



UNIVERSITY  
OF AMSTERDAM

# Machine Learning for Physics and Astronomy

Juan Rojo

VU Amsterdam & Theory group, Nikhef

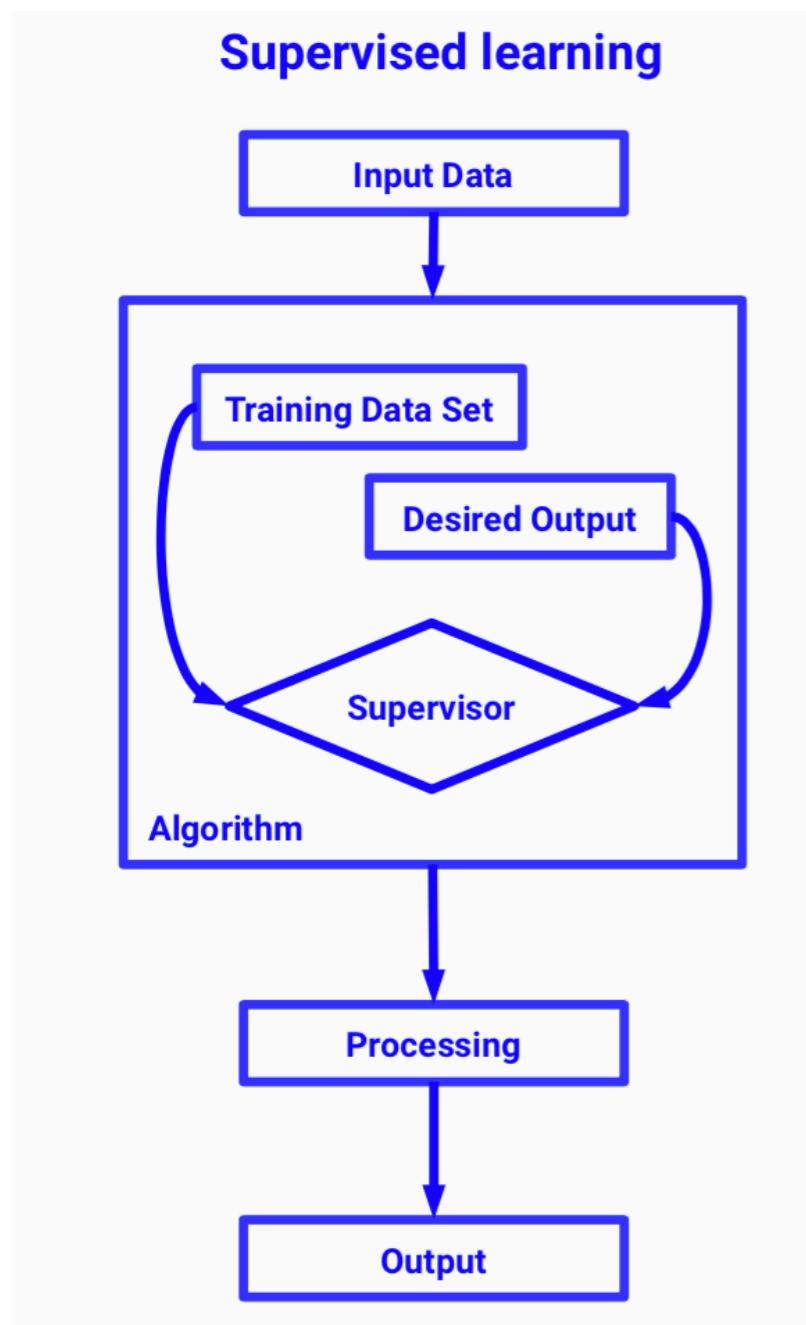
***Natuur- en Sterrenkunde BSc (Joint Degree), Honours Track***  
***Lecture 3, 21/09/2020***

# Today's lecture

- Unsupervised Learning & Clustering
- Data visualisation and dimensional reduction
- Ensemble methods and Bootstrapping
- Random Forests and Decision Trees
- Guest lecture by **Dr. Atul Chhotray** on **ML** for classification in astronomy

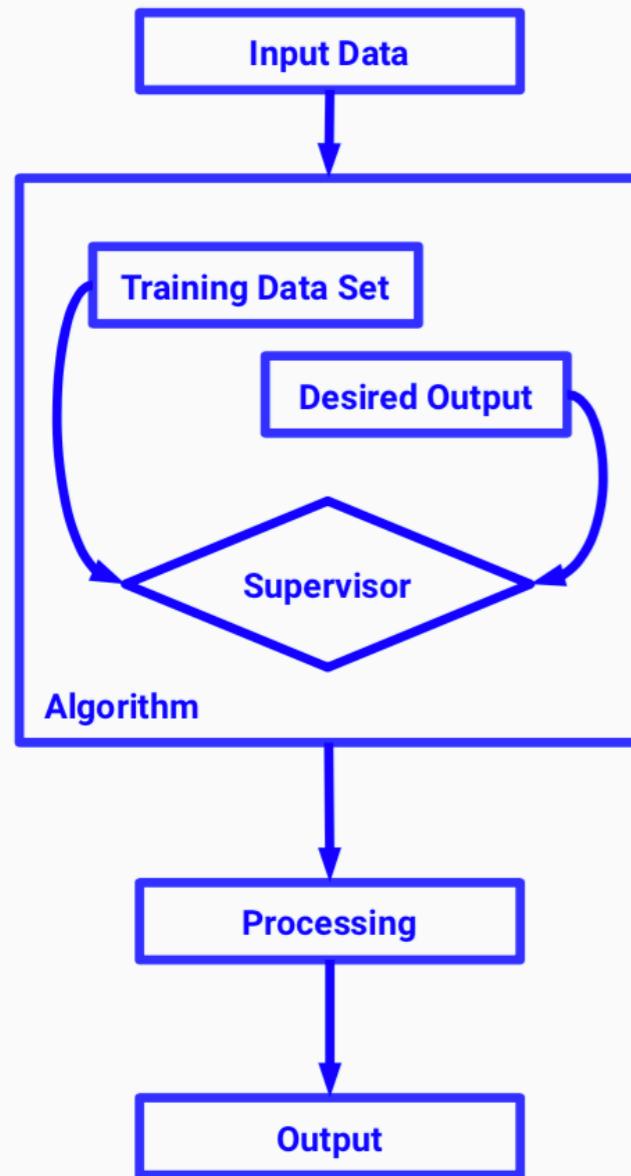
# **Unsupervised Learning**

# Supervised vs Unsupervised Learning

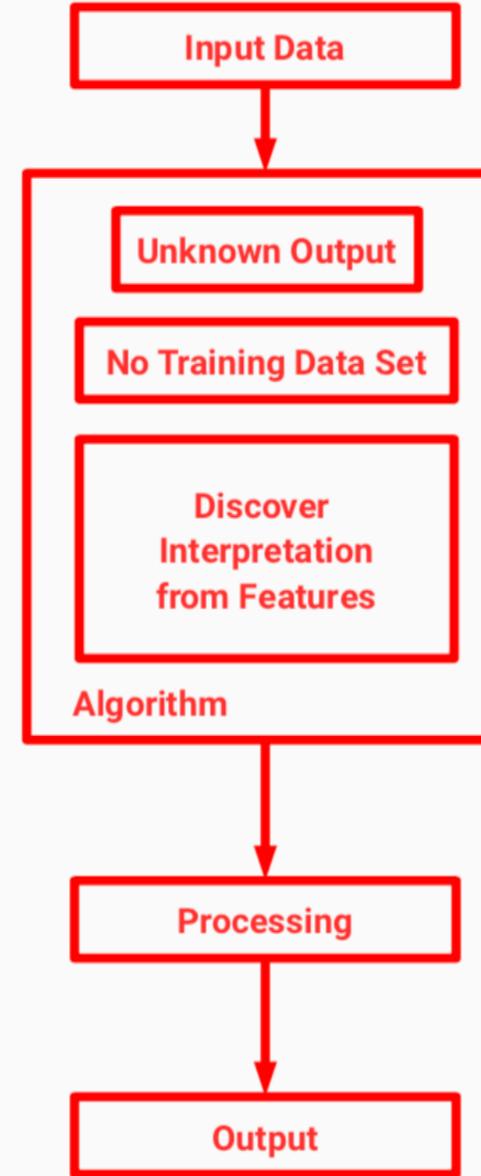


# Supervised vs Unsupervised Learning

## Supervised learning



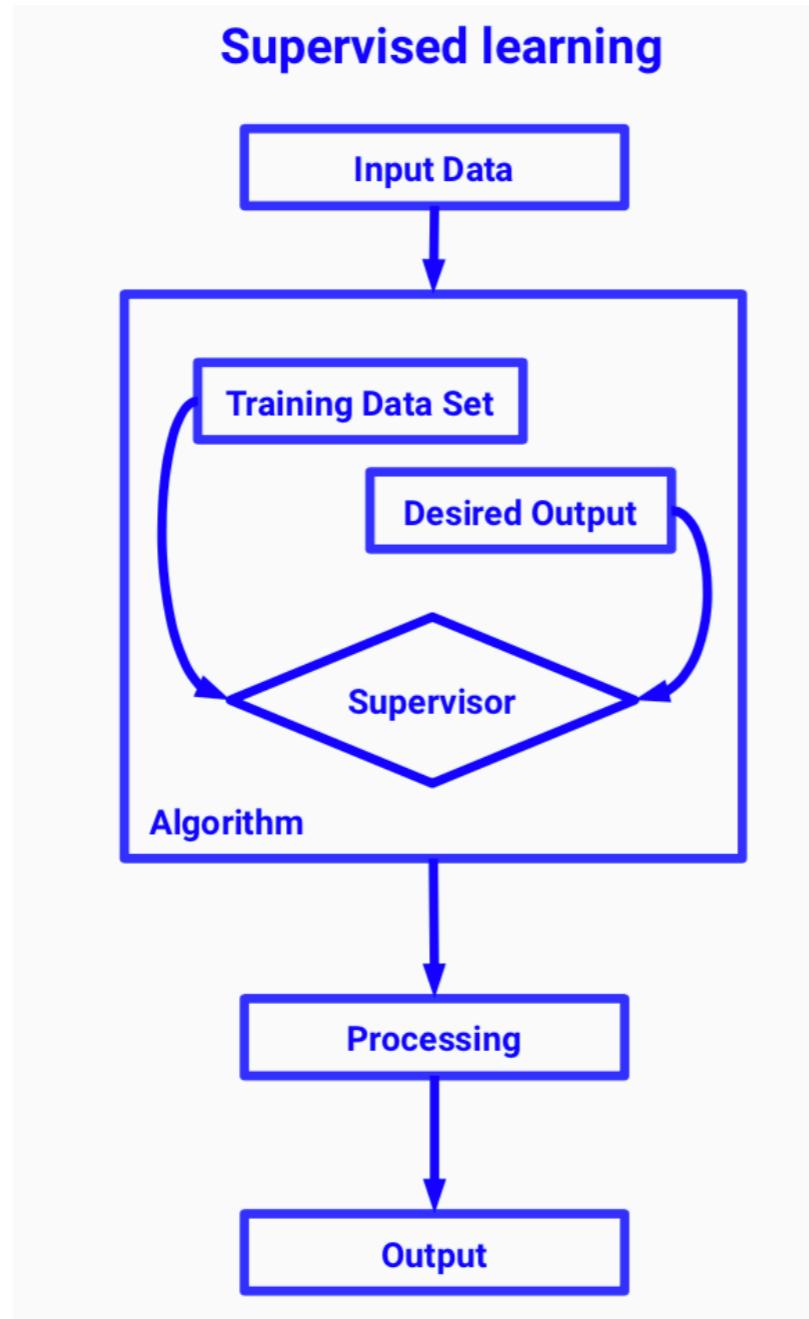
## Unsupervised learning



## Reinforcement learning

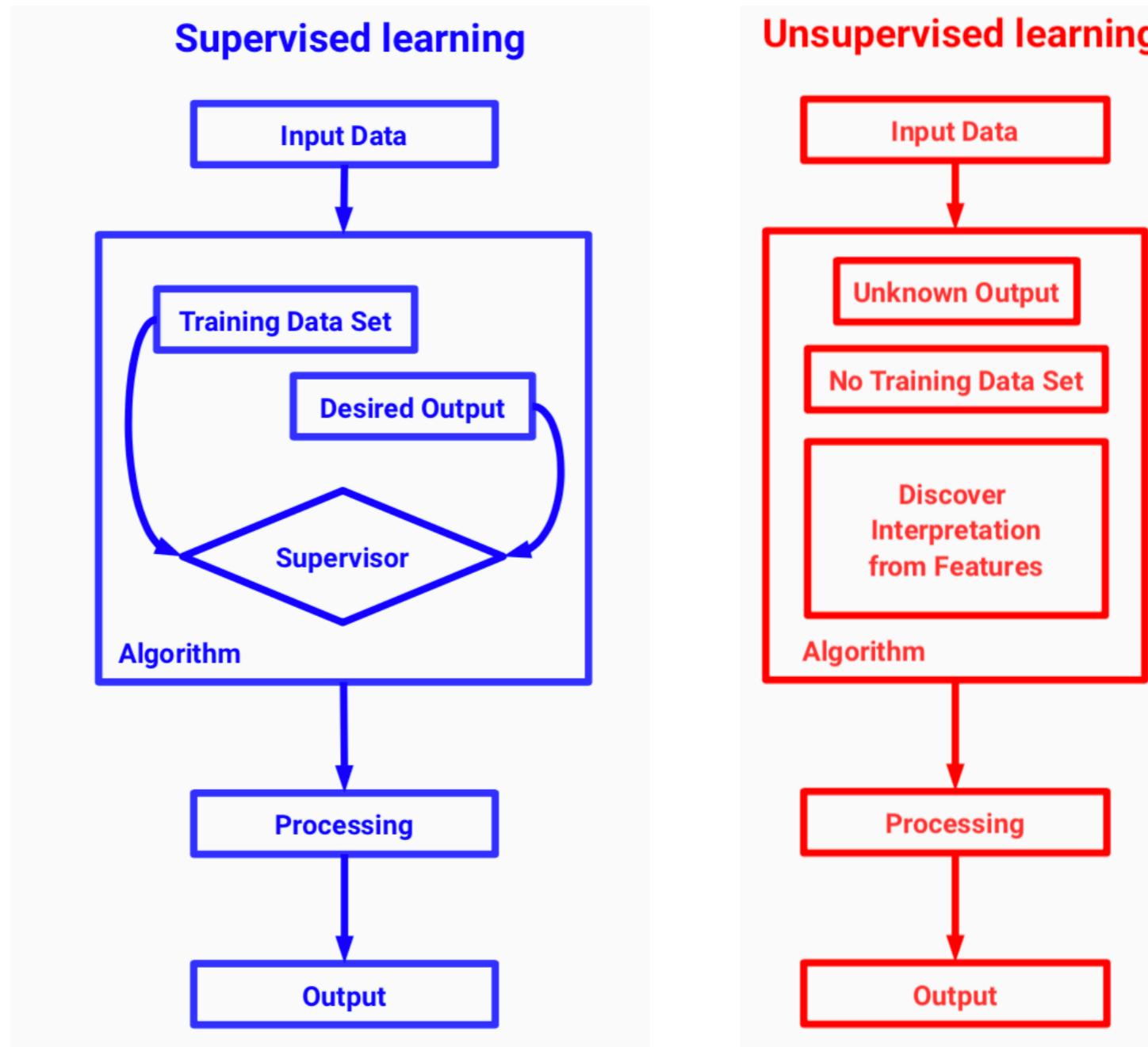


# Supervised vs Unsupervised Learning



Supervised Learning: find a **model** that reproduces the underlying law of a set of **labelled input/output patterns**

# Supervised vs Unsupervised Learning



Unsupervised Learning: there are no labels and our **aim is to identify underlying structures and connections** present in the data

# Clustering

In ML context, **unsupervised learning** is concerned with discovering underlying structures in **unlabelled data**

an important example of unsupervised learning is **clustering**: the aim is to group unlabelled data into clusters using some **distance or similarity measure**

let us illustrate these ideas with **K-means clustering**

$$\{\boldsymbol{x}_n\}_{n=1}^N \quad \boldsymbol{x}_n = (x_{n,1}, x_{n,2}, \dots, x_{n,p})$$

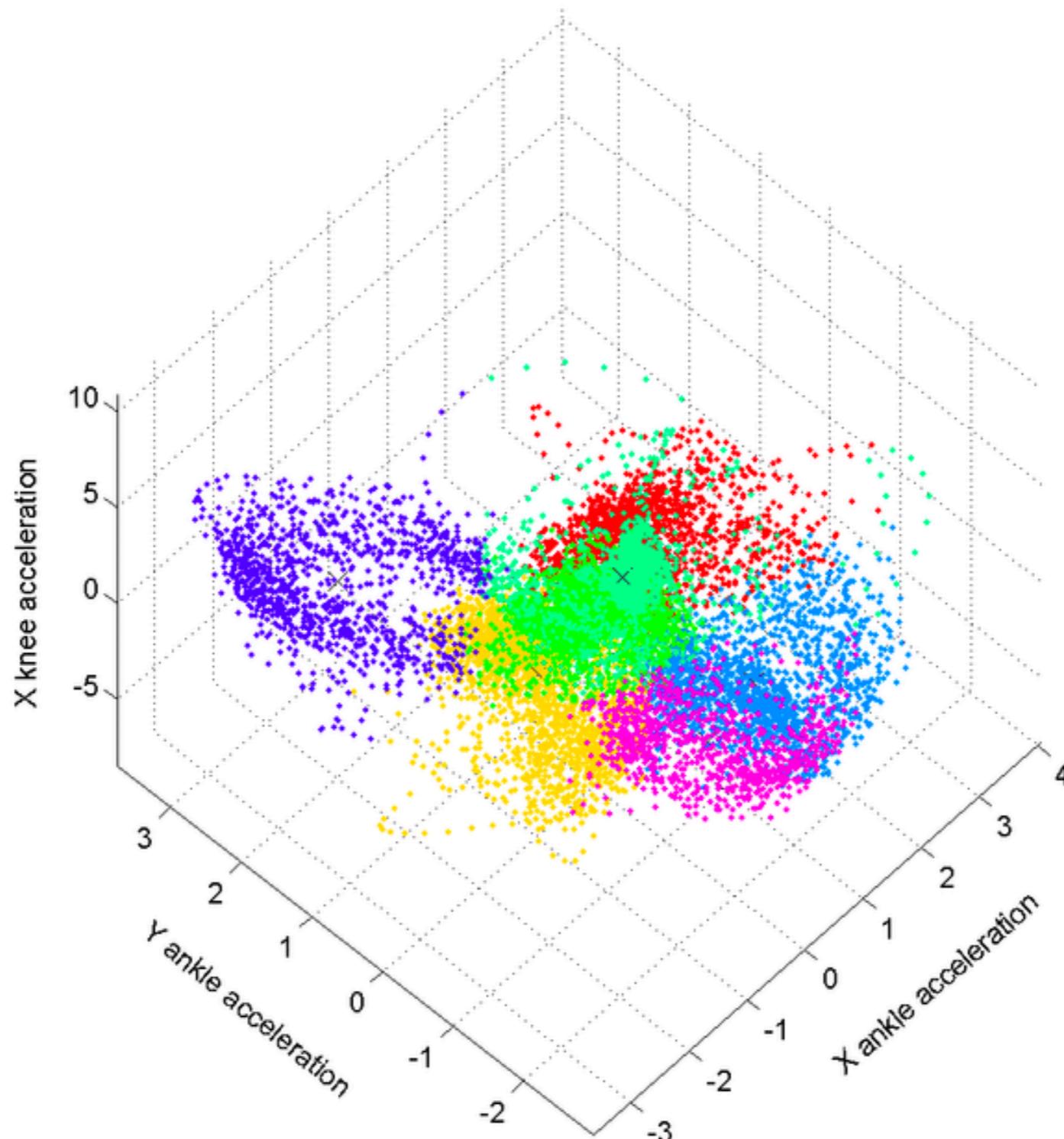
*unlabelled dataset: N points with p features each*

$$\{\boldsymbol{\mu}_k\}_{k=1}^K \quad \boldsymbol{\mu}_k = (\mu_{k,1}, \mu_{k,2}, \dots, \mu_{k,p})$$

**cluster means**: K clusters with p features each

the intuitive idea is that the cluster means represent the **main features of each cluster**, to which the data points will be assigned in the clustering procedure

# Unsupervised Learning



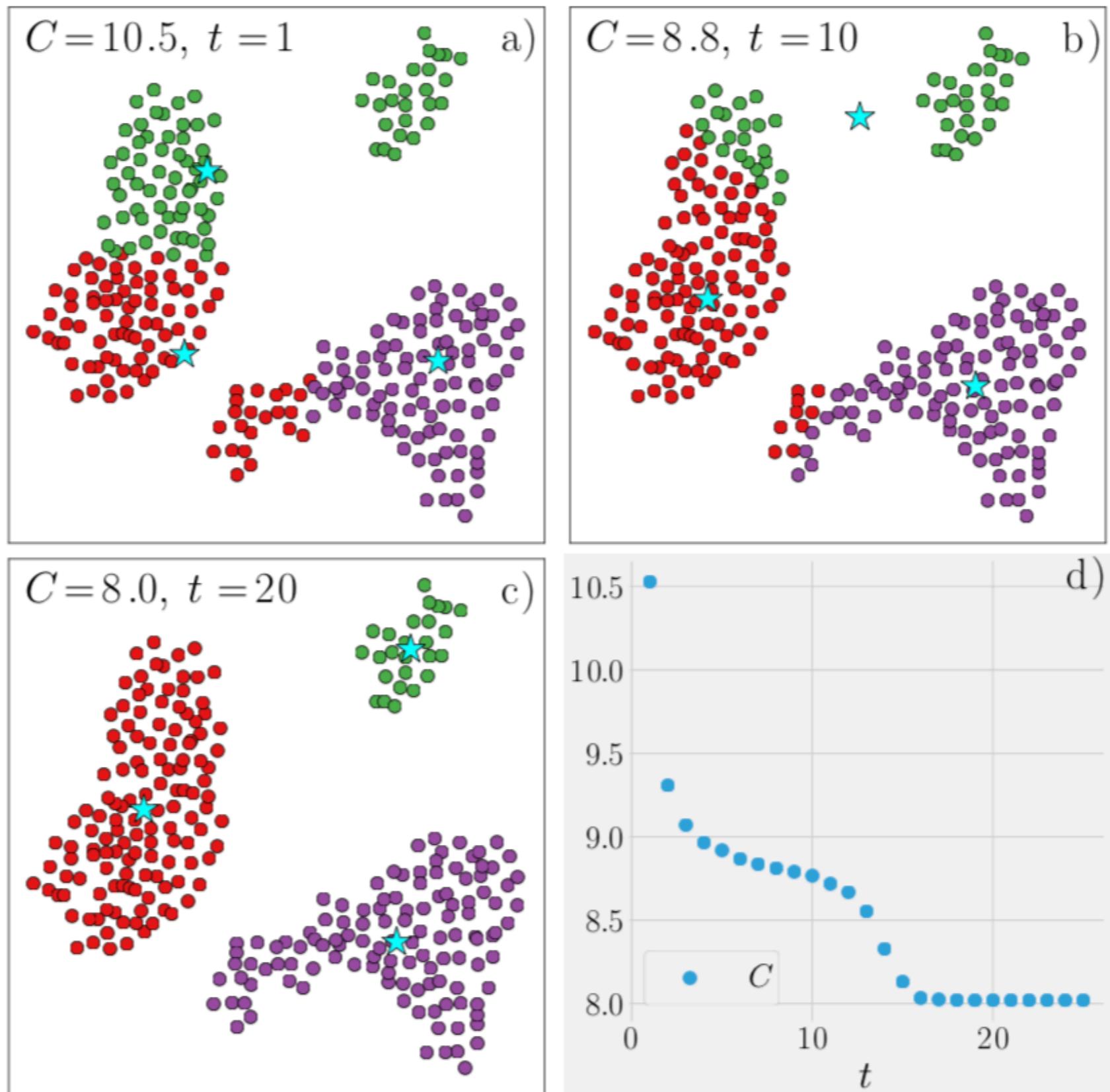
*nb color here for  
visualisation, but not  
this info not available in  
real applications!*

how many ``**groups of samples**'' do we have have in our dataset?

# Clustering

**2D example of clustering:** each colour represents a cluster, with stars indicating their **centers**

how is this clustering achieved **in practice?**



# Clustering

in  $K$ -means clustering, the **cluster means** and the **data point assignments** are determined from the minimisation of a cost function:

$$C(x; \mu) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} (x_n - \mu_k)^2$$

*binary assignment variable*

*Euclidean distance between  $n$ -th data point and  $k$ -th cluster centre*

$r_{nk} = 1 \longrightarrow$  *the  $n$ -th point is assigned to the  $k$ -th cluster*

$r_{nk} = 0 \longrightarrow$  *the  $n$ -th point is not assigned to the  $k$ -th cluster*

furthermore since **clustering is exclusive** one needs to impose:

$$\sum_{k=1}^K r_{nk} = 1 \quad \forall n$$

one sees that  $K$ -means clustering aims to **minimise the variance within each cluster**

# Clustering

Let's describe an algorithm that implements  $K$ -means clustering by minimising the cost function

$$C(\mathbf{x}; \boldsymbol{\mu}) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^2$$

this algorithm alternates iteratively between two main steps:

• (1) **Expectation:** starting from set of cluster assignments  $\{r_{nk}\}$  minimise  $C$  wrt cluster means

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} C(\mathbf{x}; \boldsymbol{\mu}) = 0 \quad \rightarrow \quad \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N r_{nk} \mathbf{x}_n \quad N_k = \sum_{n=1}^N r_{nk}$$

*number of points  
in k-th cluster*

• (2) **Maximization:** given the  $K$  cluster centers, the assignments  $\{r_{nk}\}$  should minimise  $C$ . This can be achieved by assigning each data point to its closest cluster-mean

$$r_{nk} = 1 \quad \text{if} \quad k = \arg \min_{k'} (\mathbf{x}_n - \boldsymbol{\mu}_{k'})$$

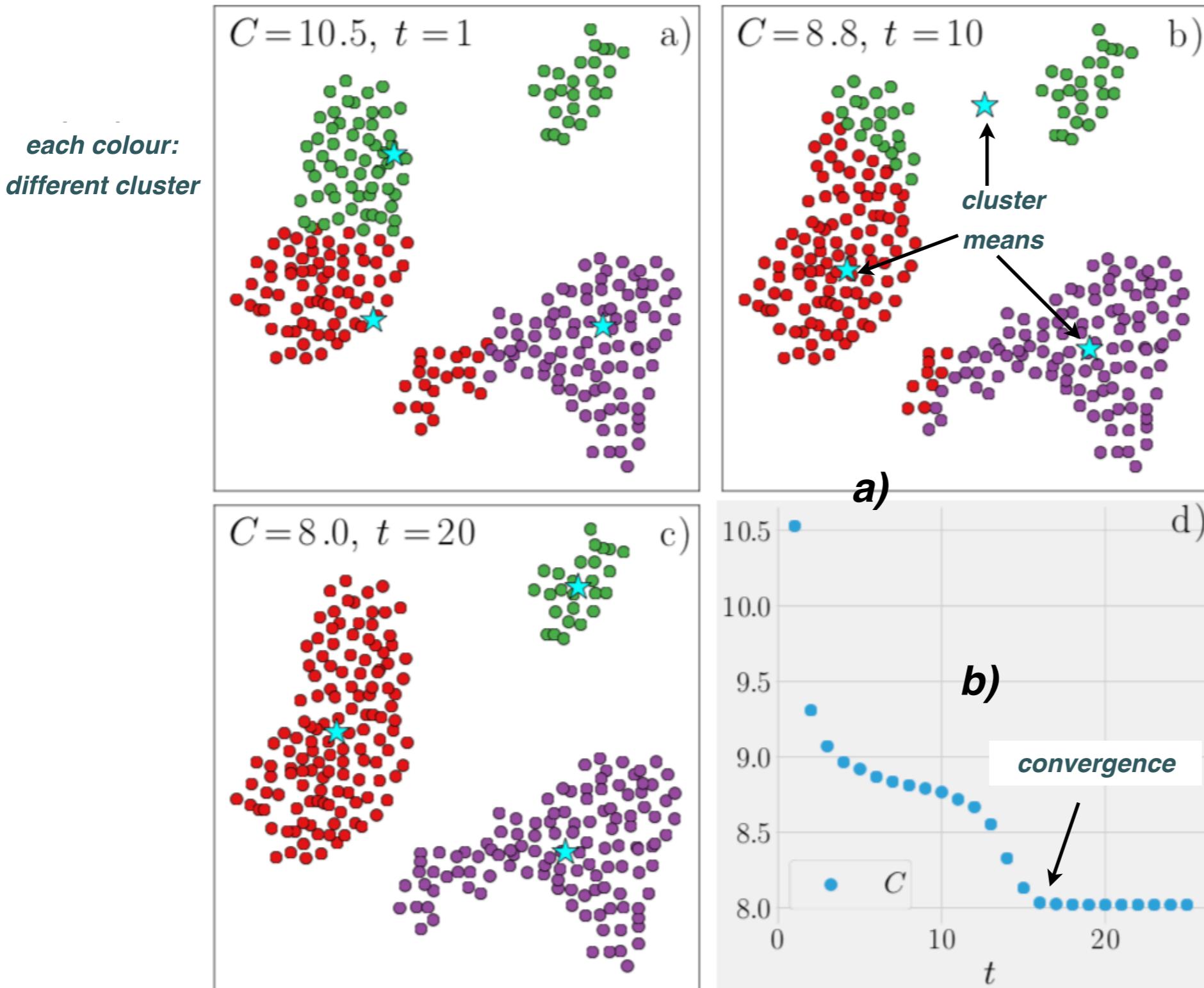
*iterate until convergence  
achieved!*

$$r_{nk} = 0 \quad \text{if} \quad k \neq \arg \min_{k'} (\mathbf{x}_n - \boldsymbol{\mu}_{k'})$$

*note that here GD not  
required, optimisation is  
semi-analytical*

# Clustering

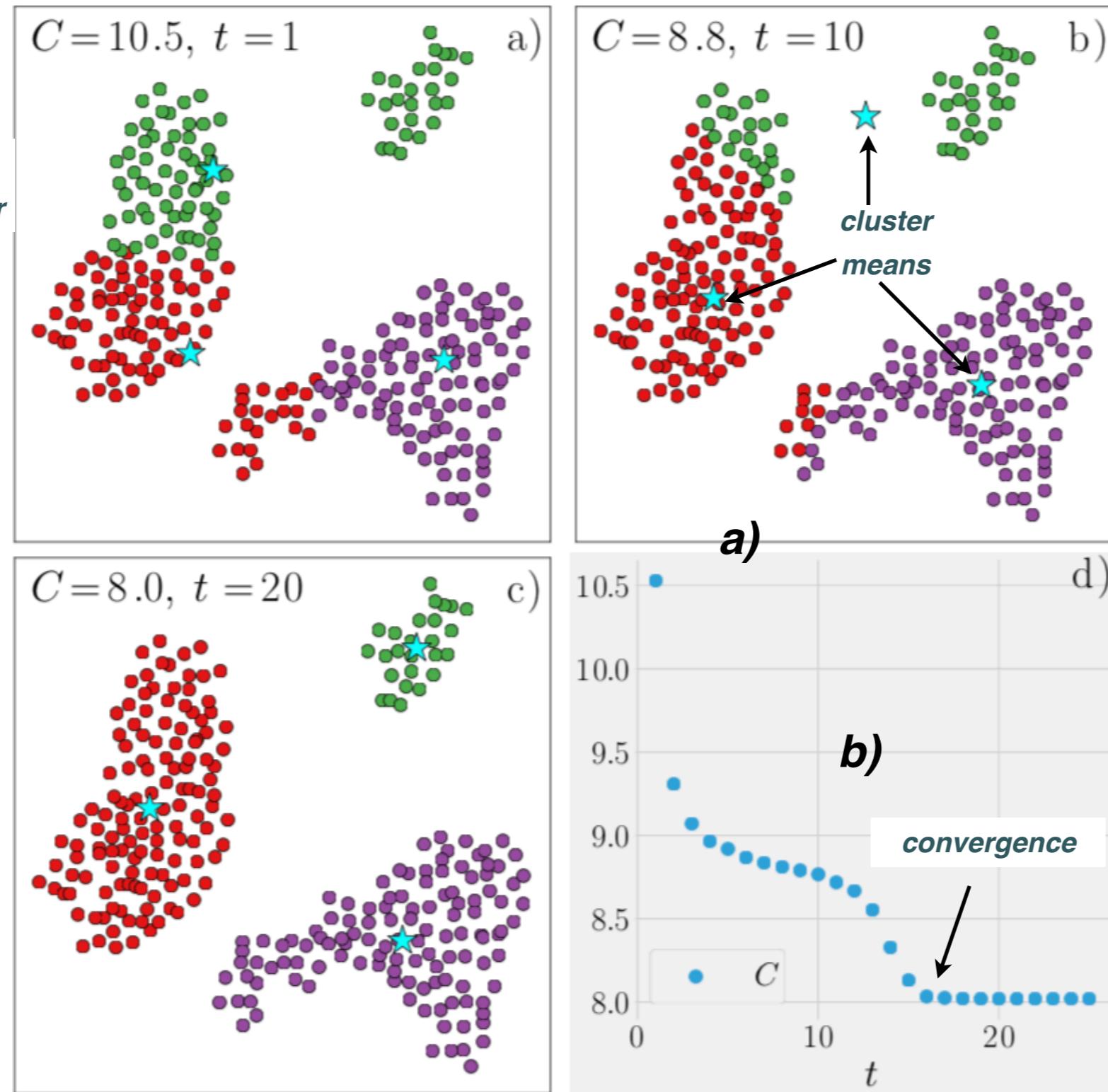
these two steps are iterated until some **convergence criterion** is achieved, e.g. when the change in the cost function between two iterations is below some threshold



here overfitting not possible:  
there exists a unique assignment  
that minimises the cost function

# Clustering

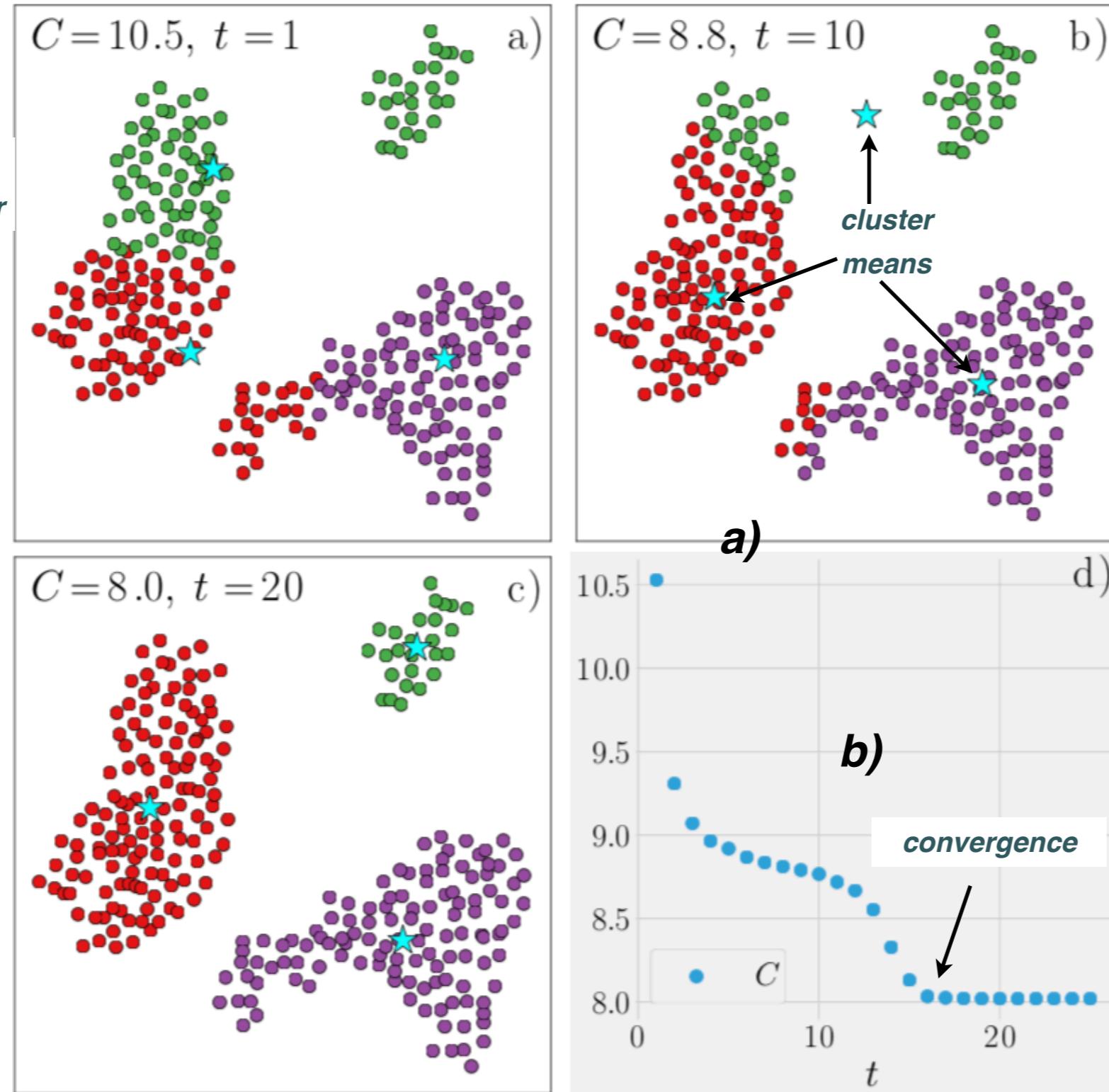
these two steps are iterated until some **convergence criterion** is achieved, e.g. when the change in the cost function between two iterations is below some threshold



*what is the underlying assumption in K-means clustering? And when it would not be justified?*

# Clustering

these two steps are iterated until some **convergence criterion** is achieved, e.g. when the change in the cost function between two iterations is below some threshold



K-means clustering can lead to spurious results since the **underlying assumption** is that the latent model has uniform variances

fails if the underlying clusters have different variances!

# Hierarchical clustering

- Another approach to clustering is based on **agglomerative methods**, where one starts from small clusters which are progressively **merged into bigger clusters**
- This hierarchical structure provides information on the relations between clusters and the **subcomponents of individual clusters**
- As before, we need to specify a **distance**, this time between two clusters  $X, Y$

$$d(X, Y) \in \mathcal{R}$$

- At each iteration, the two clusters closer to each other (quantified by  $d$ ) are *merged*

the **agglomerative cluster algorithm** works as follows:

- (1) Assign **each data point to be its own cluster**
- (2) Given the resulting set of  $K$  clusters, find the closest pair
$$(X_i, X_j) \text{ such that } (i, j) = \arg \min_{i'j'} d(X_{i'}, X_{j'})$$
- (3) Merge the pair into a single cluster. Iterate (2) and (3) until a single cluster remains

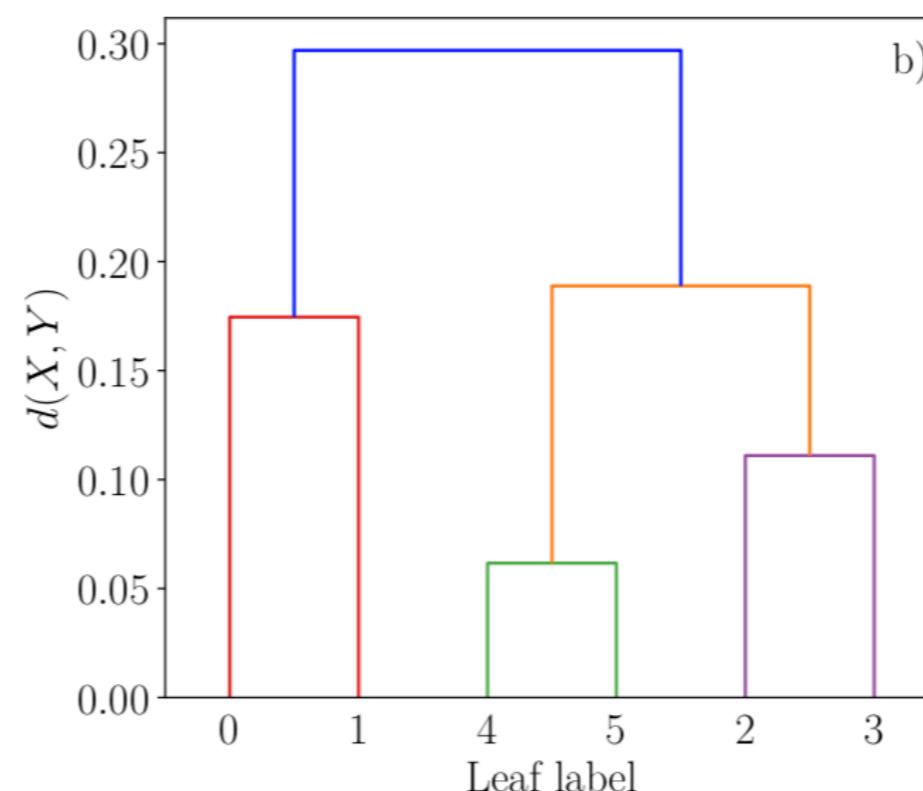
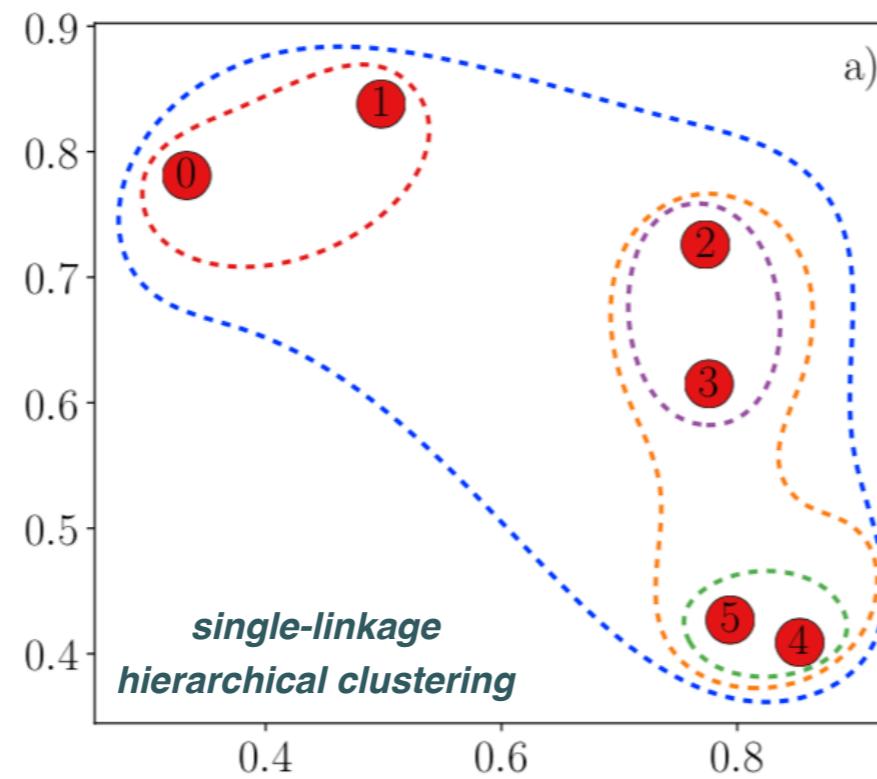
# Hierarchical clustering

Clearly the results of hierarchical clustering depend on the **choice of distance**

*distance between clusters*

single linkage  $\longrightarrow d(X_i, X_j) = \min_{x_i \in X_i, x_j \in X_j} \|x_i - x_j\|_2$  Euclidean  
distance

complete linkage  $\longrightarrow d(X_i, X_j) = \max_{x_i \in X_i, x_j \in X_j} \|x_i - x_j\|_2$



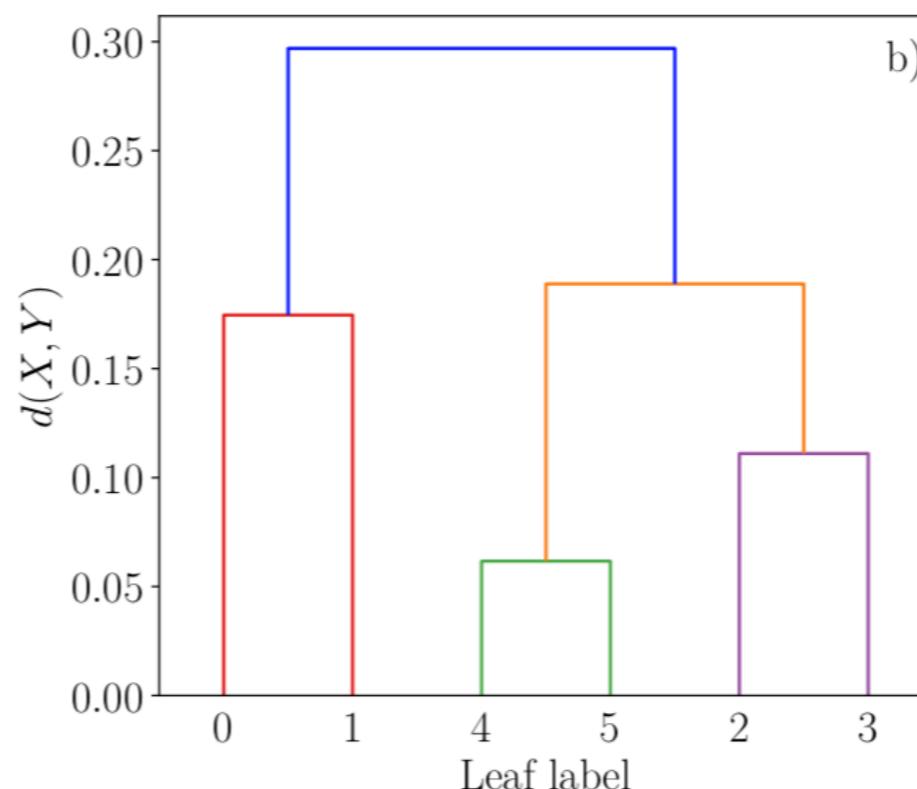
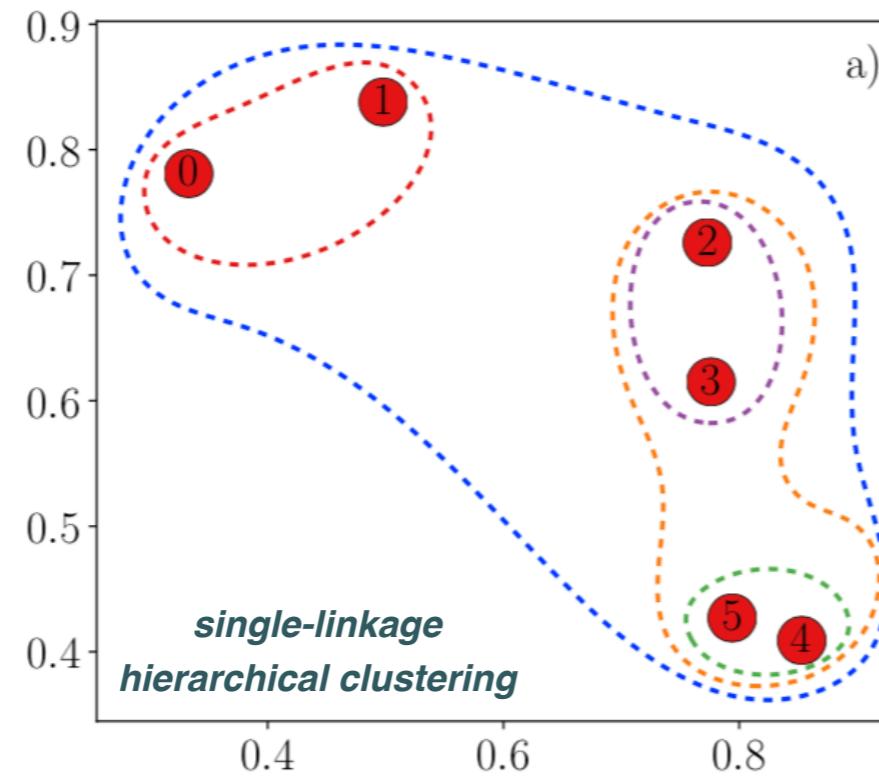
# Hierarchical clustering

Clearly the results of hierarchical clustering depend on the **choice of distance**

*distance between clusters*

single linkage  $\longrightarrow d(X_i, X_j) = \min_{x_i \in X_i, x_j \in X_j} \|x_i - x_j\|_2$  Euclidean  
distance

complete linkage  $\longrightarrow d(X_i, X_j) = \max_{x_i \in X_i, x_j \in X_j} \|x_i - x_j\|_2$



hierarchical clustering methods do not scale well for large  $N$ , so they are typically **combined with K-means clustering** in the initial steps to define small clusters

# Clustering and Latent variables

A central concept in **Unsupervised Learning** is that of a **latent or hidden variable**: not directly observable, but still they influence visible structure of data

*e.g. in clustering, a latent variable is the cluster identity of each datapoint*

One can think of clustering as an algorithm to learn **the most probable value of a latent variable**

a common feature of all Unsupervised Learning algorithms is the need for assumptions about the underlying probability distribution of the data: the **generative model**

*e.g. in K-means clustering, we assume that the points of each cluster are generated Gaussianly with respect to its mean (center)*

$$C(x; \mu) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} (x_n - \mu_k)^2$$

different **generative models** lead to different types of clustering algorithms

# Gaussian Mixture Models

in **Gaussian Mixture Models (GMM)**, a generative model used in clustering applications, points are drawn from  $K$  gaussians with different means and covariances

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim \exp \left[ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})^T \right] \quad \text{one of the } K \text{ gaussians}$$

The probability of generating a point  $\mathbf{x}$  in a GMM is given by

$$p(\mathbf{x}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}) = \sum_{k=1}^K \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \pi_k$$

*probability of drawing a point from mixture k*

$$\theta = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$$

*GMM parameters*

the probability that a data point  $\mathbf{x}$  is associated to the  $k$ -th cluster is

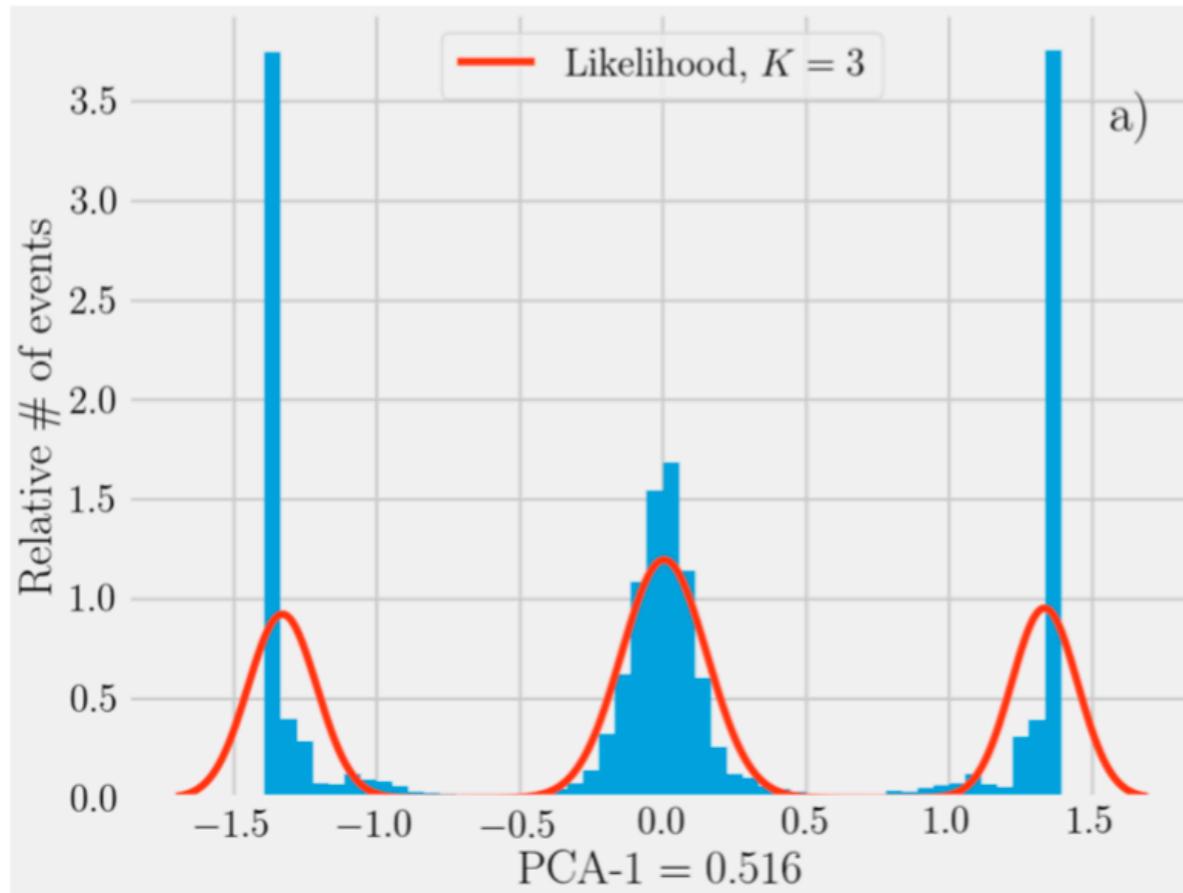
$$\gamma_k(\mathbf{x}) \sim \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \pi_k$$

model parameters found by maximising likelihood using SGD

$$\hat{\theta} = \arg \max_{\theta} \log p(X | \theta)$$

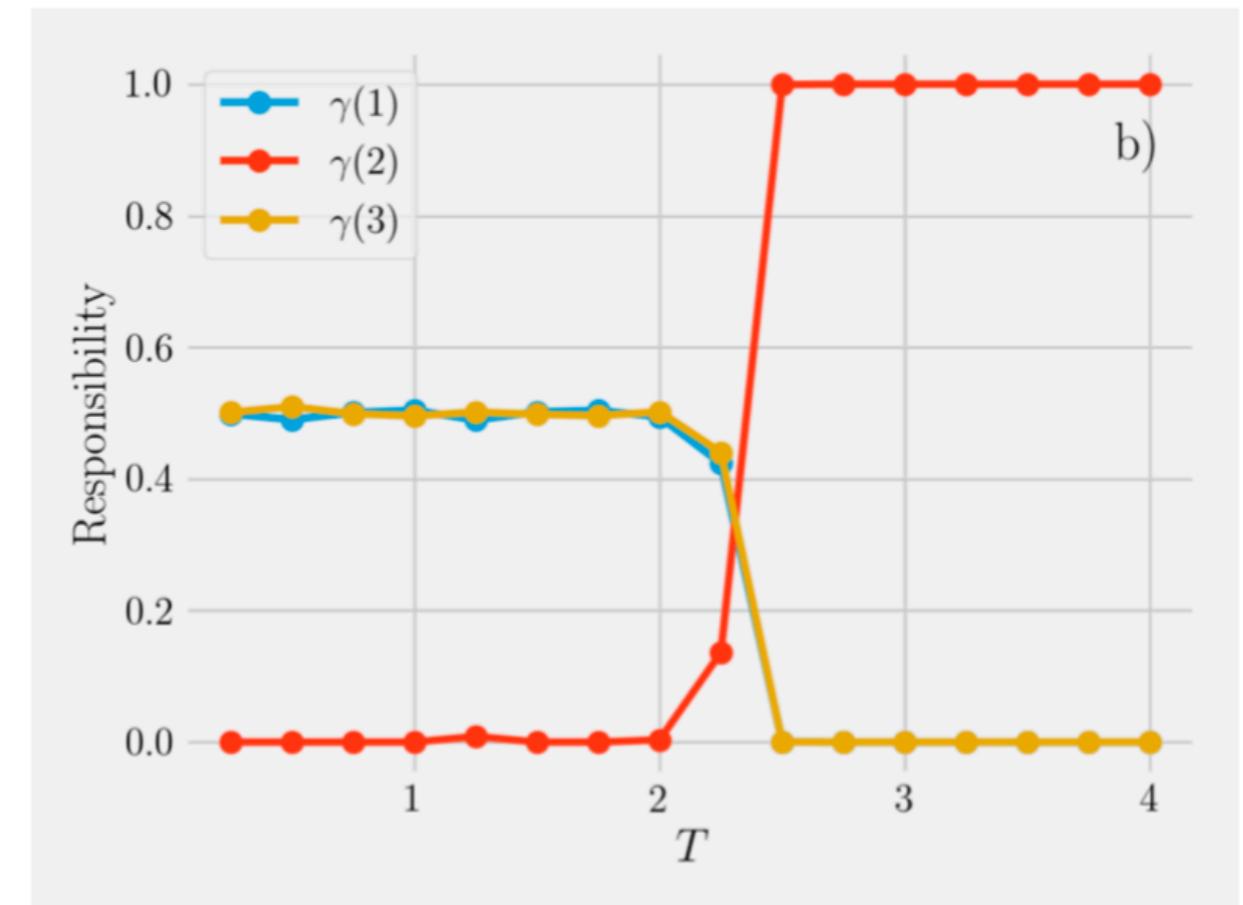
# Gaussian Mixture Models

*Ising dataset fitted to 3-component GGM*



*1st principal component  
(magnetisation)*

*Probability of being on each phase*



*probability coincide at critical point*

recall that this model has been trained only on examples: **no knowledge of the underlying physical mechanisms** whatsoever

# **Dimensional Reduction & Data Visualisation**

# Dimensional reduction

ML problems often deal with samples of **very high dimensionality!**

# Dimensional reduction

Efficient **data visualisation techniques** are essential to construct better models in ML applications eg by **identifying correlated, redundant, or irrelevant features**

Traditional data visualisation methods are not practical when the datasets involve a large number of features (such as images) and we need to project the data onto a lower-dimensional space, called the **latent space**, using **dimensional reduction**

*A good data visualisation strategy is very useful to identify the most suitable strategy to approach a ML problem*

# Dimensional reduction

Efficient **data visualisation techniques** are essential to construct better models in ML applications eg by **identifying correlated, redundant, or irrelevant features**

Traditional data visualisation methods are not practical when the datasets involve a large number of features (such as images) and we need to project the data onto a lower-dimensional space, called the **latent space**, using **dimensional reduction**

This is easier said than done: many pitfalls associated to **high-dimensionality datasets**

*“the curse of dimensionality”*

# Dimensional reduction

Efficient **data visualisation techniques** are essential to construct better models in ML applications eg by **identifying correlated, redundant, or irrelevant features**

Traditional data visualisation methods are not practical when the datasets involve a large number of features (such as images) and we need to project the data onto a lower-dimensional space, called the **latent space**, using **dimensional reduction**

This is easier said than done: many pitfalls associated to **high-dimensionality datasets**

📌 *High-dimensional data lives near the edge of the sample space*

# Dimensional reduction

Efficient **data visualisation techniques** are essential to construct better models in ML applications eg by **identifying correlated, redundant, or irrelevant features**

Traditional data visualisation methods are not practical when the datasets involve a large number of features (such as images) and we need to project the data onto a lower-dimensional space, called the **latent space**, using **dimensional reduction**

This is easier said than done: many pitfalls associated to **high-dimensionality datasets**

💡 *High-dimensional data lives near the edge of the sample space*

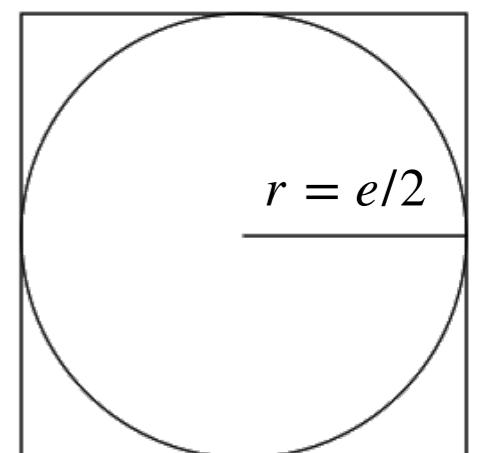
consider data distributed at random in a  $D$ -dimensional hypercube  $C = [-e/2, e/2]^D$

consider a  $D$ -dimensional sphere  $S$  of radius  $e/2$  centered at origin

probability that random point from  $C$  is sampled inside the sphere  $S$  is

$$P(x_i \in S) \simeq \frac{\pi^{D/2} (e/2)^D / \Gamma(D/2 + 1)}{e^D} = \simeq \frac{\pi^{D/2}}{2^D D^D} \rightarrow 0$$

so most of the data lies close to the hypercube **edge**!



# Dimensional reduction

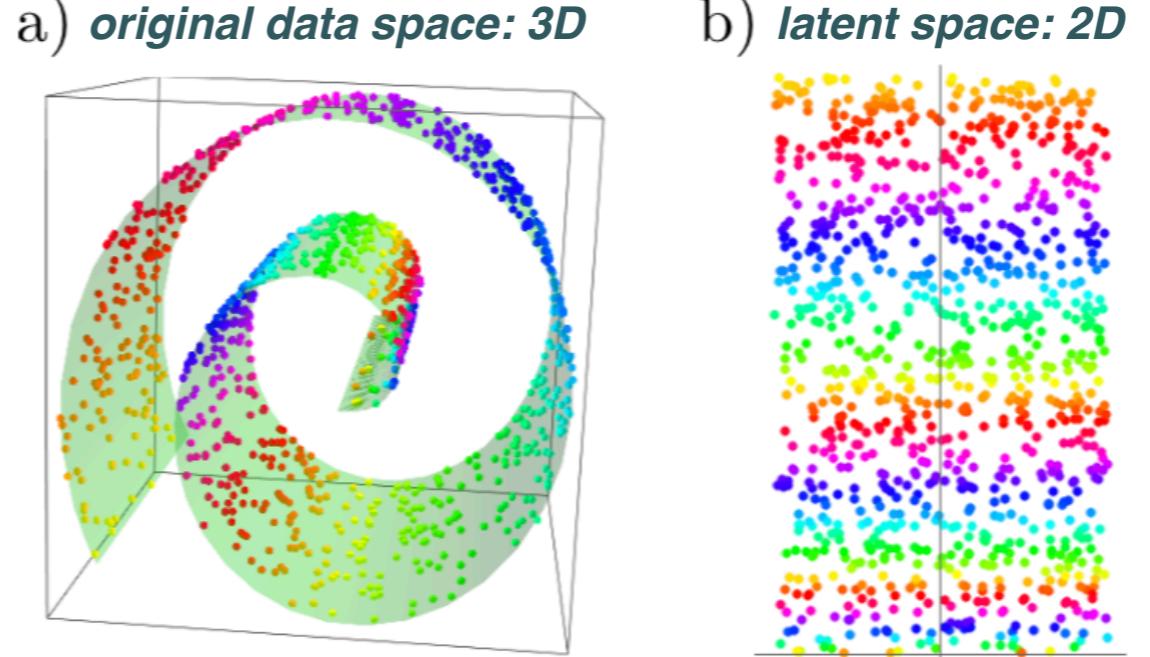
Efficient **data visualisation techniques** are essential to construct better models in ML applications eg by **identifying correlated, redundant, or irrelevant features**

Traditional data visualisation methods are not practical when the datasets involve a large number of features (such as images) and we need to project the data onto a lower-dimensional space, called the **latent space**, using **dimensional reduction**

This is easier said than done: many pitfalls associated to high-dimensionality datasets

- *High-dimensional data lives near the edge of the sample space*
- *We need to conserve information on original pair-wise distances or similarities when transforming to the latent space*

the minimum number of parameters needed to capture the original patterns is the **intrinsic dimensionality of the data**



# Dimensional reduction

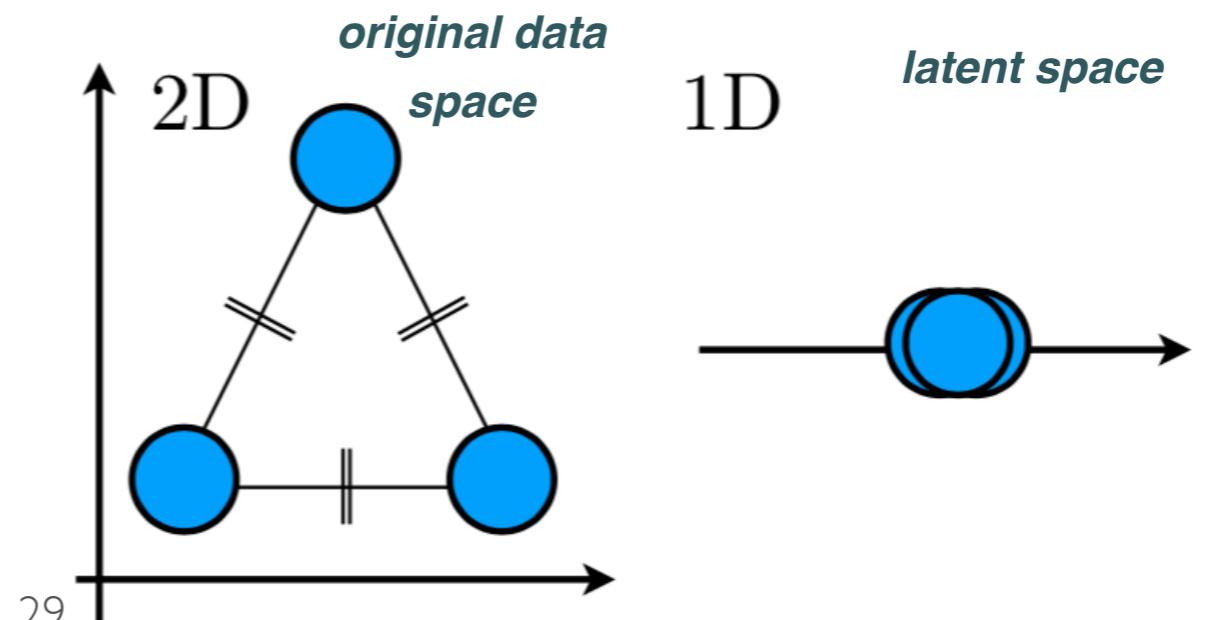
Efficient **data visualisation techniques** are essential to construct better models in ML applications eg by identifying correlated, redundant, or irrelevant features

Traditional data visualisation methods are not practical when the datasets involve a large number of features (such as images) and we need to project the data onto a lower-dimensional space, called the **latent space**, using **dimensional reduction**

This is easier said than done: many pitfalls associated to high-dimensionality datasets

- ➊ *High-dimensional data lives near the edge of the sample space*
- ➋ *We need to conserve information on original pair-wise distances or similarities when transforming to the latent space*
- ➌ *Dimensional reduction cannot be such to destroy info on original patterns in the data*

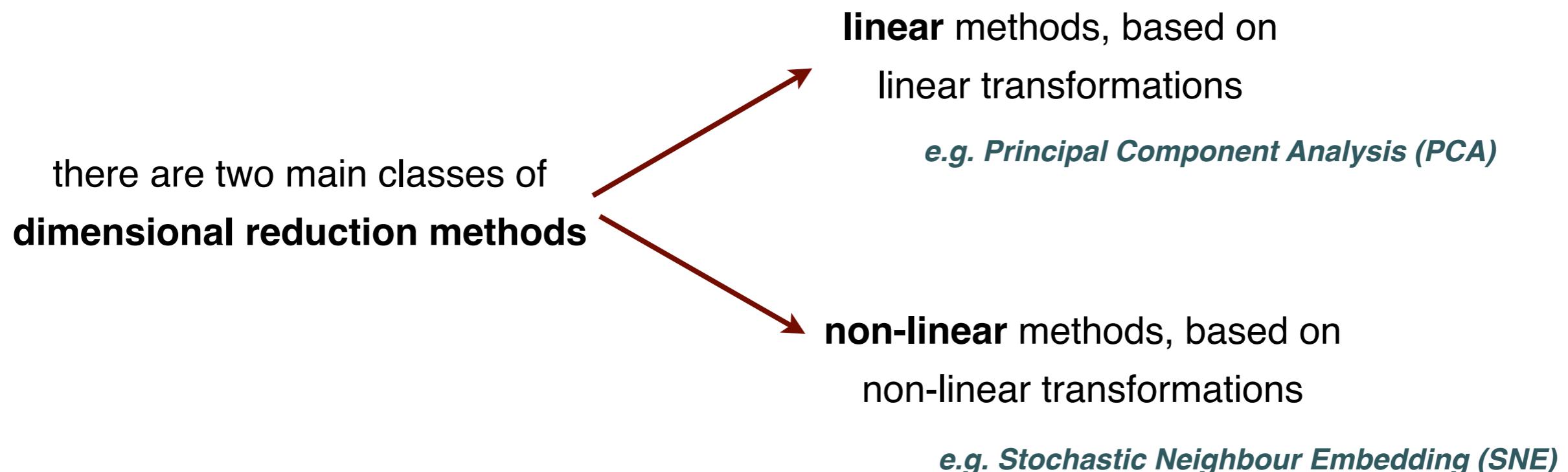
*“the crowding problem”*



# Dimensional reduction

Efficient **data visualisation techniques** are essential to construct better models in ML applications eg by identifying correlated, redundant, or irrelevant features

Traditional data visualisation methods are not practical when the datasets involve a large number of features (such as images) and we need to project the data onto a lower-dimensional space, called the **latent space**, using **dimensional reduction**



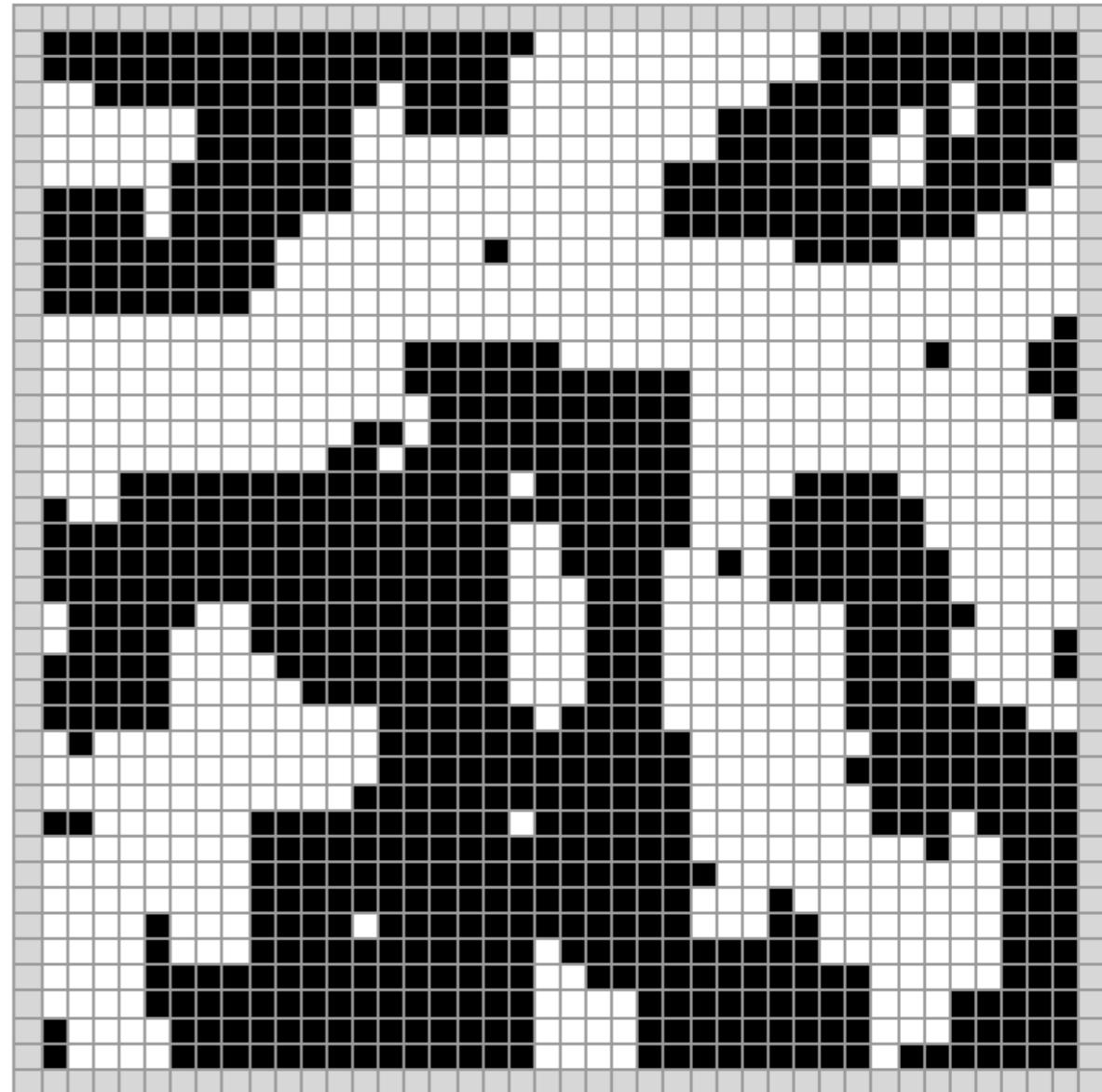
# Principal Component Analysis

PCA projections often capture the **large-scale structure** of high-dimensional datasets  
consider for example the **Ising Model in 2D** with 40 spins: 1600-dimensional space  
can we **measure ``order''** with few parameters?

*disordered (random) phase*



*ordered phase*

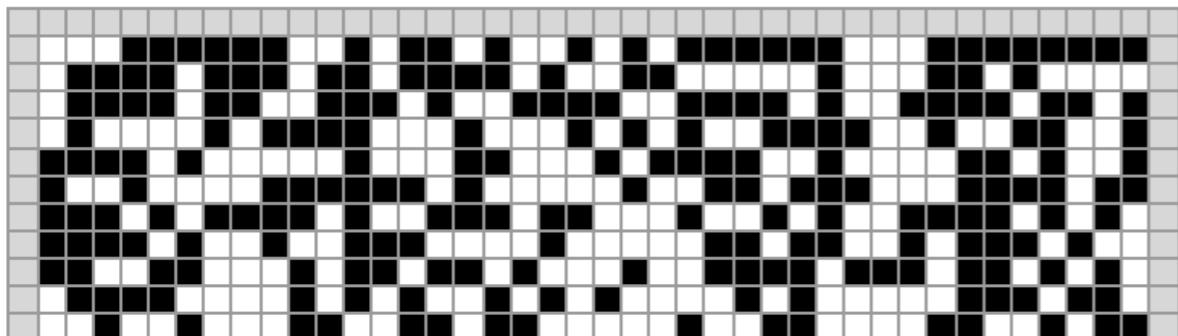


<https://demonstrations.wolfram.com/The2DIzingModelMonteCarloSimulationUsingTheMetropolisAlgorithm/>

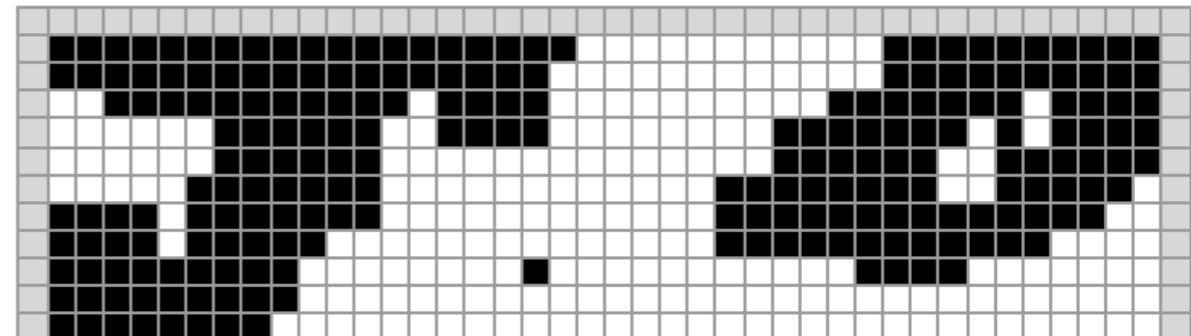
# Principal Component Analysis

PCA projections often capture the **large-scale structure** of high-dimensional datasets  
consider for example the **Ising Model in 2D** with 40 spins: 1600-dimensional space  
can we **measure ``order''** with few parameters?

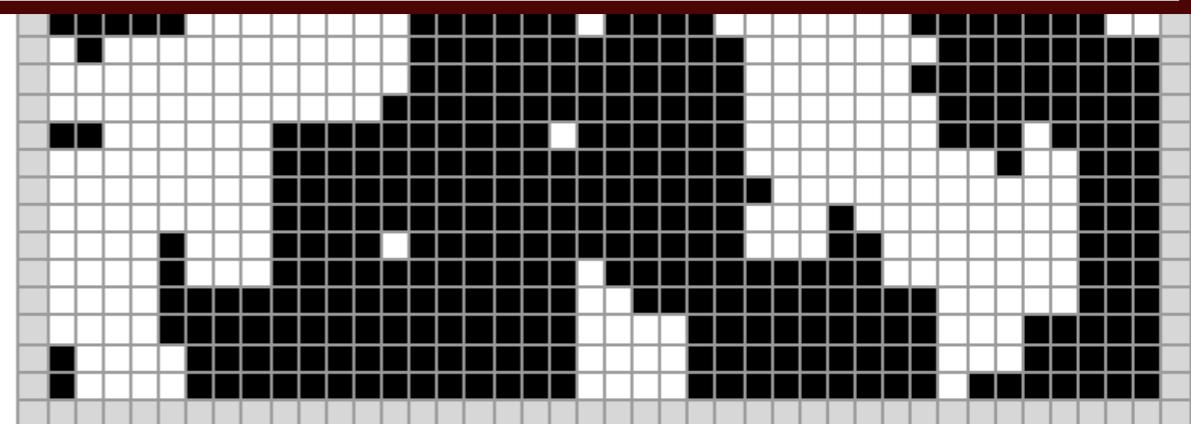
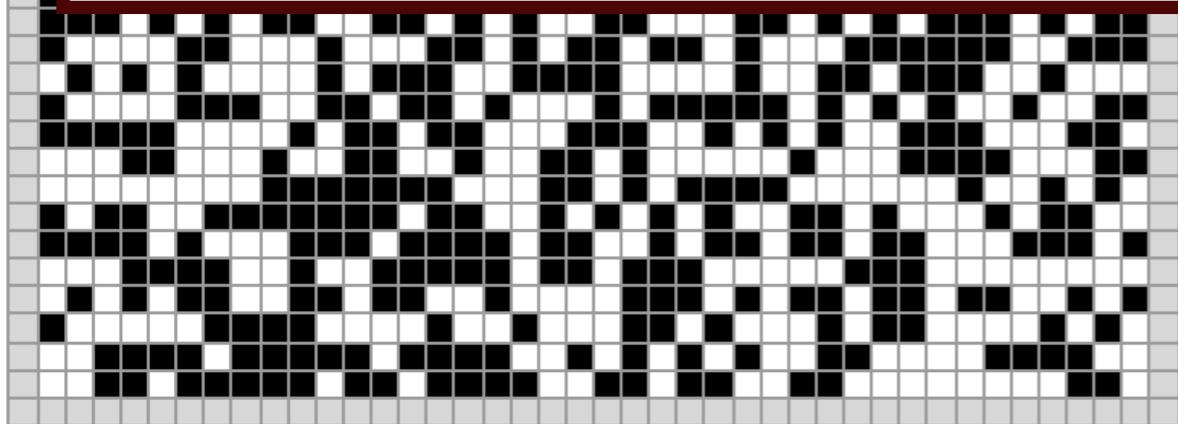
*disordered (random) phase*



*ordered phase*



how many parameters are needed to characterise the state of this system?  
Is this number (the **intrinsic dimensionality**) less than the original 1600?



<https://demonstrations.wolfram.com/The2DIzingModelMonteCarloSimulationUsingTheMetropolisAlgorit/>

# Principal Component Analysis

consider  $n$  data points that live in a  $p$ -dimensional feature space

$$\{\boldsymbol{x}_i\}_{i=1}^n \quad \boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,p})$$

assume for simplicity that the mean of these points vanishes

$$\bar{\boldsymbol{x}} = \frac{1}{n} \sum_{i=1}^n \boldsymbol{x}_i = 0$$

Now denote the **design matrix** as

$$\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^T$$

*In a design matrix, each row represents an individual data point and each column one of the data features*

where rows are the data points and columns are features

$$\boldsymbol{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \dots & \dots & \dots & \dots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{pmatrix} \xrightarrow{\text{data points}}$$

  
**features**

# Principal Component Analysis

the associated (symmetric)  $p$ -dimensional **covariance matrix** is then

$$\Sigma(X) = \frac{1}{n-1} X^T X \quad \rightarrow \quad \Sigma_{lm} = \frac{1}{n-1} \sum_{i=1}^n X_{li} X_{im}$$

*p-dimensional  
in space of features*      *sum over data points*

from where we see that  $\Sigma_{lm}$  measures the correlation between **features  $l$  and  $m$**

The goal of PCA is to rotate this matrix to a **new feature-basis** (different from the one present in the original data) that emphasises high-variability directions. This can be done by a linear transformation that **reduces the covariance between features**

recall the **eigenvalue decomposition** for a square matrix

$$A = Q \Lambda Q^T$$

Diagram illustrating the eigenvalue decomposition of a matrix  $A$ :

- Red arrows point to the columns of  $Q$ , labeled "columns are eigenvectors of  $A$ ".
- A red arrow points to the diagonal matrix  $\Lambda$ , labeled "diagonal matrix of eigenvalues".

# Principal Component Analysis

the associated (symmetric)  $p$ -dimensional **covariance matrix** is then

$$\Sigma(X) = \frac{1}{n-1} X^T X \quad \rightarrow \quad \Sigma_{lm} = \frac{1}{n-1} \sum_{i=1}^n X_{li} X_{im}$$

*p-dimensional  
in space of features*

*sum over data points*

from where we see that  $\Sigma_{lm}$  measures the correlation between **features  $l$  and  $m$**

The goal of PCA is to rotate this matrix to a **new feature-basis** (different from the one present in the original data) that emphasises high-variability directions. This can be done by a linear transformation that **reduces the covariance between features**

for this we will use **Singular Value Decomposition** (SVD), a factorisation of a real or complex matrix that generalizes the eigendecomposition of a positive semidefinite matrix

$$X = U S V^T$$

Diagram illustrating the Singular Value Decomposition (SVD) of a matrix  $X$ :

- Red arrows point to the components:
  - Left singular vectors of  $X$  (columns of  $U$ )
  - Diagonal matrix of singular values
  - Right singular vectors of  $X$  (columns of  $V$ )
- Text labels provide additional context:
  - left (right) singular vectors of  $X$ : orthonormal eigenvectors of  $XX^*$  ( $X^*X$ )
  - columns are left singular vectors of  $X$
  - diagonal matrix of singular values
  - columns are right singular vectors of  $X$

# Principal Component Analysis

Using SVD we can express the data covariance matrix as

$$\Sigma(X) = \frac{1}{n-1} VSU^T USV^T = V \left( \frac{S^2}{n-1} \right) V^T \equiv V \Lambda V^T$$

*U,V are unitary matrices*      *diagonal matrix with eigenvalues in decreasing order along diagonal*

columns of  $V$  → principal directions of  $\Sigma$

at this point we are ready to use PCA to reduce the dimensionality of data from  $p$  to  $p' < p$

$$\widetilde{Y} = X \widetilde{V}_{p'}$$

*reduced dataset: n points with  $p' < p$  features each in the rotated-feature matrix*      *original dataset: n points with  $p$  features each*      *projection matrix: select the  $p'$  largest eigenvectors*

with PCA only the  $p'$  directions with higher variability remain

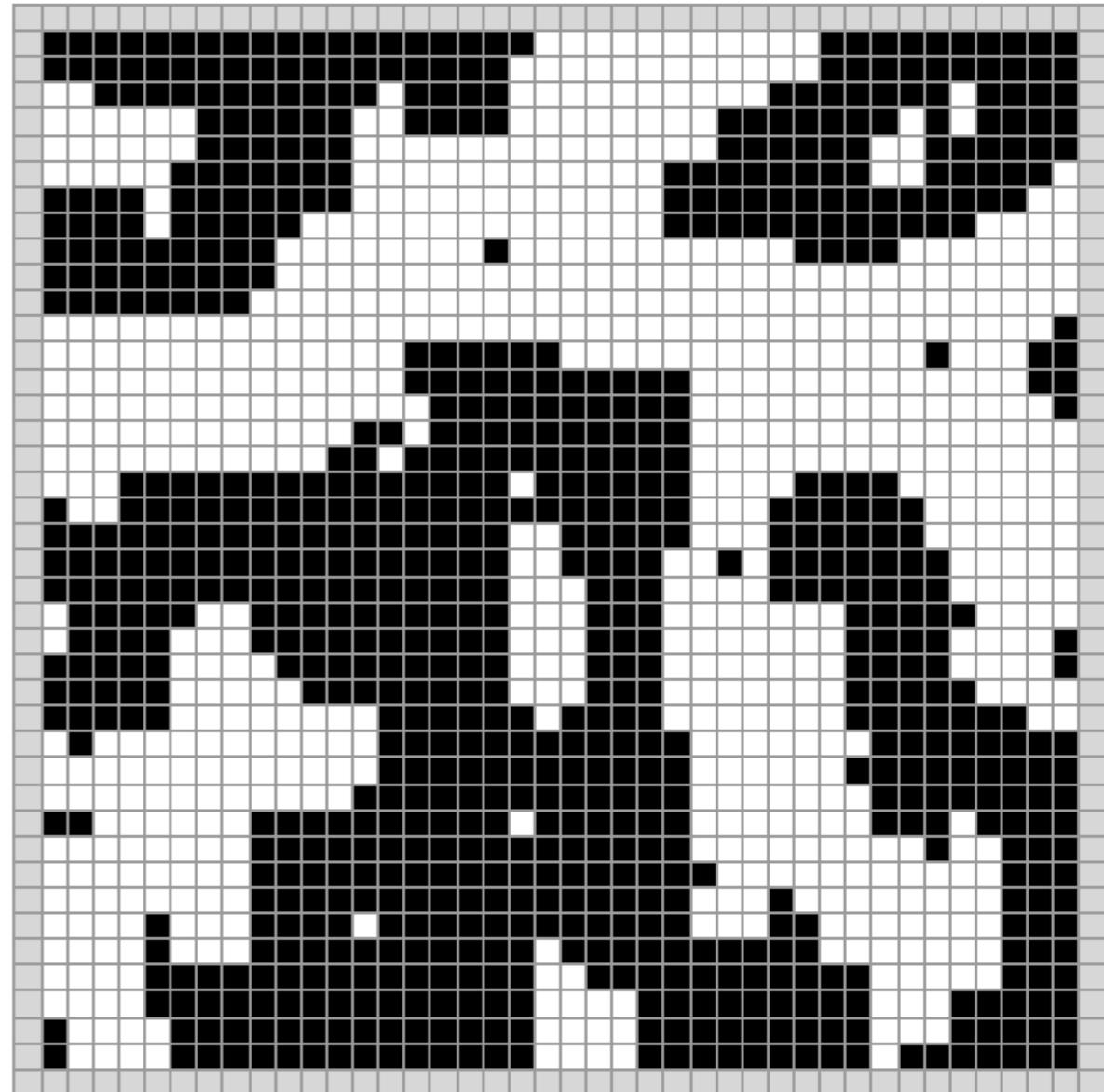
# Principal Component Analysis

PCA projections often capture the **large-scale structure** of high-dimensional datasets  
consider for example the **Ising Model in 2D** with 40 spins: 1600-dimensional space  
can we **measure ``order''** with few parameters?

*disordered (random) phase*



*ordered phase*

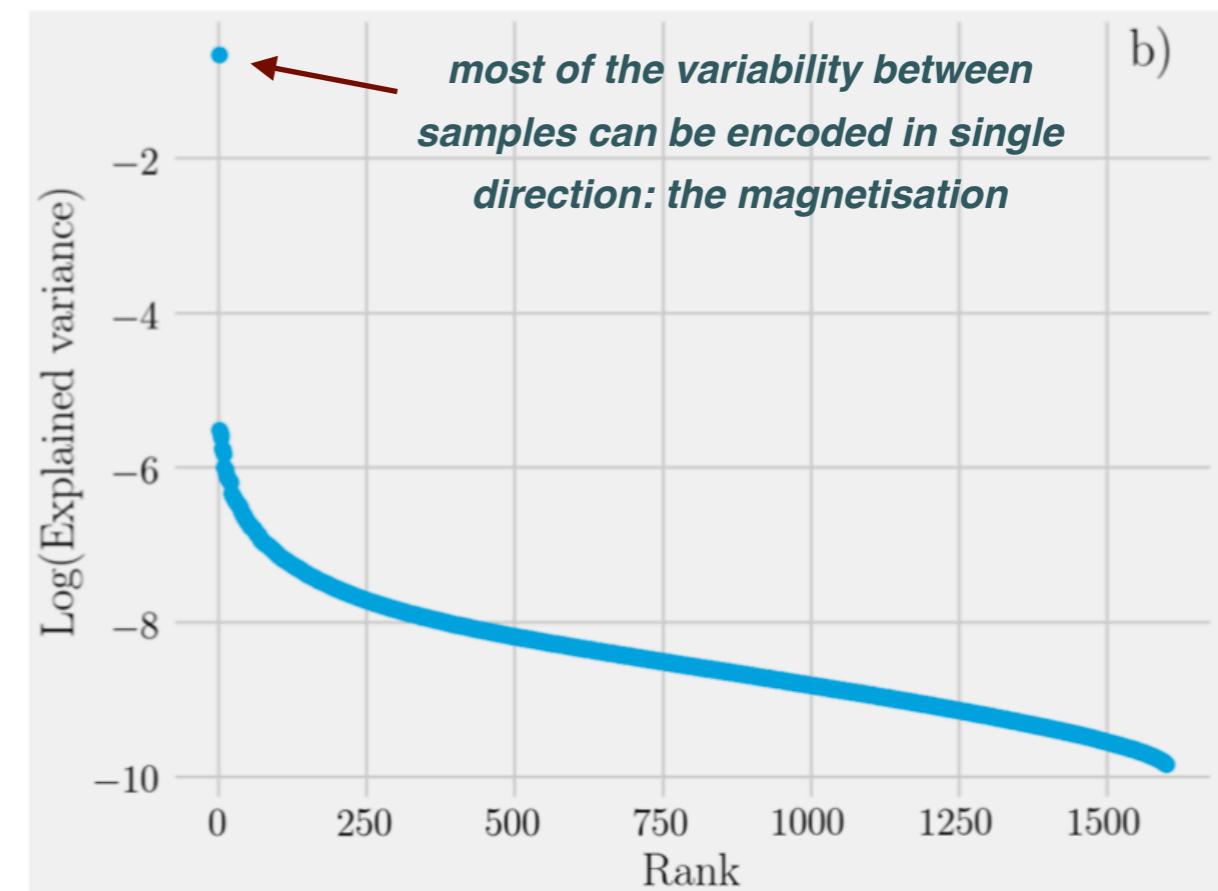
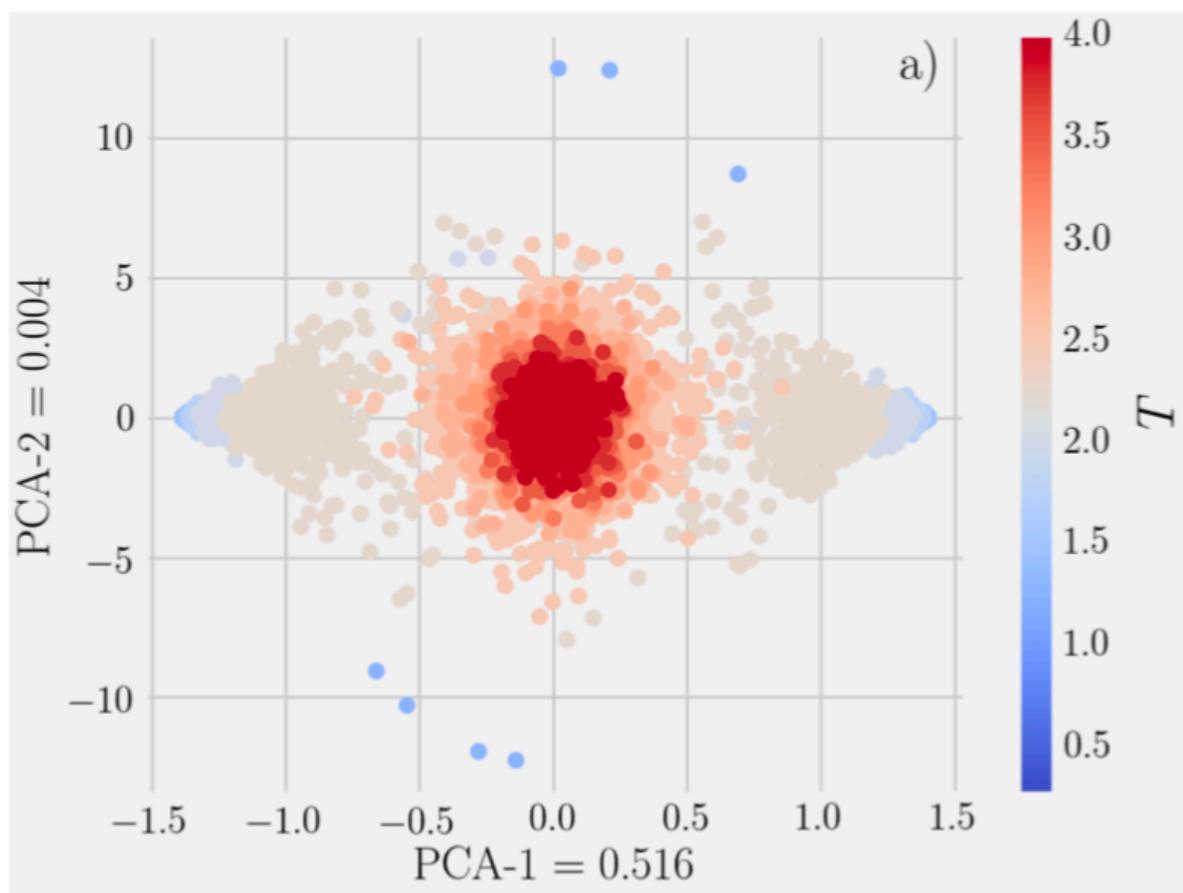


<https://demonstrations.wolfram.com/The2DIzingModelMonteCarloSimulationUsingTheMetropolisAlgorithm/>

# Principal Component Analysis

PCA projections often capture the **large-scale structure** of high-dimensional datasets  
consider for example the **Ising Model in 2D** with 40 spins: 1600-dimensional space  
can we **measure ``order''** with few parameters?

*1000 samples for each T*

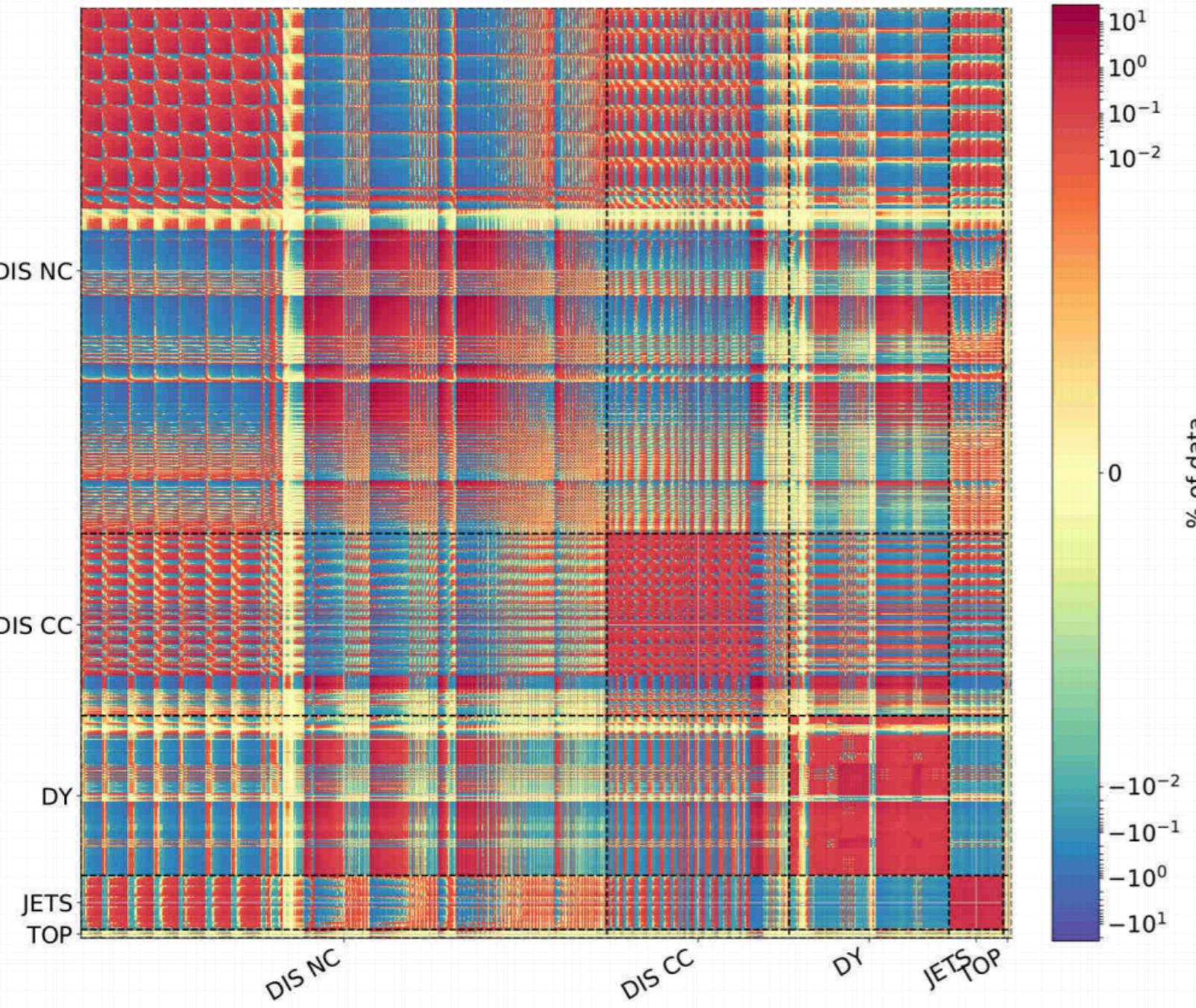


*The first principal component accounts for > 50% of total variability!*

This first PCA component corresponds to the **magnetisation order parameter**, which we have thus identified without any prior physical knowledge of the system

# PCA for propagation of theory errors

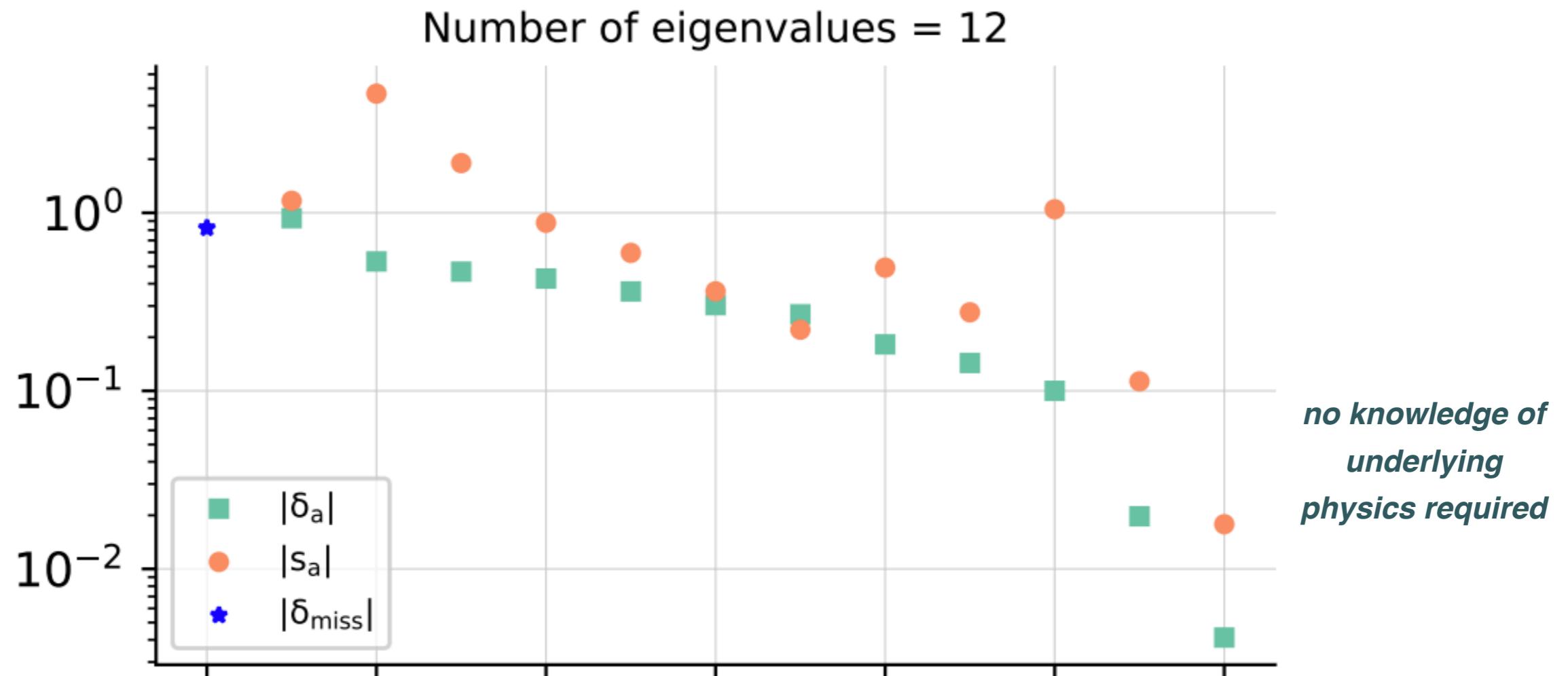
assume that some gives you the **covariance matrix of theory errors** of a global PDF fit (3000-dimensional space!). What are the **relevant components?**



Process Type	Dataset	Reference	$N_{\text{dat}}$	$N_{\text{dat}} (\text{total})$
DIS NC	NMC	[28, 29]	134	1593
	SLAC	[30]	12	
	BCDMS	[31, 32]	530	
	HERA $\sigma_{NC}^p$	[36]	886	
	HERA $\sigma_{NC}^c$	[37]	31	
DIS CC	NuTeV dimuon	[33, 34]	41	552
	CHORUS	[35]	430	
	HERA $\sigma_{CC}^p$	[36]	81	
DY	ATLAS $W, Z$ , 7 TeV 2010	[42]	30	484
	ATLAS $W, Z$ , 7 TeV 2011	[43]	34	
	ATLAS low-mass DY 2011	[44]	4	
	ATLAS high-mass DY 2011	[45]	5	
	ATLAS $Z$ $p_T$ 8 TeV ( $p_T^{ll}, M_{ll}$ )	[46]	44	
	ATLAS $Z$ $p_T$ 8 TeV ( $p_T^{ll}, y_Z$ )	[46]	48	
	CMS Drell-Yan 2D 2011	[51]	88	
	CMS $W$ asy 840 pb	[52]	11	
	CMS $W$ asy 4.7 pb	[53]	11	
	CMS $W$ rap 8 TeV	[54]	22	
	CMS $Z$ $p_T$ 8 TeV ( $p_T^{ll}, M_{ll}$ )	[55]	28	
	LHCb $Z$ 940 pb	[60]	9	
	LHCb $Z \rightarrow ee$ 2 fb	[61]	17	
	LHCb $W, Z \rightarrow \mu$ 7 TeV	[62]	29	
	LHCb $W, Z \rightarrow \mu$ 8 TeV	[63]	30	
	CDF $Z$ rap	[38]	29	
JET	D0 $Z$ rap	[39]	28	164
	D0 $W \rightarrow e\nu$ asy	[40]	8	
	D0 $W \rightarrow \mu\nu$ asy	[41]	9	
	ATLAS jets 2011 7 TeV	[47]	31	
TOP	CMS jets 7 TeV 2011	[56]	133	
	ATLAS $\sigma_{tt}^{\text{top}}$	[48, 49]	3	26
	ATLAS $t\bar{t}$ rap	[50]	10	
	CMS $\sigma_{tt}^{\text{top}}$	[57, 58]	3	
Total	CMS $t\bar{t}$ rap	[59]	10	
			2819	2819

# PCA for propagation of theory errors

assume that some gives you the **covariance matrix of theory errors** of a global PDF fit (3000-dimensional space!). What are the **relevant components**?



only **O(10)** directions encode all the variability of **3000-dimensional covmat**  
physical reason: theory errors are **highly correlated** among processes

# t-SNE

- In dimensional reduction and data visualisation techniques, it is often desirable to **preserve local structures** in high-dimensional datasets
- In many cases **non-linear methods** (unlike PCA, which is linear) are required
- One of these is **t-stochastic neighbour embedding** (t-SNE), where each high-dimensional training point is mapped to low-dimensional embedding coordinates optimized to preserve the local structures in the data

the main idea is to **associate a probability distribution** to the neighbour of each data point

$$p_{i|j} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma_i\right)}{\sum_{k \neq i} \exp\left(-||x_i - x_k||^2/2\sigma_i\right)} \quad x_i \in \mathbb{R}^p$$

**Data Space**

*probability that j  
is neighbour of i*

*model parameter*

The diagram shows the t-SNE probability formula. A red arrow points from the term  $p_{i|j}$  to the left side of the equation. Another red arrow points from the term  $\sigma_i$  to the right side of the equation, under the label "model parameter".

# t-SNE

the main idea is to **associate a probability distribution** to the neighbour of each data point

**Data Space**

$$p_{i|j} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma_i\right)}{\sum_{k \neq i} \exp\left(-||x_i - x_k||^2/2\sigma_i\right)} \quad x_i \in \mathbb{R}^p$$

and then **constructing a similar probability** in the **lower-dimensional latent space**

**Latent Space**

$$q_{i|j} = \frac{\left(1 + ||y_i - y_j||^2\right)^{-1}}{\sum_{k \neq i} \left(1 + ||y_i - y_k||^2\right)^{-1}} \quad y_i \in \mathbb{R}^{p'}, \quad p' < p$$

the latent space coordinates are determined by minimising the **Kullback-Leibler divergence** between the two probability distributions

**cost function:**  $C(Y) = D_{\text{KL}}(p || q) \equiv \sum_{ij} p_{i|j} \log \frac{p_{i|j}}{q_{i|j}}$

*minimisation using  
e.g. Gradient Descent  
output of minimisation:  
latent-space coordinates  $\{y_i\}$   
of each data point  $\{x_i\}$*

# The Kullback-Leibler divergence

The KL divergence, a measure of the similarity between two probability distributions  $p(\mathbf{x})$  and  $q(\mathbf{x})$  plays an important role in machine learning applications

$$D_{KL}(p \parallel q) = \int d\mathbf{x} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \quad D_{KL}(q \parallel p) = \int d\mathbf{x} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})}$$

which can be symmetrised to construct a **squared metric** (distance)

$$D_{JS}(p \parallel q) = \frac{1}{2} \left( D_{KL}\left(p \middle\| \frac{p+q}{2}\right) + D_{KL}\left(q \middle\| \frac{p+q}{2}\right) \right)$$

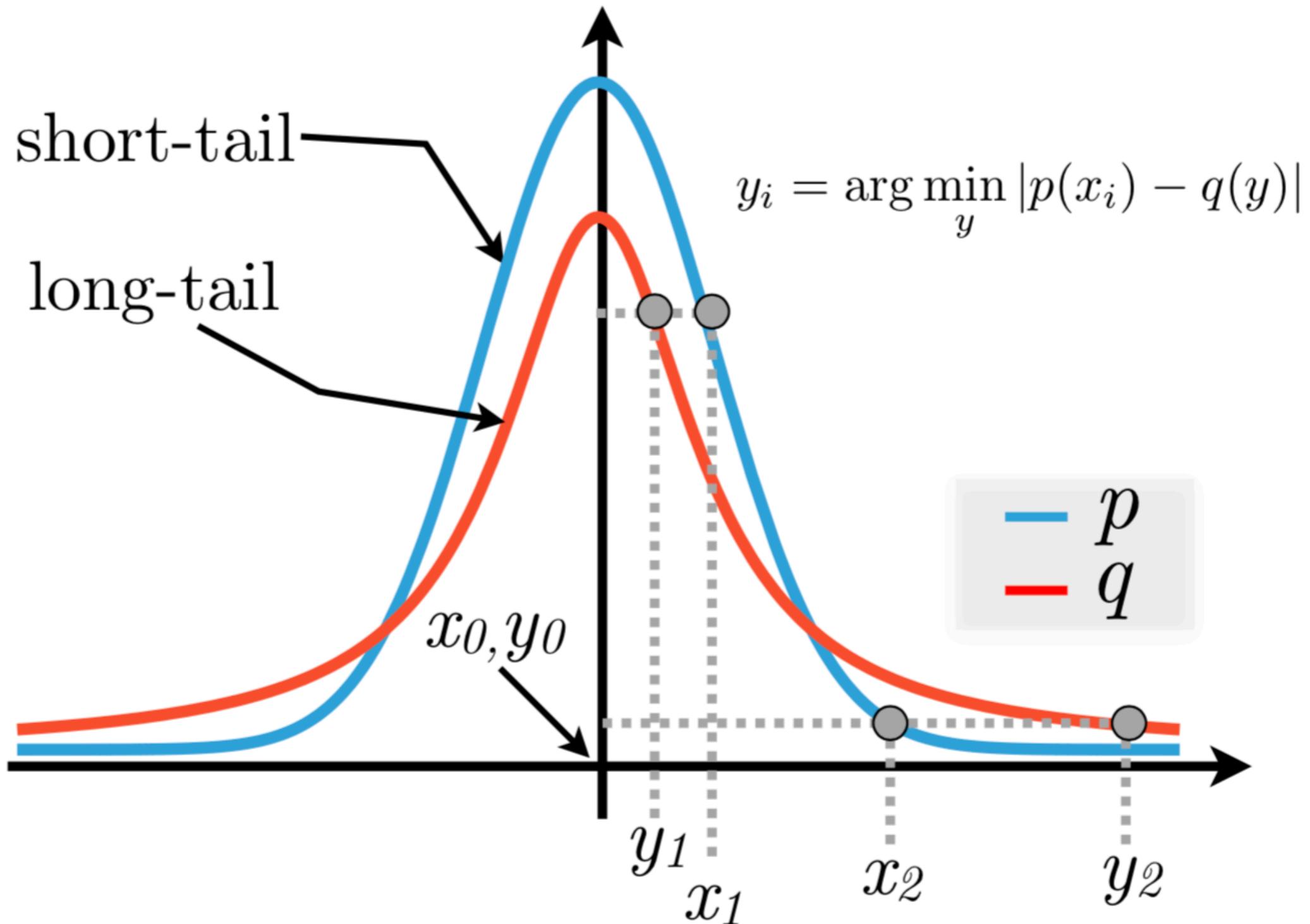
*Jensen-Shannon divergence*

The KL-divergence is **positive-definite**, and only vanishes when  $p(\mathbf{x})=q(\mathbf{x})$

$$D_{KL}(p \parallel q) \geq 0$$

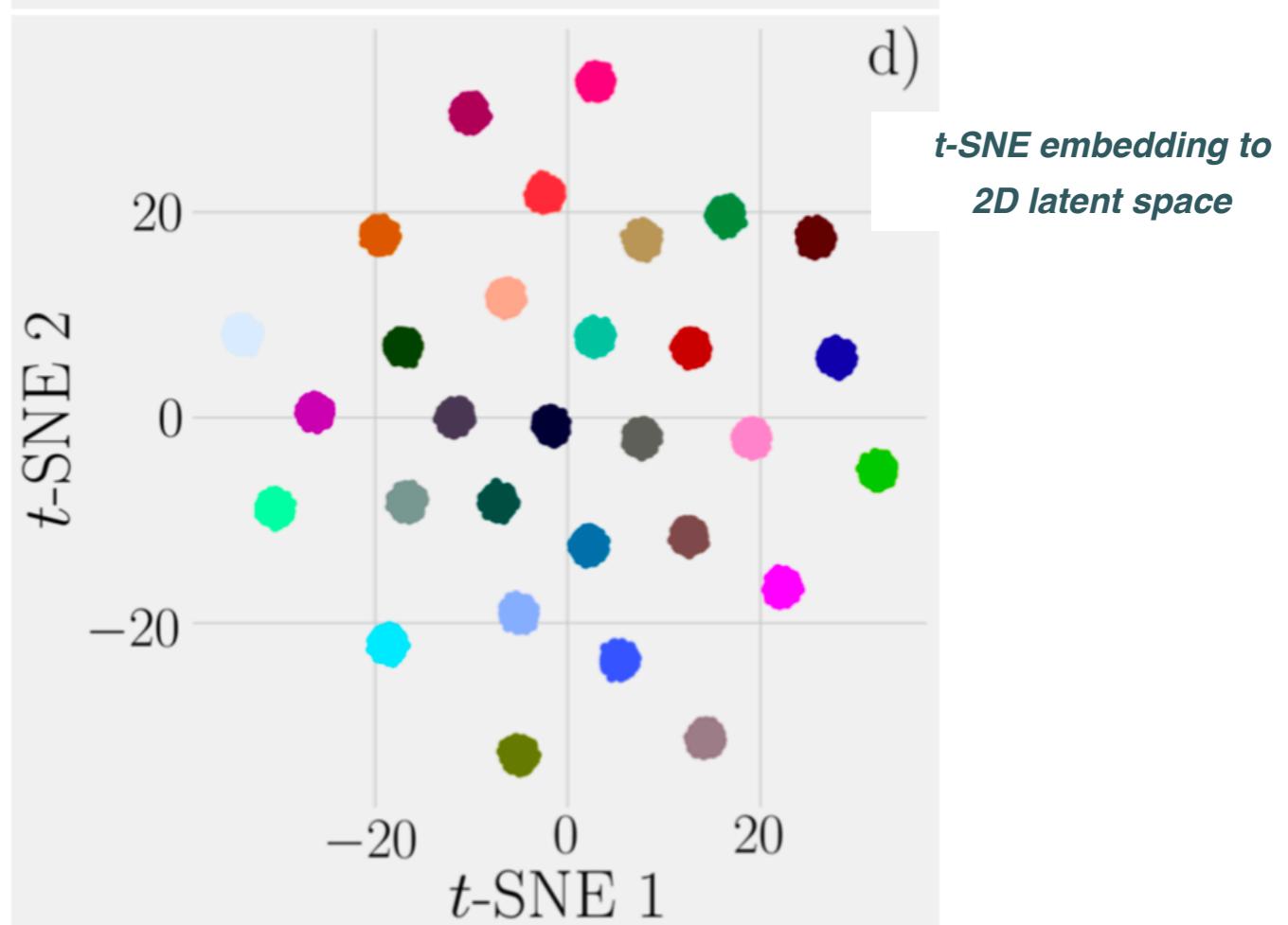
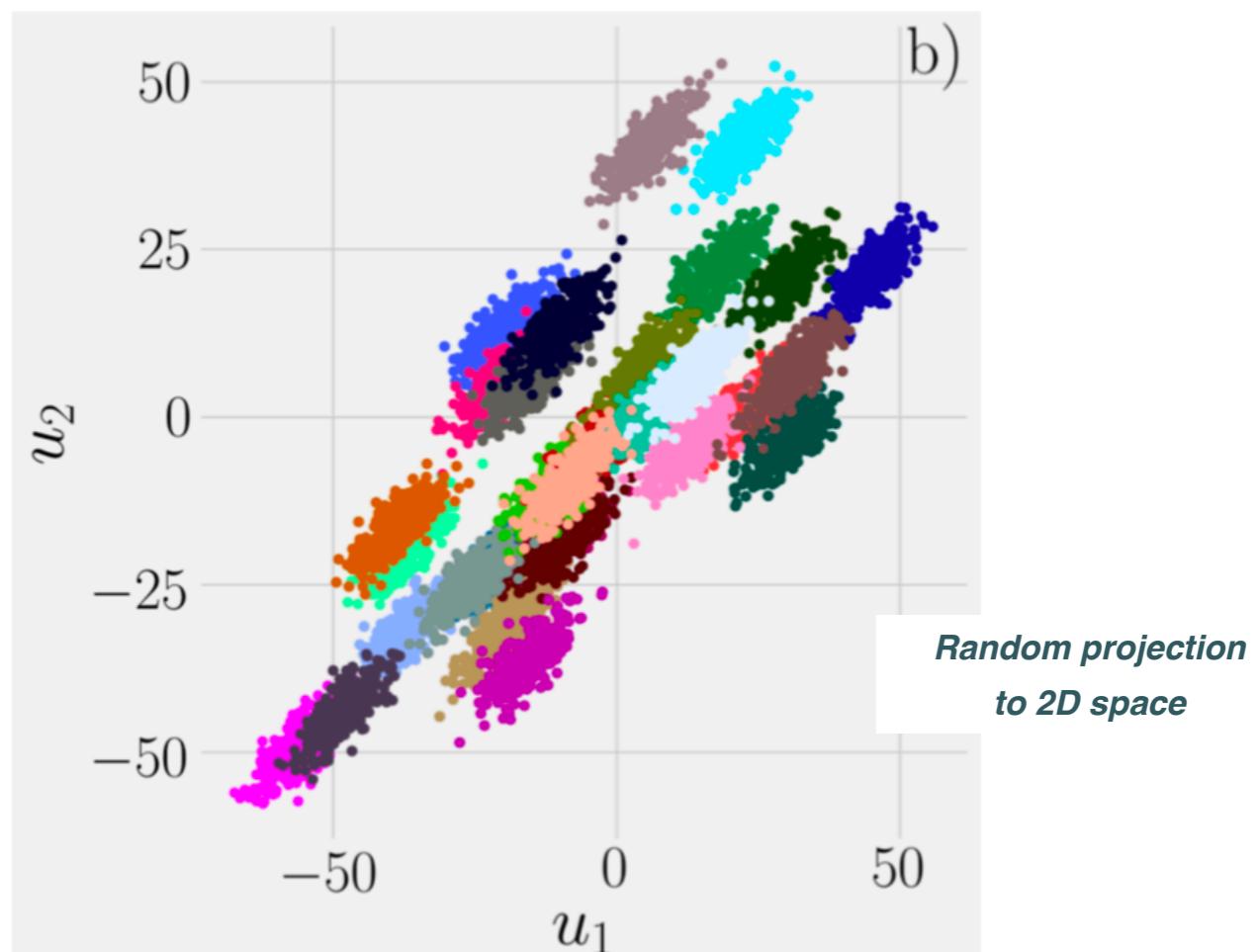
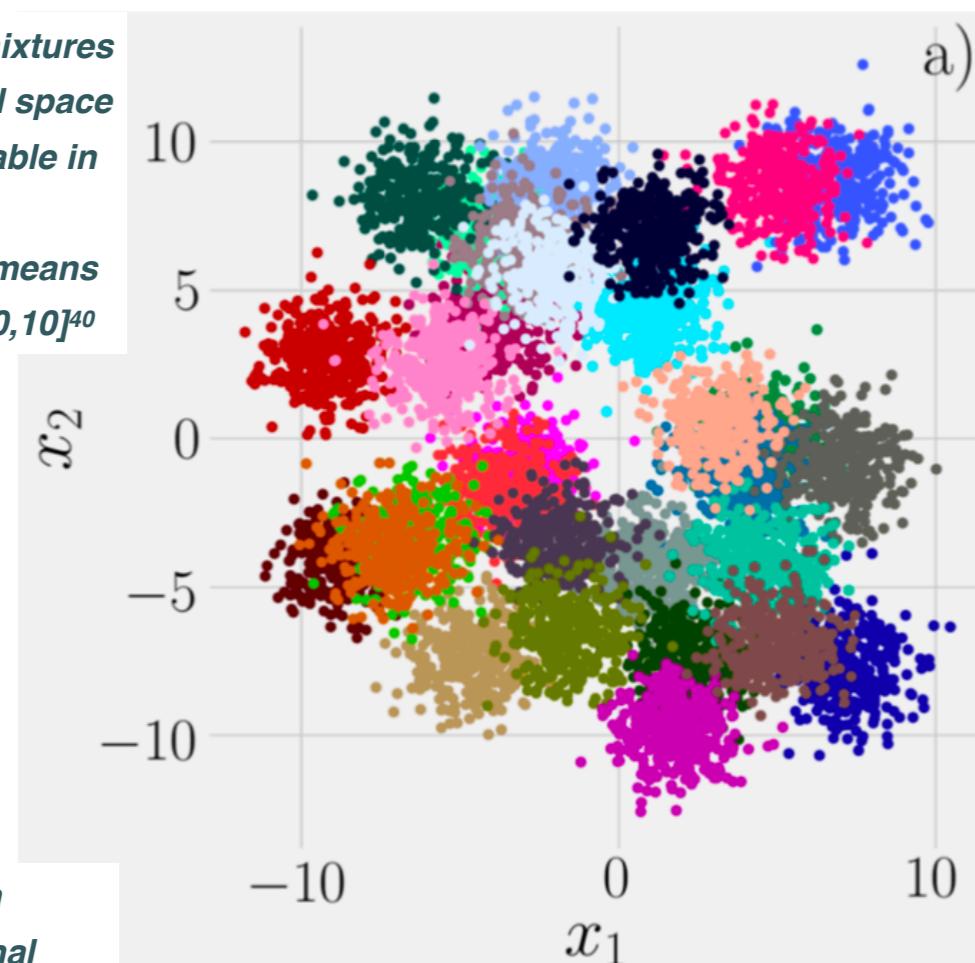
in general the integral cannot be computed and one needs to sample the two probability distributions by means of a suitable binning

# t-SNE



Note that  $q$  is a **long-tail distribution** (Cauchy): this preserves short distance information while repelling two points that are far apart in the original space

**K=30 Gaussian mixtures  
in 40-dimensional space  
(labels not available in  
real case!)  
Same variance, means  
at random in  $[-10, 10]^{40}$**



# **Ensemble Methods & Bootstrapping**

# Combining models

a powerful strategy in machine learning is that of **ensemble methods** that **combine predictions from multiple statistical models** to improve predictive performance

- 💡 Key aspect: assess the **degree of correlation** between the models of the ensemble
- 💡 The reason is that **combining correlated models reduces the overall variance less** than in the uncorrelated case
- 💡 Also these correlations can **increase the bias of the combined model**, offsetting potential reductions in variance from ensemble averaging

# Combining models

a powerful strategy in machine learning is that of **ensemble methods** that **combine predictions from multiple statistical models** to improve predictive performance

- Key aspect: assess the **degree of correlation** between the models of the ensemble
  - The reason is that **combining correlated models reduces the overall variance less** than in the uncorrelated case
  - Also these correlations can **increase the bias of the combined model**, offsetting potential reductions in variance from ensemble averaging
- e. g. assume that your model predicts  $X$ , and you have  $n$  models. If each of these models has variance  $\sigma$ , then the variance of their sum is

$$\text{Var} \left( \sum_{i=1}^n X_i \right) = \sum_{i=1}^n \text{Var}(X_i) + 2 \sum_{1 \leq i \leq j \leq n} \text{Cov}(X_i, X_j)$$

$$\text{Var}(\bar{X}) = \text{Var} \left( \frac{1}{n} \sum_{i=1}^n X_i \right) = \frac{1}{n^2} \sum_{i=1}^n \text{Var}(X_i)$$

$$\text{Var}(\bar{X}) = \frac{\sigma^2}{n} + \frac{n-1}{n} \rho \sigma^2$$

*reduction as  $n$  increased*

*correlations increase variance*

# Combining models

a powerful strategy in machine learning is that of **ensemble methods** that **combine predictions from multiple statistical models** to improve predictive performance

- 💡 Key aspect: assess the **degree of correlation** between the models of the ensemble
- 💡 The reason is that **combining correlated models reduces the overall variance less** than in the uncorrelated case
- 💡 Also these correlations can **increase the bias of the combined model**, offsetting potential reductions in variance from ensemble averaging
  - e. g. assume that your model predicts  $X$ , and you have  $n$  models. If each of these models has variance  $\sigma$ , then the variance of their sum is

$$\text{Var}(\bar{X}) = \frac{\sigma^2}{n} + \frac{n-1}{n}\rho\sigma^2$$

*for fully correlated models*   $\text{Var}(\bar{X}) = \frac{\sigma^2}{n} + \frac{n-1}{n}\sigma^2 \rightarrow \sigma^2$

*no reduction of variance in the combination*

# Bagging

**Bootstrap AGGregation (Bagging) is a popular ensemble combination method**

we start from a large dataset that is **partitioned** into  $M$  smaller datasets:

$$\mathcal{L} \rightarrow \{\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_M\} \quad \sum_{m=1}^M \mathcal{L}_m = \mathcal{L}$$

so that each of the  $M$  datasets is large enough to train a predictor. The **aggregate predictor** is then constructed from those trained in each separate dataset

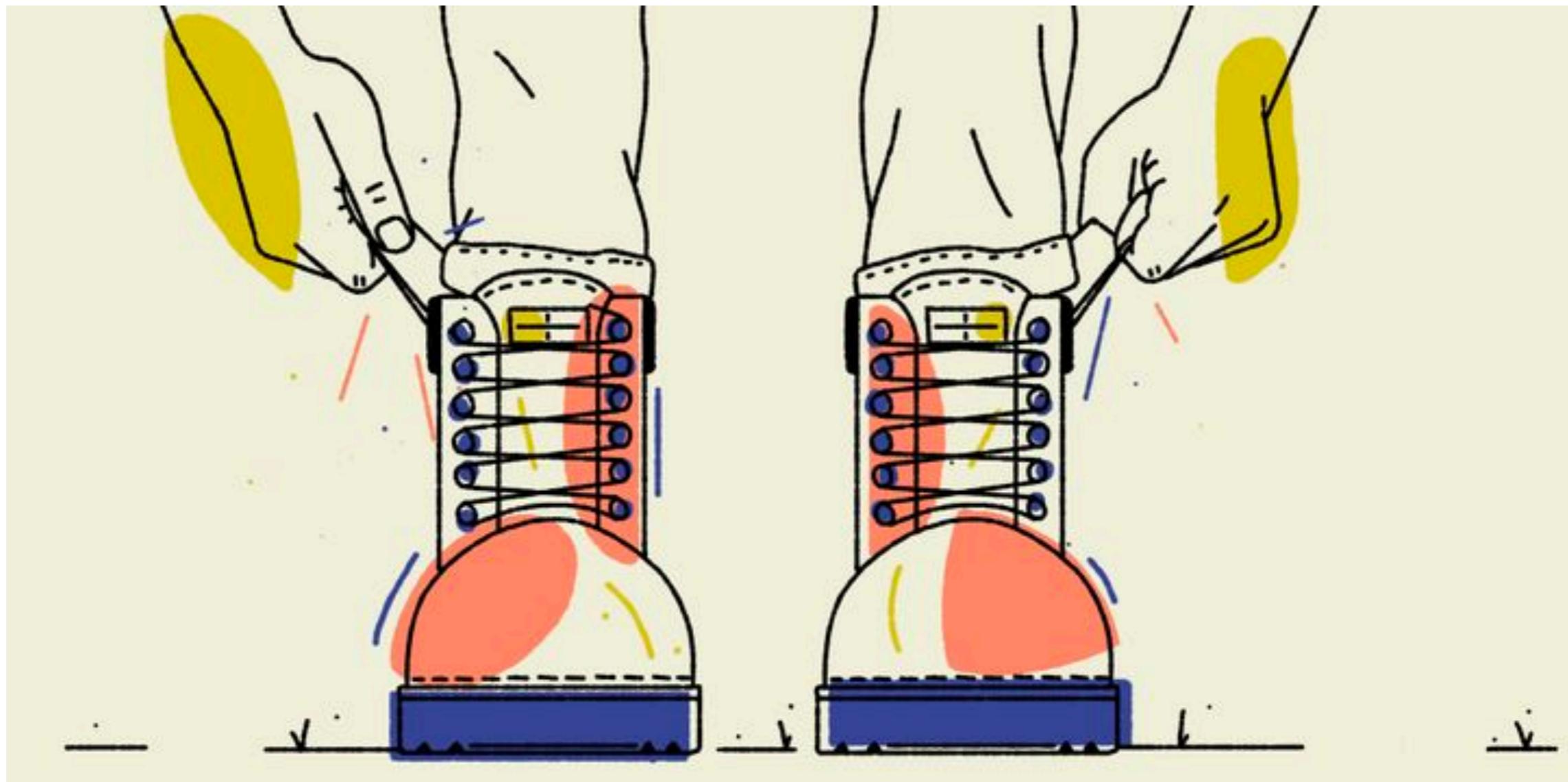
$$\hat{g}_{\mathcal{L}}^A(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M g_{\mathcal{L}_i}^A(\mathbf{x}) \quad \textit{for continuous datasets, analogous expressions for discrete classifiers}$$

such aggregation can **reduce the variance without increasing the bias**

But what should we do if we don't have a very large dataset? If each partitioned set one has few data points then the prediction will be poor ...

*and if using more data/examples is not possible?*

# Bootstrapping



Bootstrapping quantifies properties of an estimator by measuring those properties when **sampling from an approximating distribution** (e.g. a subset of the original data)

# Bootstrapping

- Assume we are given a training dataset and we want to compute e.g. confidence intervals

$$\mathcal{D} = \{X_1, \dots, X_n\}$$

- This can be done by **sampling  $n$  points with replacement** to get  **$B$  new datasets**

$$\begin{aligned}\mathcal{D}^{*(1)} &= \{X_1^{*(1)}, \dots, X_n^{*(1)}\} \\ &\vdots \\ \mathcal{D}^{*(B)} &= \{X_1^{*(B)}, \dots, X_n^{*(B)}\}\end{aligned}$$

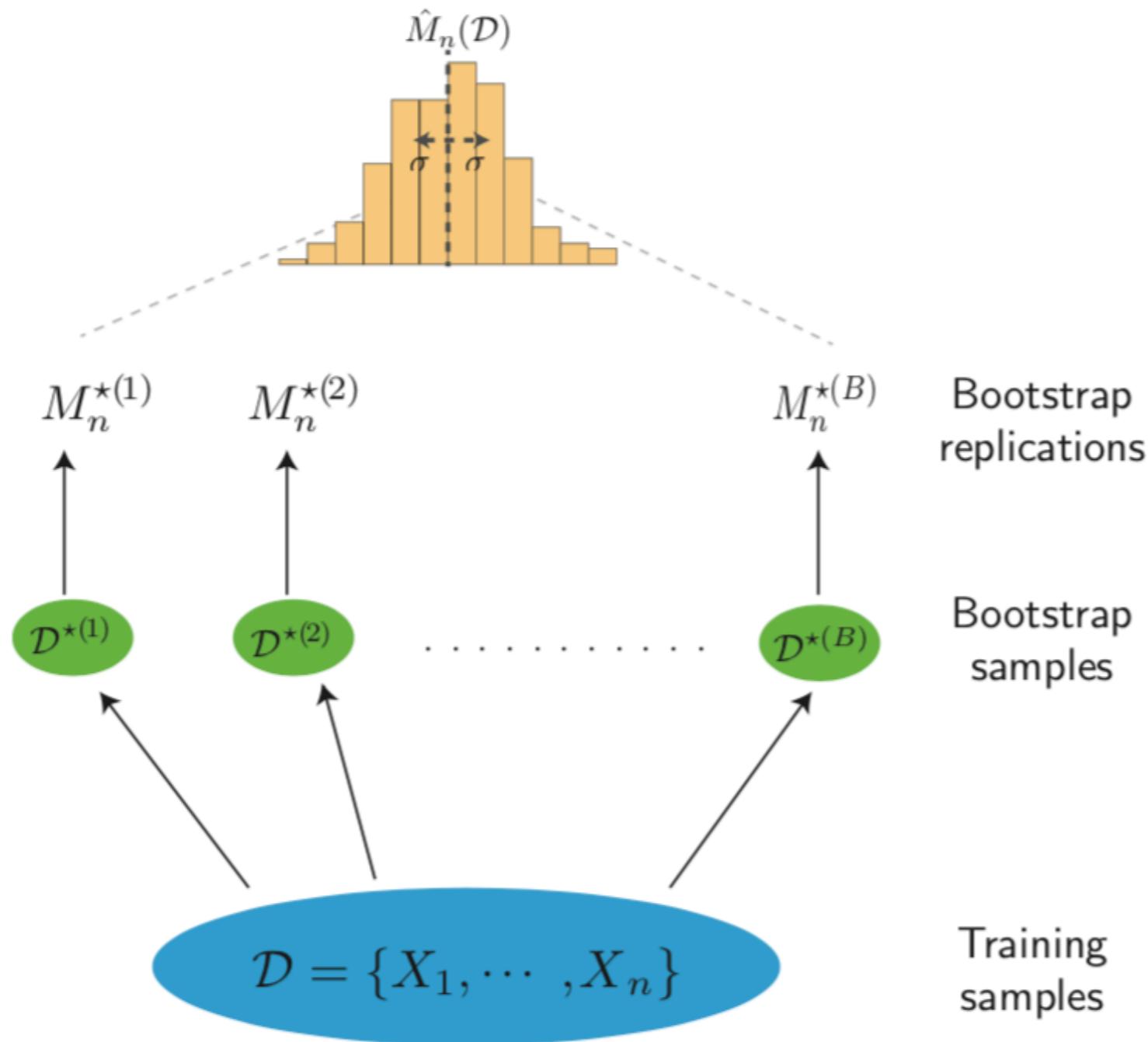
*bootstrap samples  
(with repeated elements)*

- With the **bootstrapped samples** we can construct statistical quantities of interest:

$$\widehat{\text{Var}}_B(M_n) = \frac{1}{B-1} \sum_{k=1}^B \left( M_n^{*(k)} - \bar{M}_n^* \right)^2 \quad \bar{M}_n^* = \frac{1}{B} \sum_{k=1}^B M_n^{*(k)}$$

it can be shown that in the large  $n$  limit the **bootstrap distributions** approximate well the **sampling distributions** from which the training dataset was obtained

# Bootstrapping



In ML applications, bootstrapping is frequently used to **assign measures of accuracy** (defined in terms of bias, variance, correlations etc to sample estimates

It can be shown that in the large  $n$  limit the **bootstrap distributions** approximate well the **sampling distributions** from which the training dataset was obtained

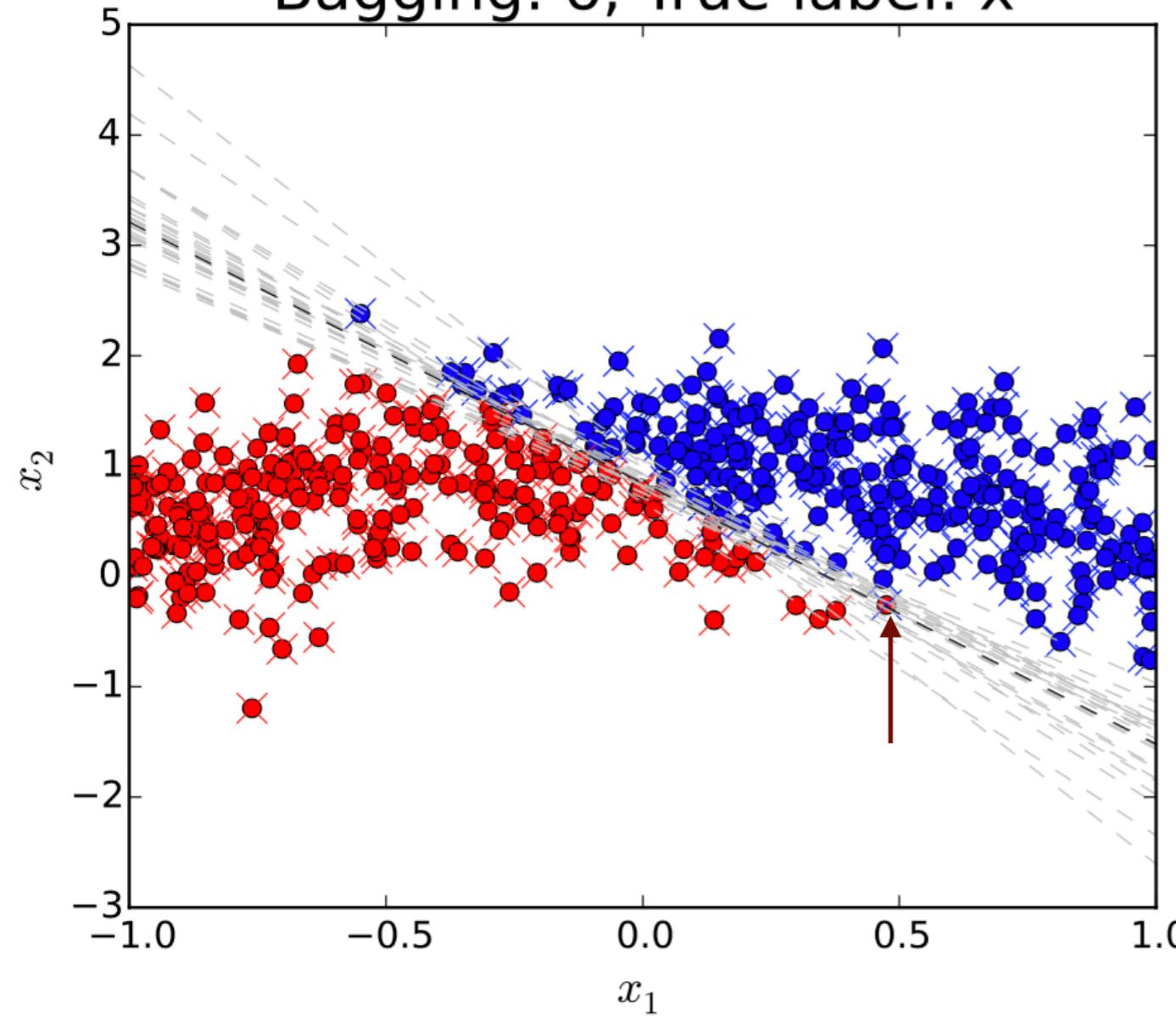
# Bagging with bootstrap

same as before, but now the datasets have been partitioned with bootstrapping

$$\hat{g}_{\mathcal{L}}^{\text{BS}}(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M g_{\mathcal{L}_i^{\text{BS}}}^A(\mathbf{x})$$

which can lead to a **variance reduction** to the price of an **increase in the bias**

Bagging: o, True label: x



ex: bagging with bootstrap (**2D classification**)

$n=500, B=25$  (50 points each)

grey dashed: bootstrap predictions

black dashed: bagging average

individual predictors poor, bagging much better!

bagging is specially useful for **unstable learning algorithms** where small changes in the training dataset result in large changes in the prediction