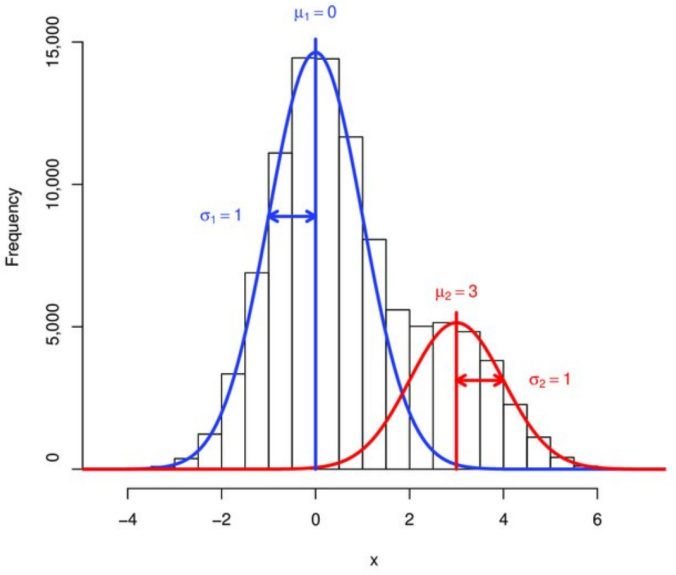

Hard clustering
Distance-based
Easy iterative algorithm
(sensitive to initialisation)

# Summary: 3 clustering methods

## *K*-Means




## GMM

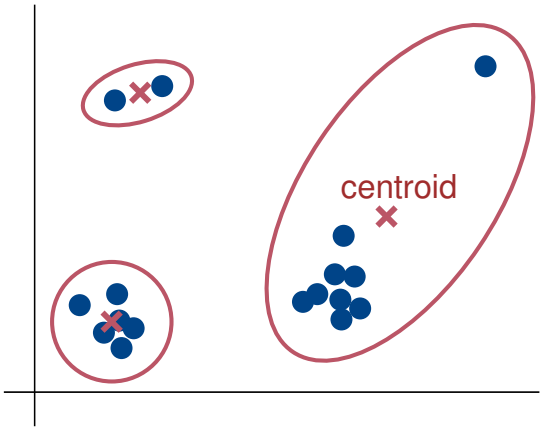



Hard clustering
Distance-based
Easy iterative algorithm
(sensitive to initialisation)

Soft clustering
Distribution-based
EM iterative algorithm
(more costly but flexible)

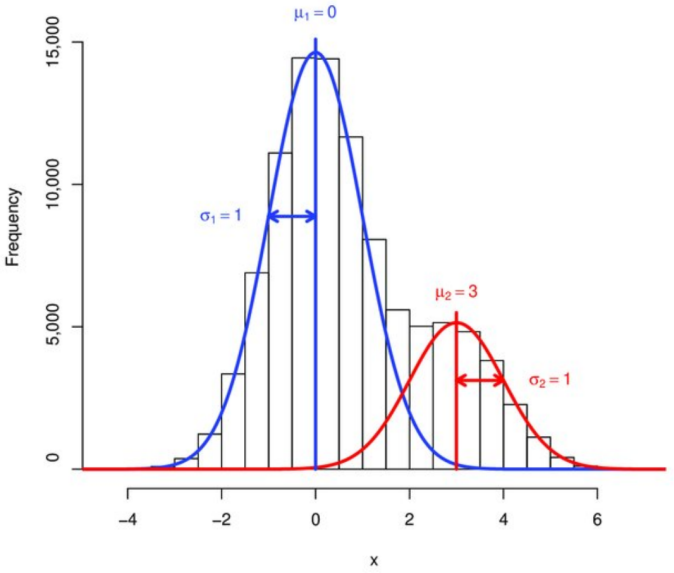
# Summary: 3 clustering methods

## *K*-Means



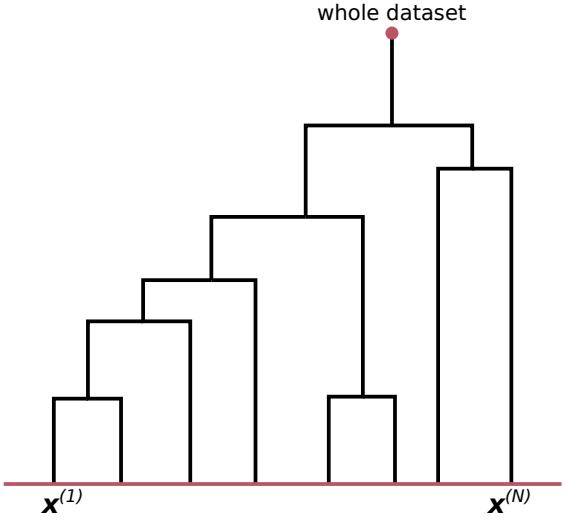

Hard clustering
Distance-based
Easy iterative algorithm
(sensitive to initialisation)

## GMM




Soft clustering
Distribution-based
EM iterative algorithm
(more costly but flexible)

## Hierarchical




Hard clustering
Distance-based
**Multi-resolution**
similarity structure

**Need to choose *K***