Machine Learning for Physics and Astronomy

Juan Rojo

VU Amsterdam & Theory group, Nikhef

Natuur- en Sterrenkunde BSc (Joint Degree), Honours Track

Lecture 4, 28/09/2020

The path so far

- **Supervised Learning** requires *i)* input dataset, *ii)* ML model, and *iii)* definition of a cost function / figure of merit

- Artificial NNs are suitable for **ML regression problems** in many dimensions

- Stochastic Gradient Descent combined with **Backpropagation** represents a powerful training strategy for deep networks

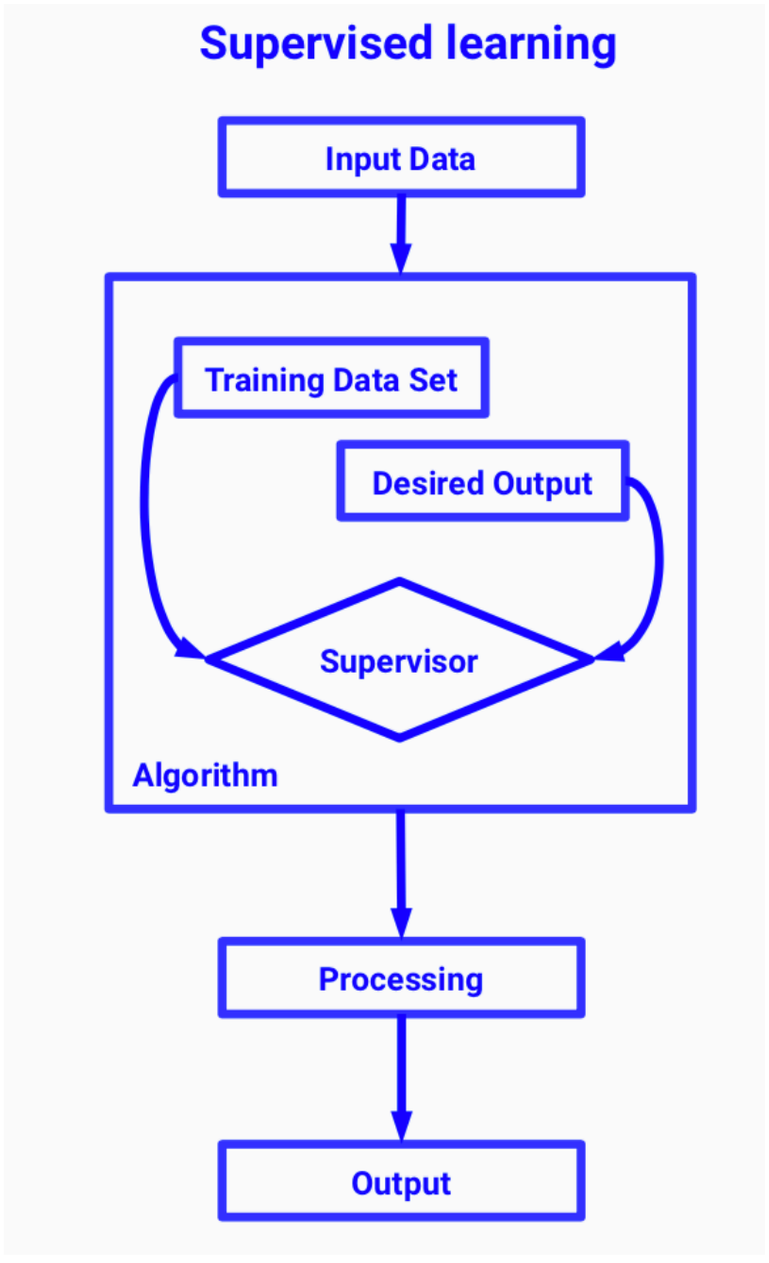
- NNs should never be used as black boxes: large number of **hyperparameters** need to be judiciously chosen (ideally in an automated manner)

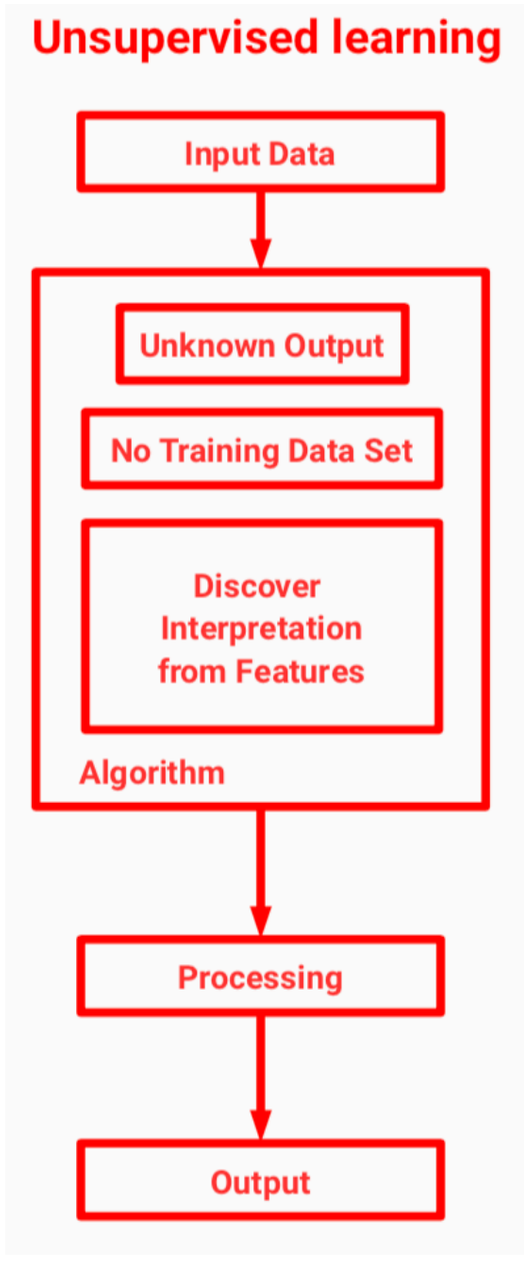
Today's lecture

- Unsupervised vs supervised learning
- K-means clustering and beyond
- Data visualisation
- Dimensionality reduction in ML

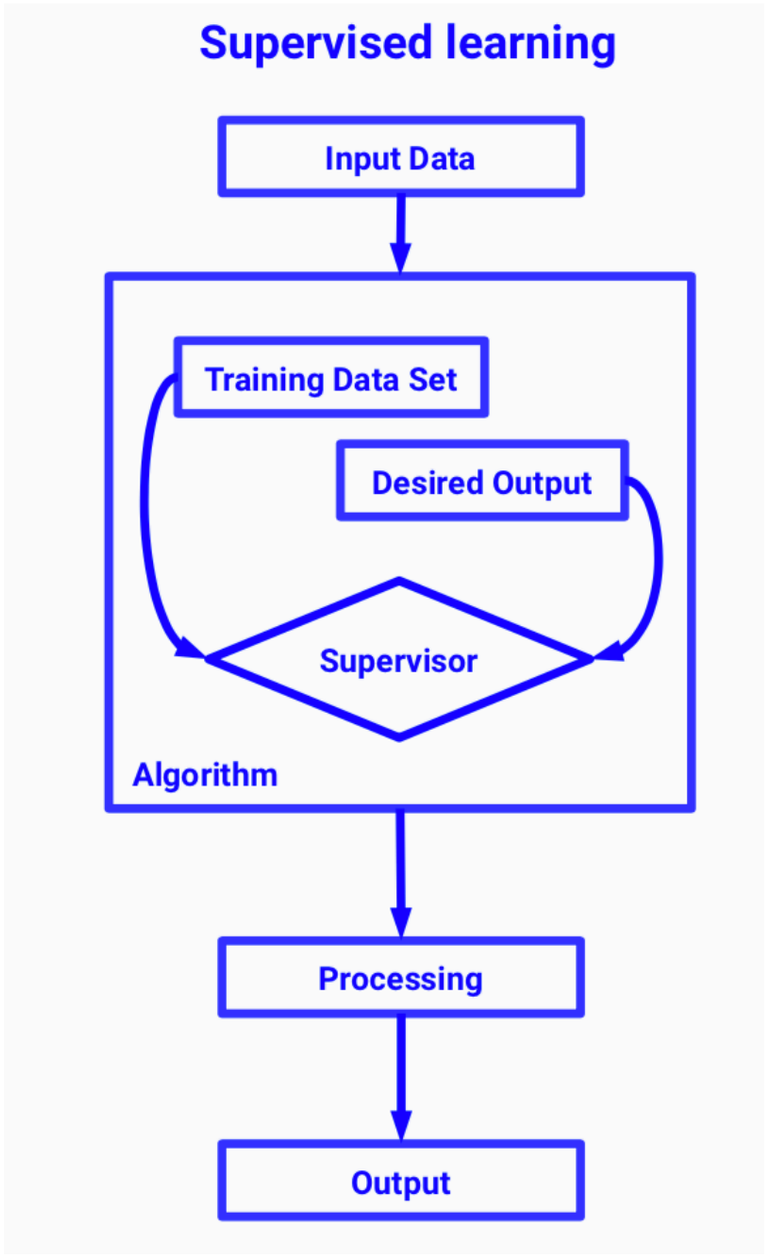
Unsupervised Learning

Supervised vs Unsupervised Learning





Supervised vs Unsupervised Learning

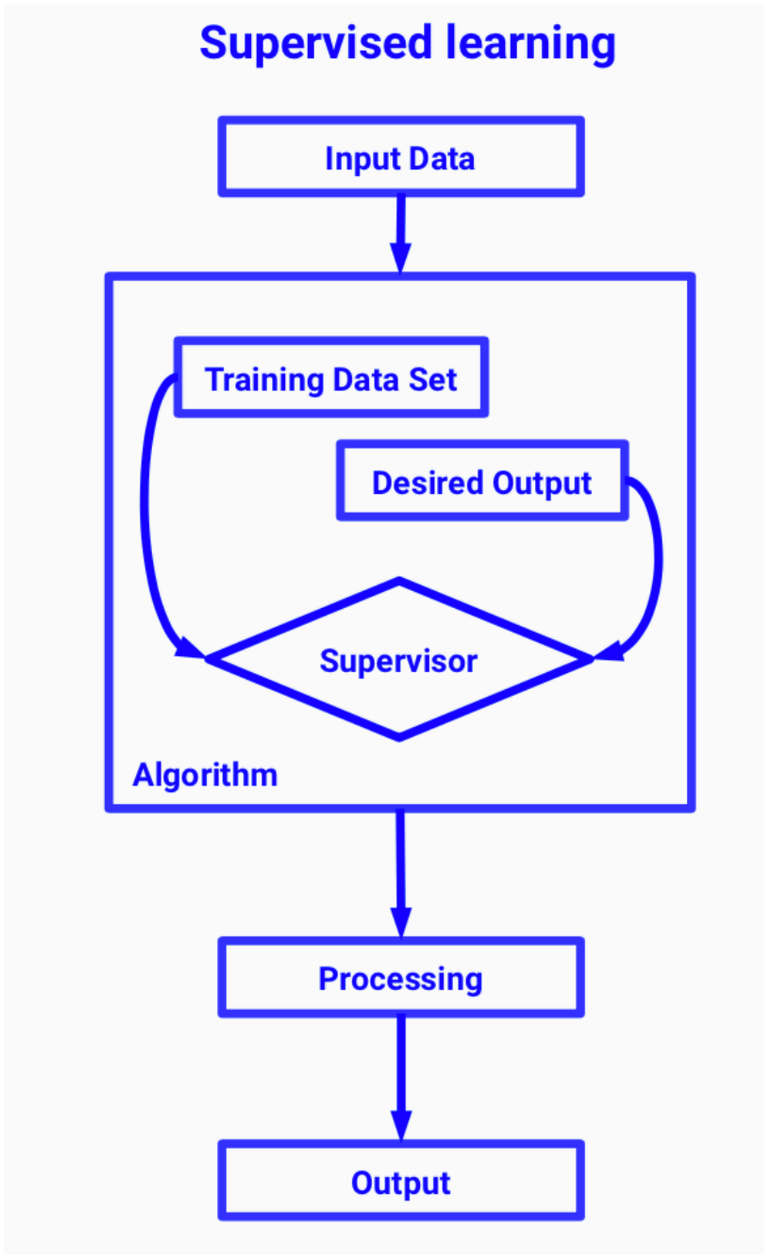


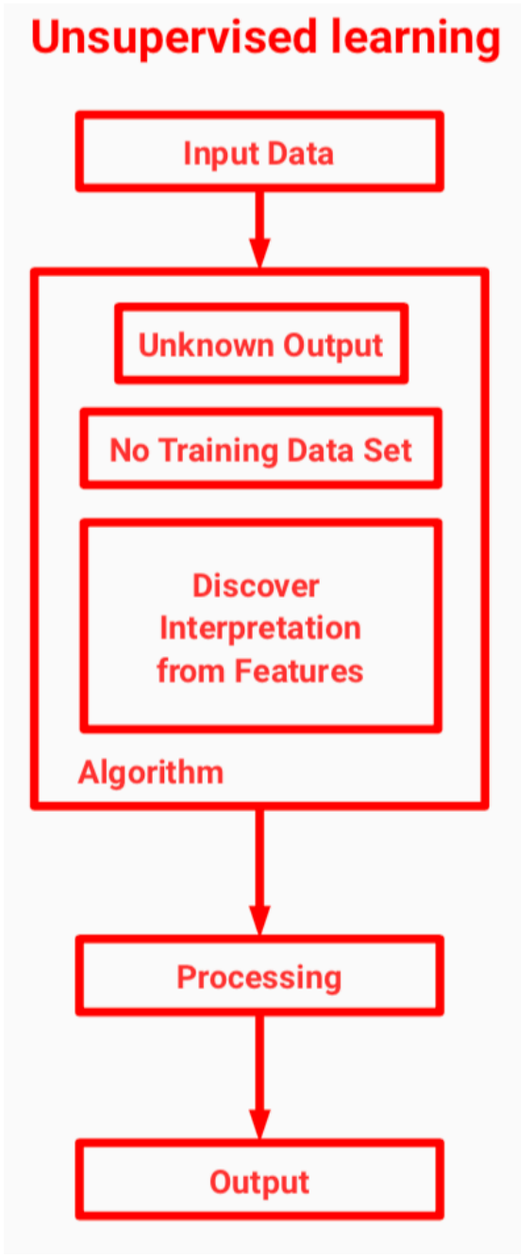
e.g. learn a function f(x) from a finite number of points $\{(x_1,f(x_1)), \ldots, (x_n,f(x_n))\}$

e.g. learn how to separate cat pictures from dog pictures (lecture 5!)

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

Potato Unsupervised Learning

sample

Cluster/group

what are we learning here? and is this the unique pattern that we could have identified in the data?

Potato Unsupervised Learning

sample

Cluster/group

here learning is carried out done in ``potato color'' space, but one could also consider ``potation size'' or ``potato shape'' spaces ...

i) identify features + ii) use them to group the data points

Unsupervised Learning

What are the **benefits** of unsupervised learning?

☑ Identify unknown patterns in unlabelled data.

e.g. potatoes come in different colours, sizes, and shapes

☑ Find features which can be useful for categorisation.

e.g. sort out my potatoes as a function of their type

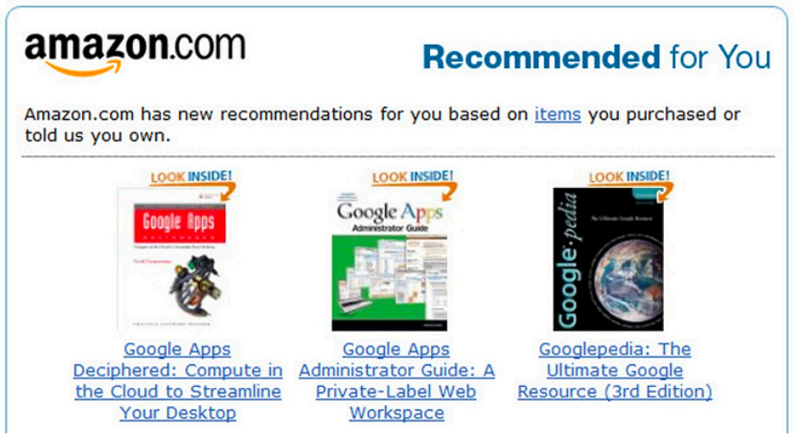
☑ It is easier to get automatically unlabelled data than labeled data, which often needs manual intervention.

e.g. I don't need to label my potatoes with their variety!

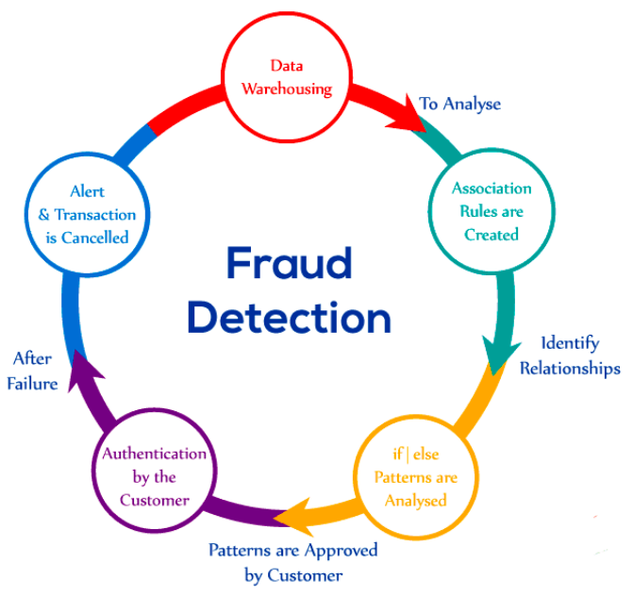
Unsupervised Learning

Some representative applications of unsupervised learning

☑ **Group customers** as a function of previous purchase and browsing history, to offer tailored recommendations



☑ Detect **anomalies** in credit card transactions that may indicate fraudulent use



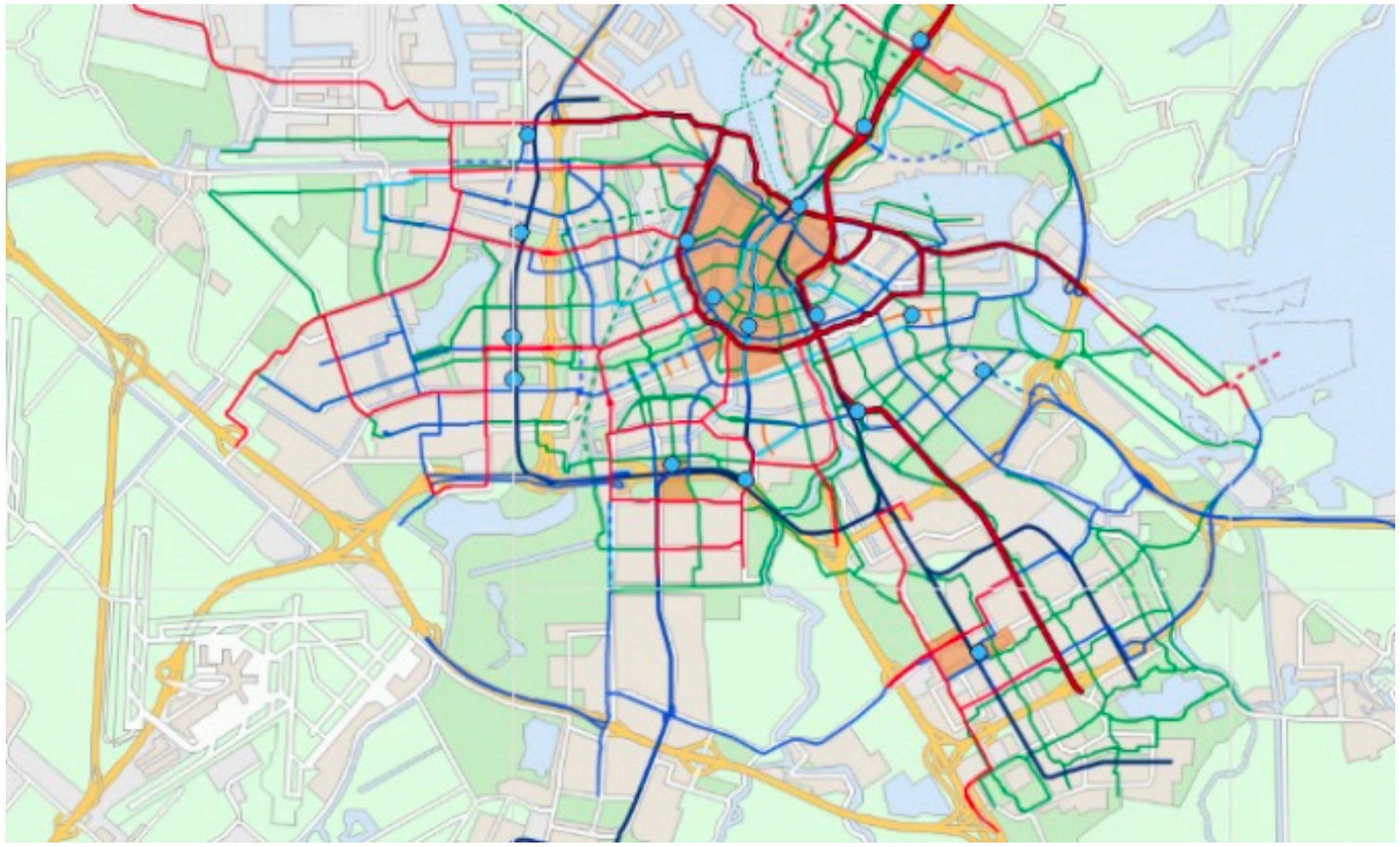
what would be my ``parameter space'' here?

Unsupervised Learning

Some representative applications of unsupervised learning

☑ Identify **travel patterns** e.g. of ride-sharing (bicycle, scooter, car) services

what information we should look for? And why it is important?

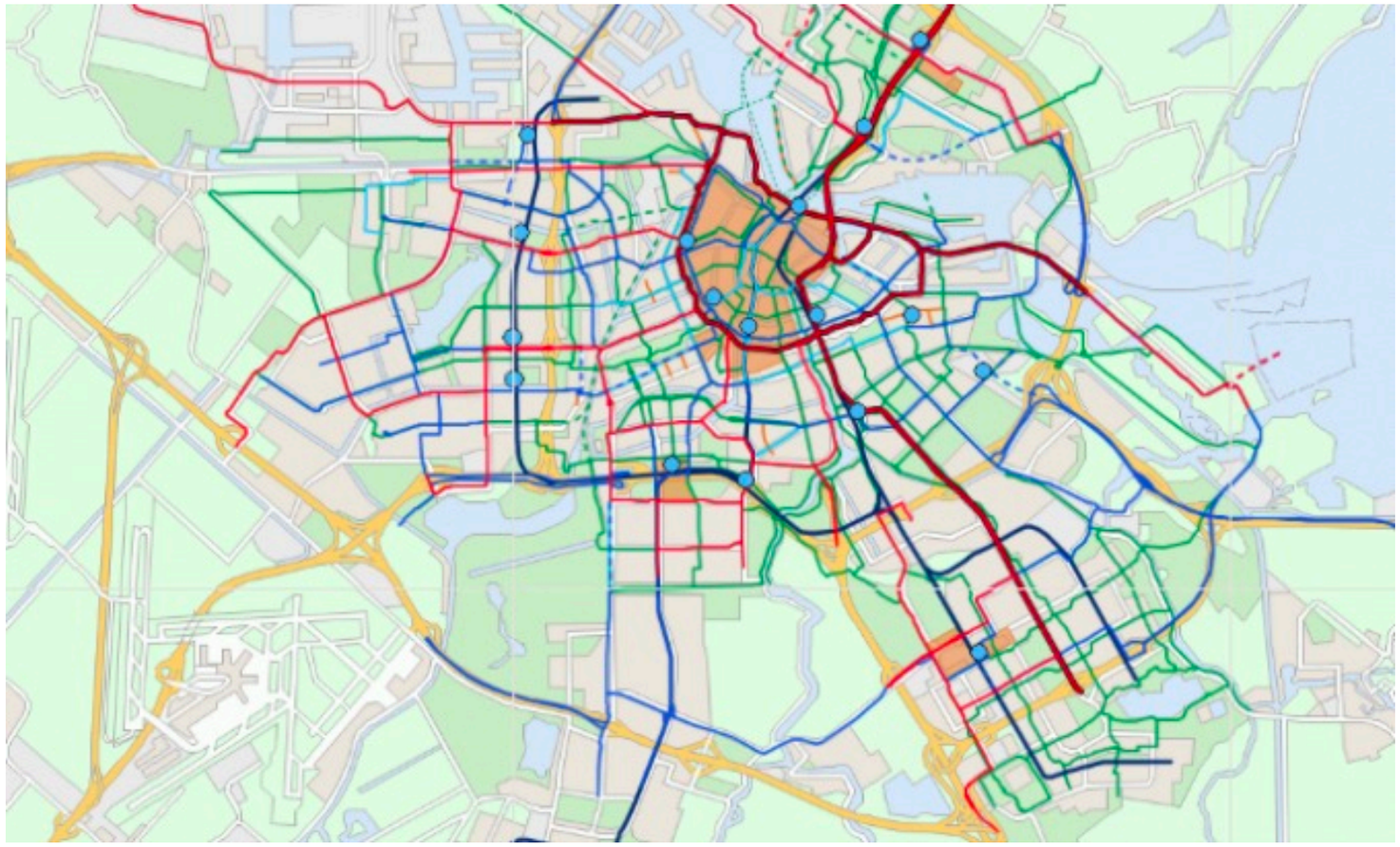


Unsupervised Learning

Some representative applications of unsupervised learning

☑ Identify **travel patterns** e.g. of ride-sharing (bicycle, scooter, car) services

where to put the scooters, at what time, how many per hub etc



Unsupervised Learning

What are the **disadvantages** of unsupervised learning?

☑ Some **interpretation** work is required after classification

e.g. what do the identified patterns actually mean?

a peculiar credit card translation does not always imply fraud!

☑ The lack of labels makes the process less efficient than in supervised learning

e.g. is the algorithm identifying the relevant features?

☑ Non-trivial to obtain **model prediction**: how to construct a rule that can be applied to identify the same patterns in new data?

e.g. what happens if I now have new potatoes?

Unsupervised Learning:

Clustering

Clustering

sample

Cluster/group

data points: potatoes with different features (size, shape, colour)

desired output: cluster potatoes wrt some relevant feature e.g. potato variety

note: no labels attached to the potatoes!

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**

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

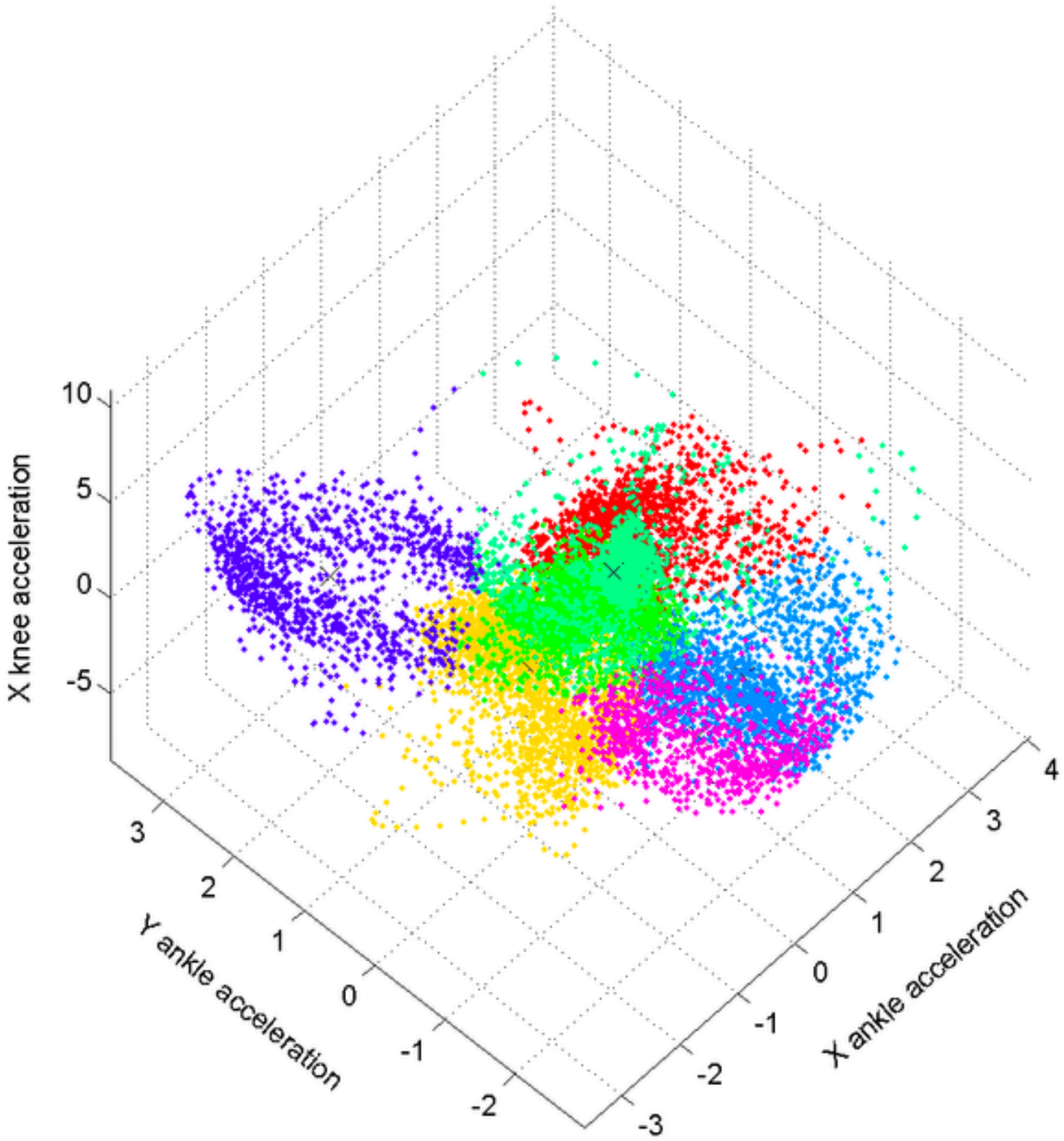
unlabelled dataset: N points with p features each

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

***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



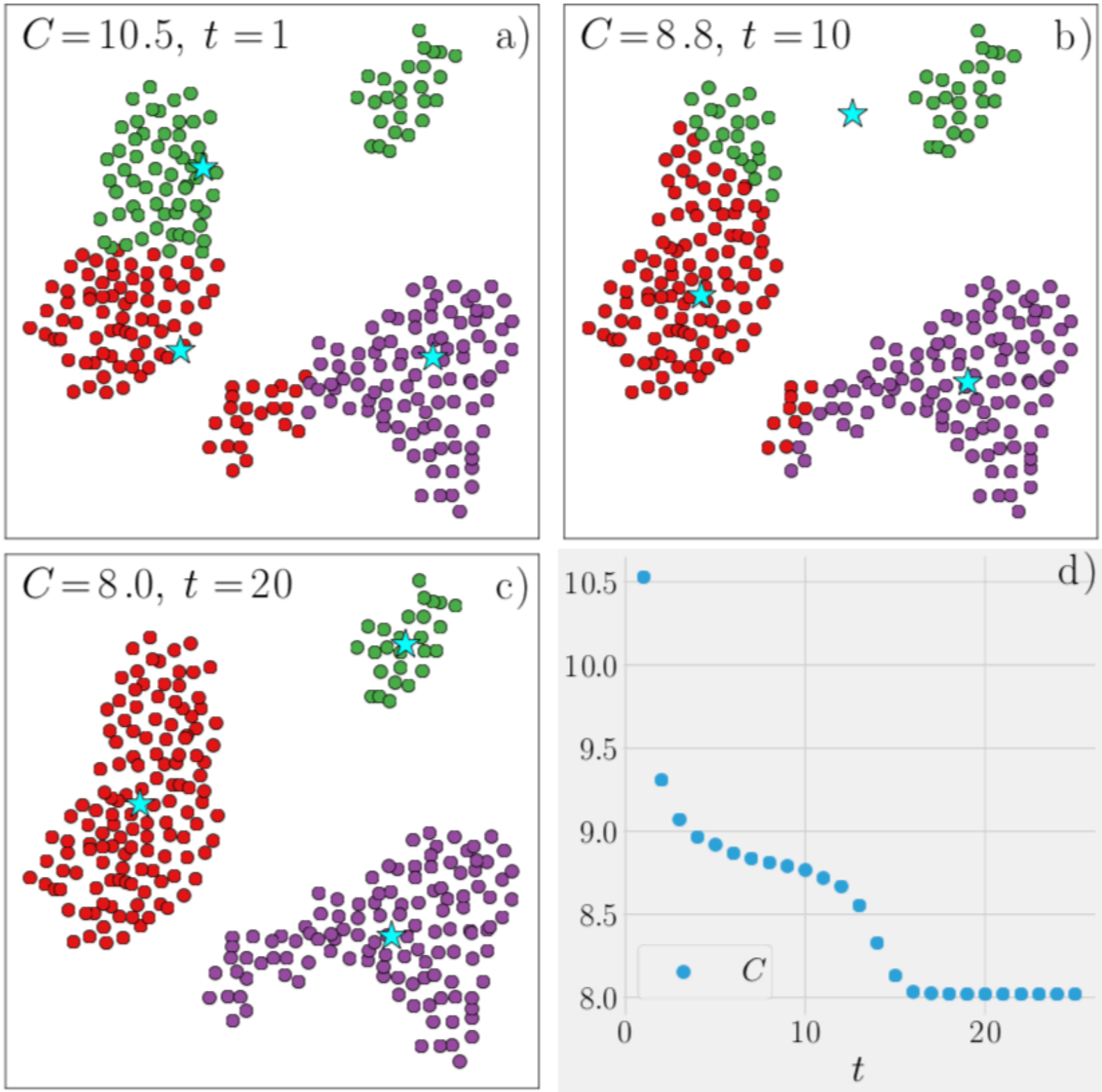
nb color here for visualisation, but not this info not available in real applications!

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

Q: what is a suitable ``distance'' or ``metric'' between two data points of the training set?

- ***training set: points in n-dimensional Cartesian space?***
- ***training set: state vectors in QM Hilbert space?***
- ***training set: pictures of cats and dogs?***
- ***training set: four-momenta of particles produced in a high-energy collision?***

defining ``closeness'' in unsupervised learning often non-trivial

Clustering

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

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

other distances possible: the user needs to define what ``closeness'' means!

binary assignment variable

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

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

$r_{nk} = 0$ ⟶ *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

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

this algorithm alternates iteratively between two main steps:

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

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

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

$$r_{nk}=1 \quad \text{if} \quad k=\arg\min_{k'}(\boldsymbol{x}_n-\boldsymbol{\mu}_{k'})$$
$$r_{nk}=0 \quad \text{if} \quad k\neq\arg\min_{k'}(\boldsymbol{x}_n-\boldsymbol{\mu}_{k'})$$

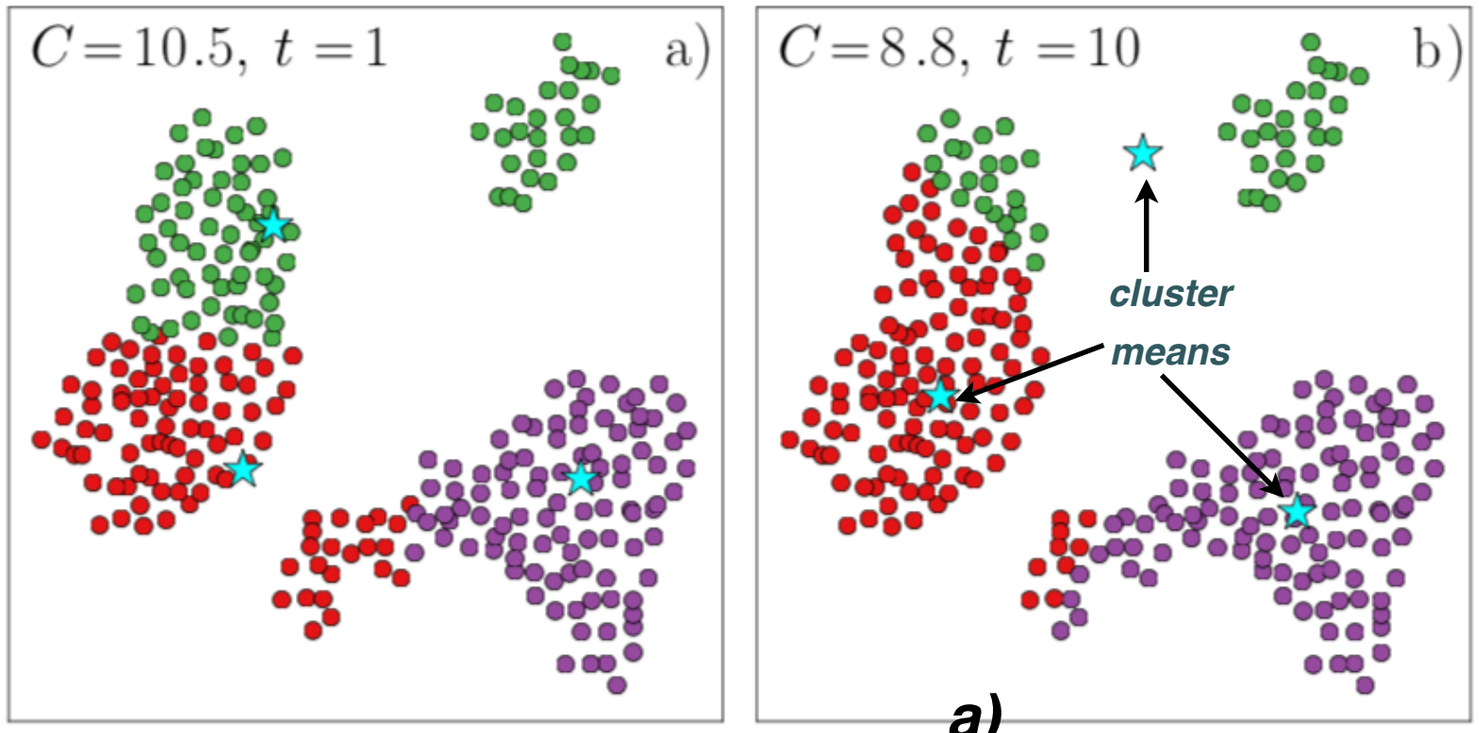
iterate until convergence achieved!

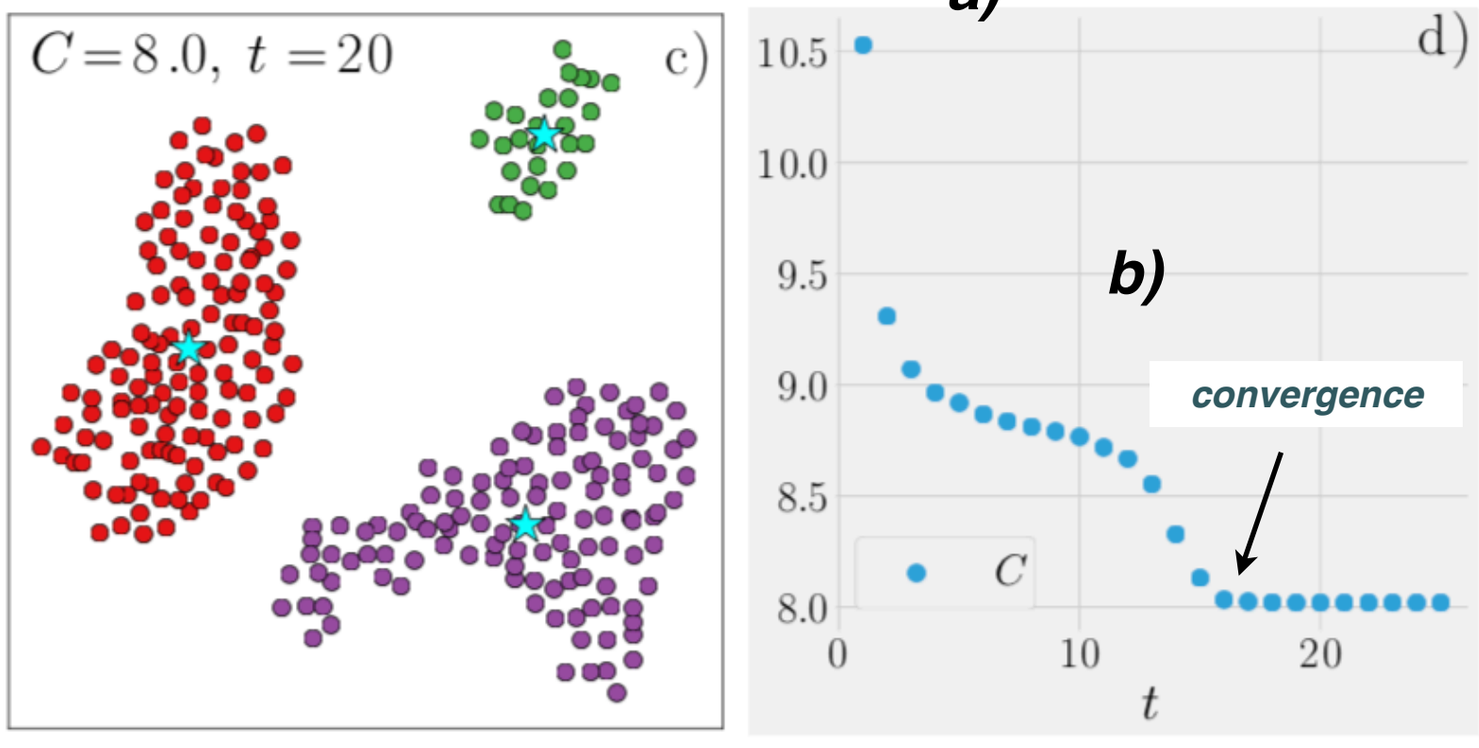
note that here GD not required, optimisation is semi-analytical

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

each colour: different cluster



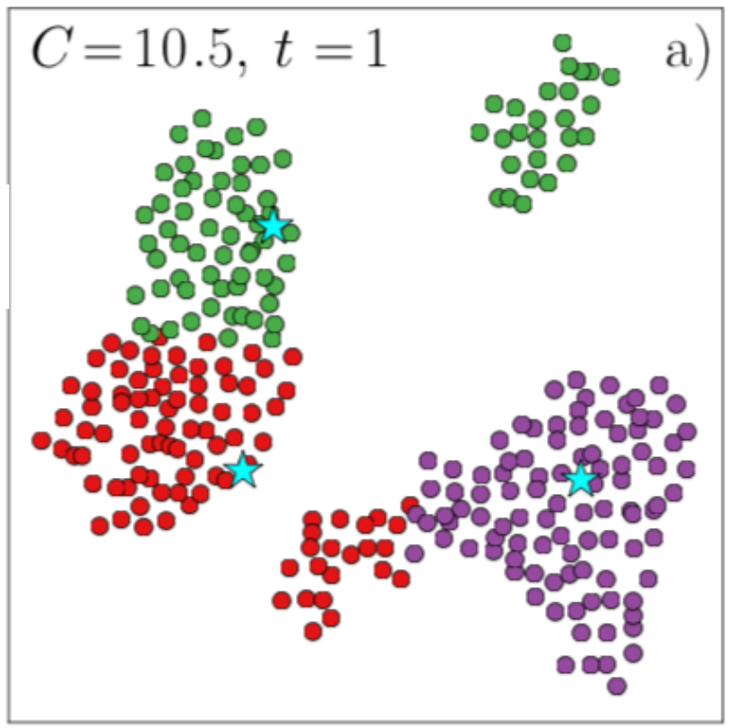


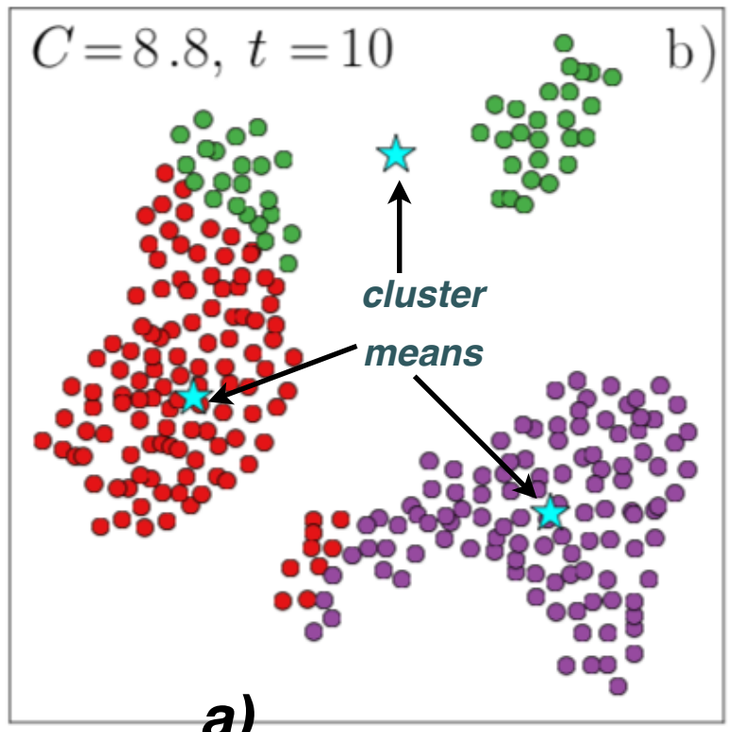
here overfitting not possible: there exists a unique assignment that minimises the cost function

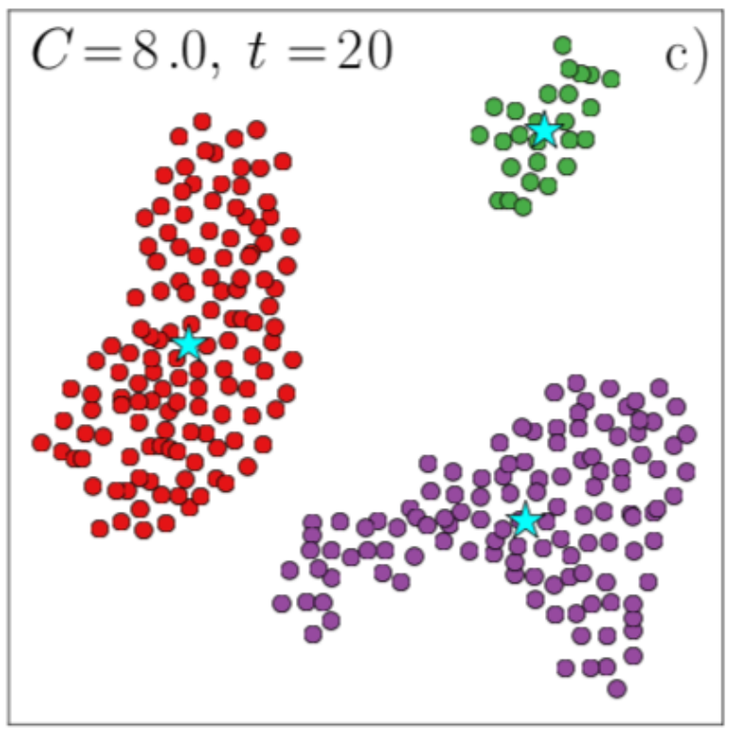
Clustering

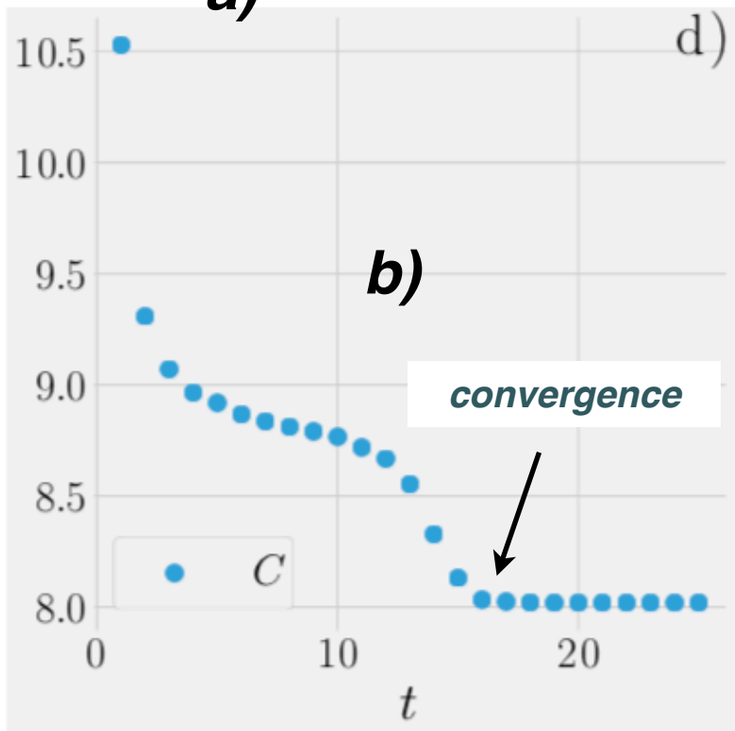
these two steps are iterated until some **convergence criterion** is achieved, *e.g.* when the change in the cost function between two iterations is below some threshold

each colour: different cluster







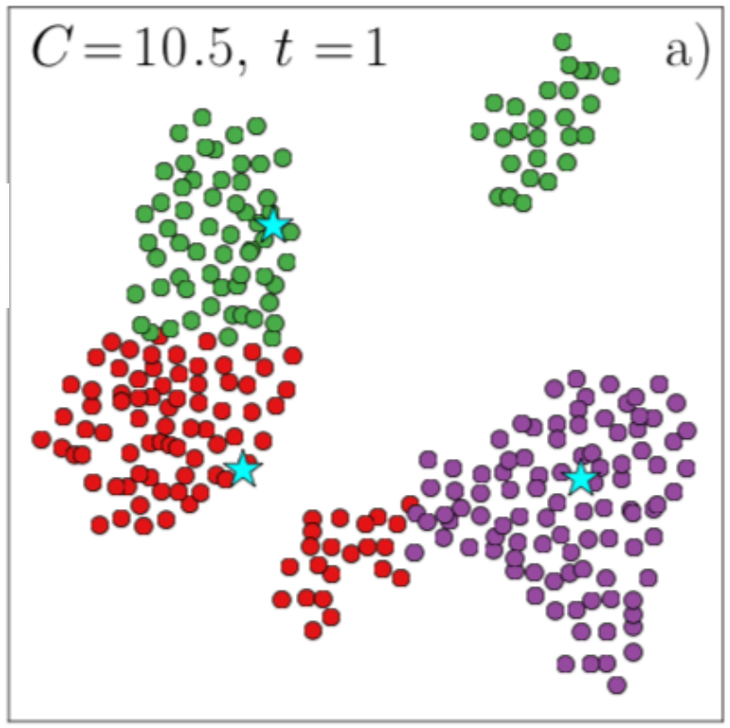


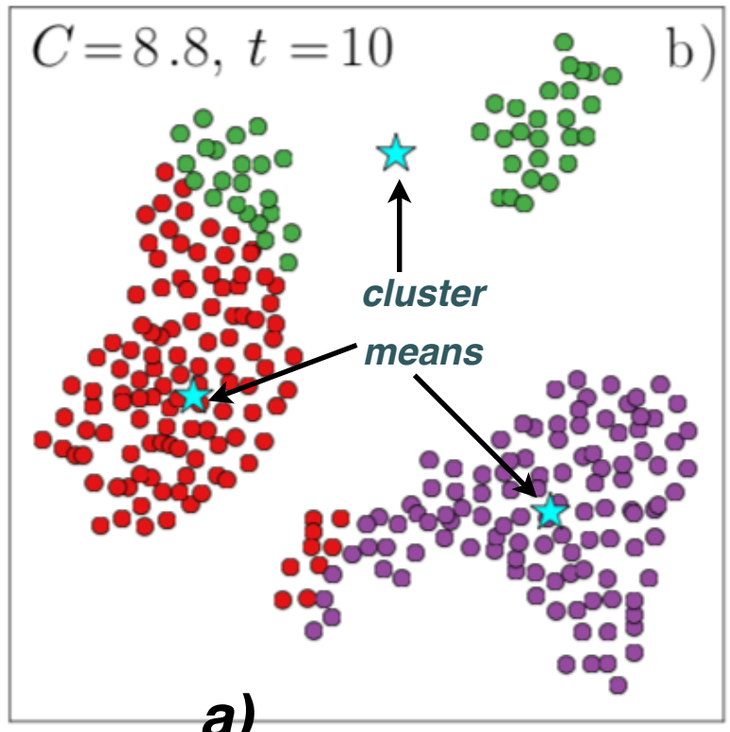
what is the underlying assumption in K-means clustering? And when it would not be justified?

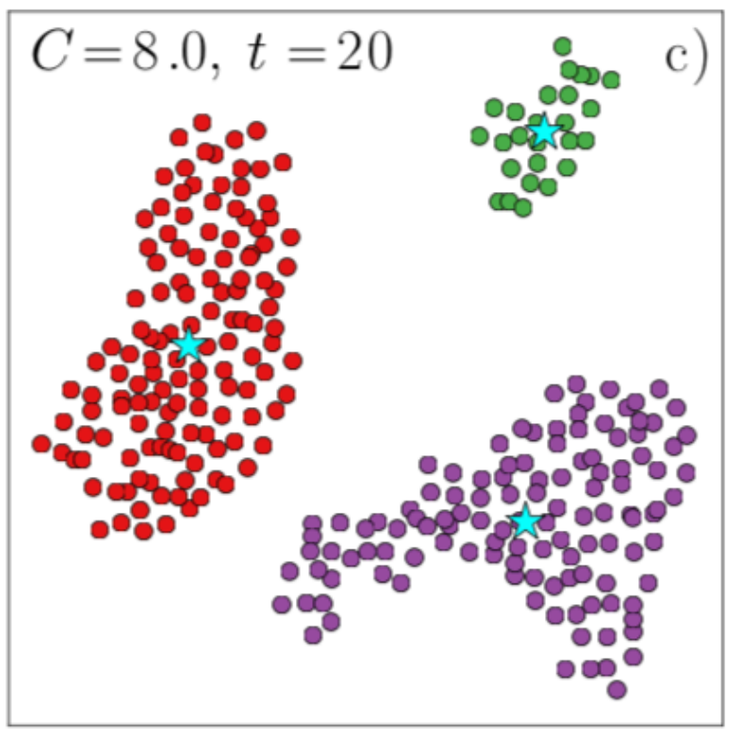
Clustering

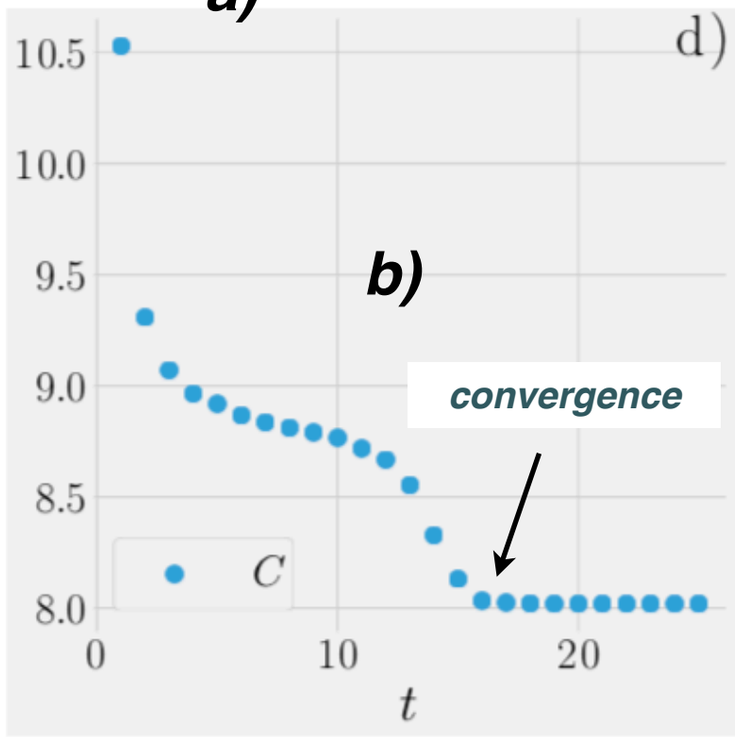
these two steps are iterated until some **convergence criterion** is achieved, *e.g.* when the change in the cost function between two iterations is below some threshold

each colour: different cluster



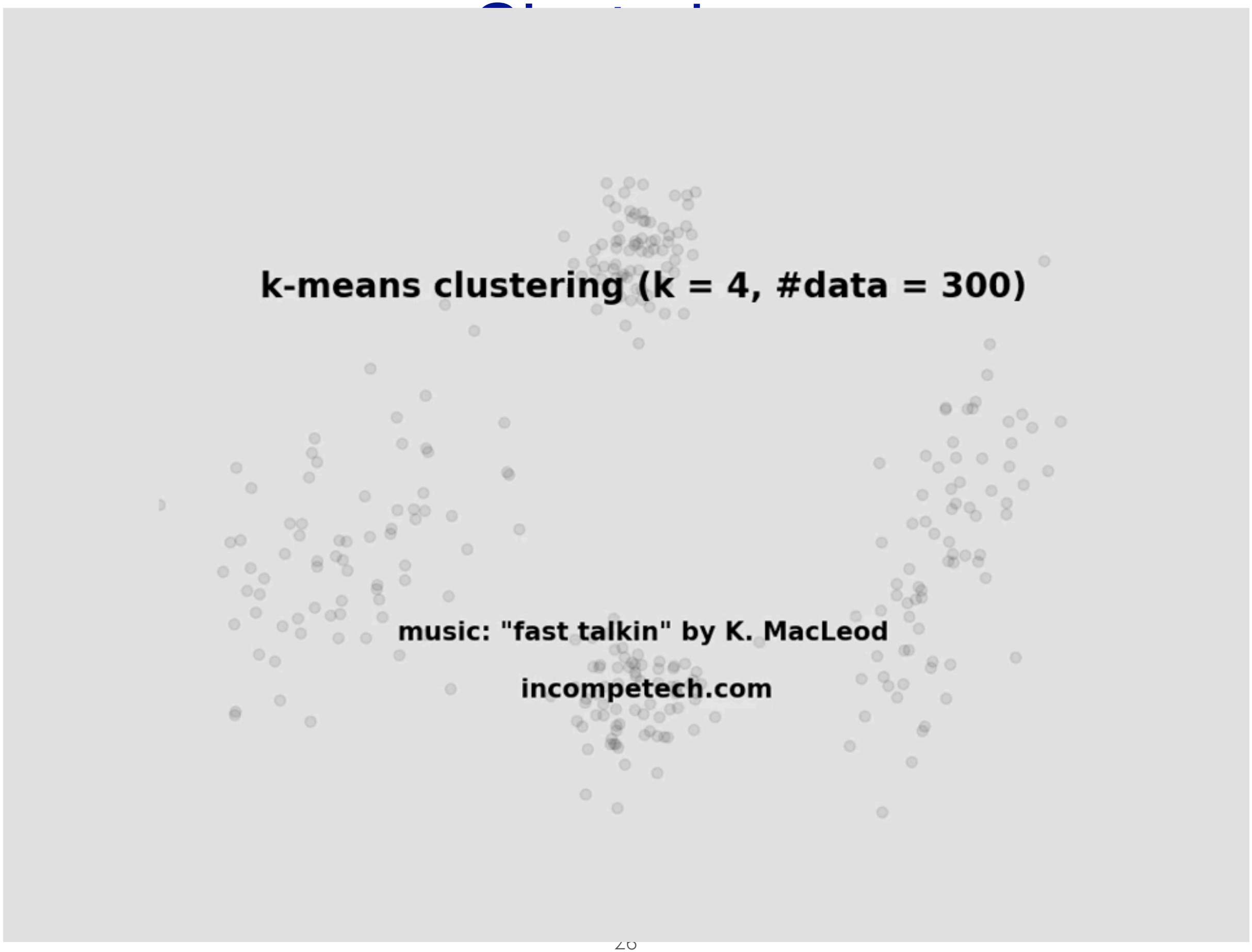






*K-means clustering can lead to spurious results since the **underlying assumption** is that the latent model has uniform variances*

fails if the underlying clusters have different variances!


k-means clustering (k = 4, #data = 300)
music: "fast talkin" by K. MacLeod
incompetech.com

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*

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

- 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

$$(X_i, X_j) \quad \text{such that} \quad (i, j) = \arg\min_{i'j'} d(X_{i'}, X_{j'})$$

- ***(3)*** Merge the pair into a single cluster. Iterate ***(2)*** and ***(3)*** until a single cluster remains

Hierarchical clustering

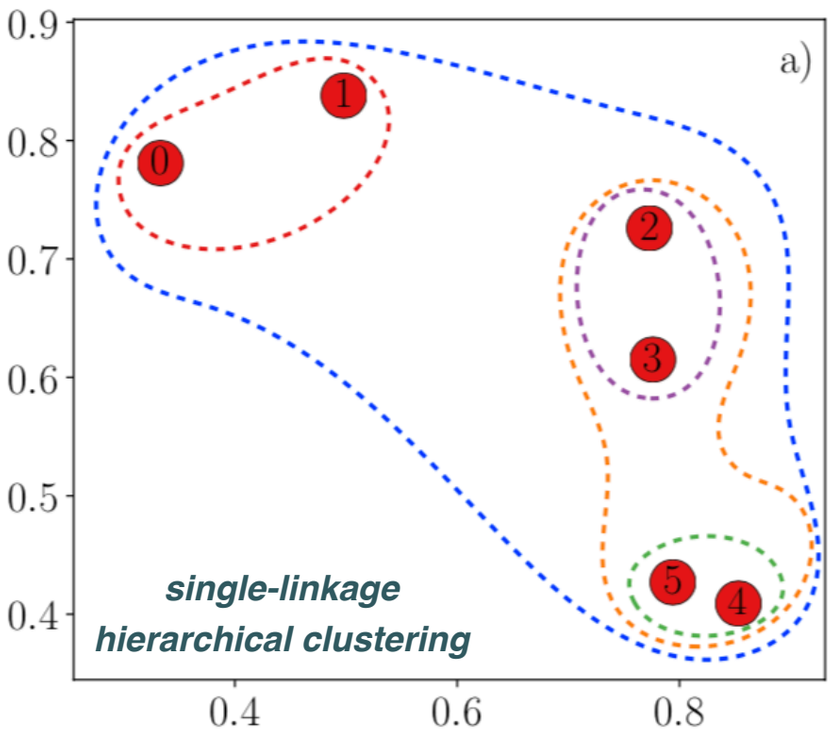
Clearly the results of hierarchical clustering depend on the **choice of distance**

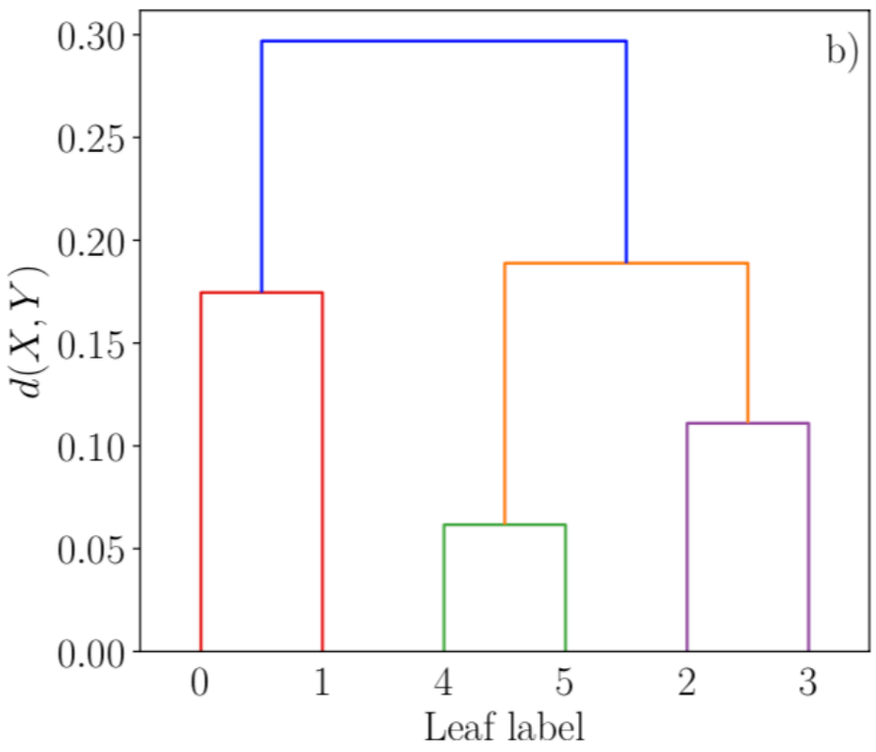
distance between clusters

single linkage → $d(X_i, X_j) = \min_{\boldsymbol{x}_i \in X_i, \boldsymbol{x}_j \in X_j} ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2$

complete linkage → $d(X_i, X_j) = \max_{\boldsymbol{x}_i \in X_i, \boldsymbol{x}_j \in X_j} ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2$

Euclidean distance





Hierarchical clustering

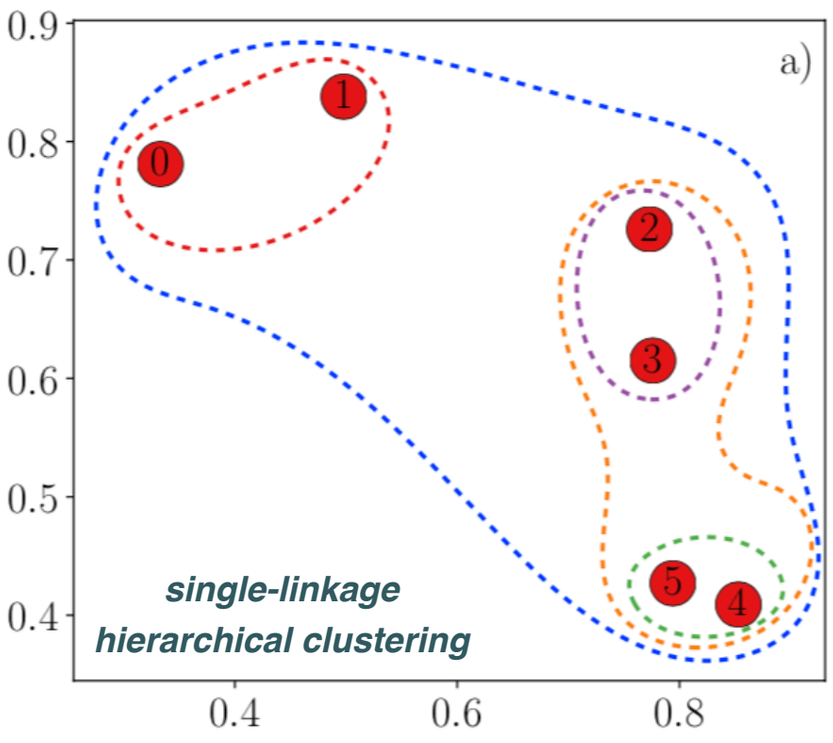
Clearly the results of hierarchical clustering depend on the **choice of distance**

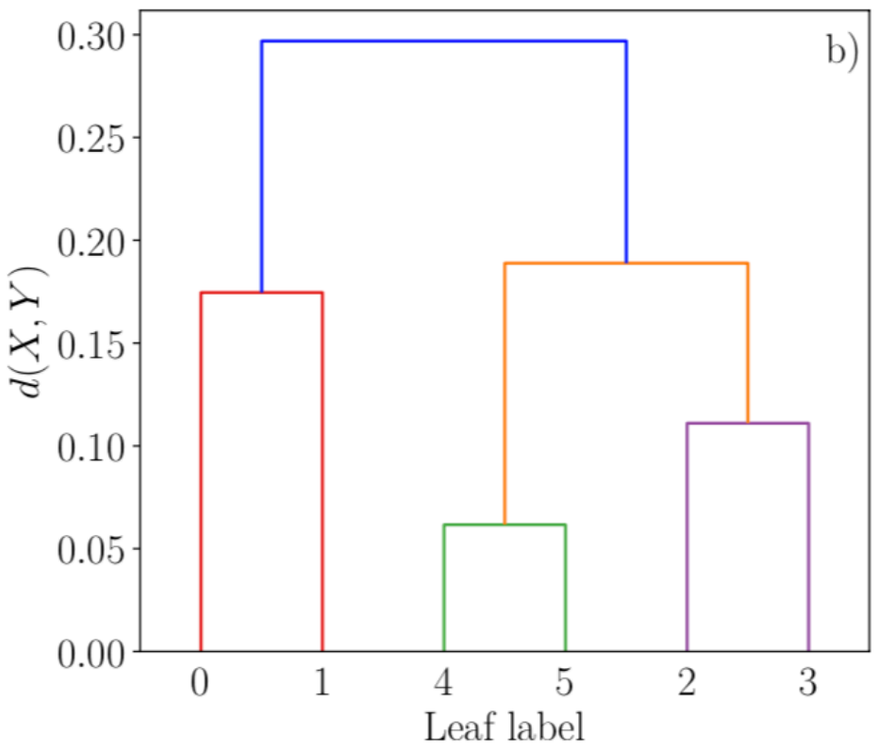
distance between clusters

single linkage → $d(X_i, X_j) = \min_{\boldsymbol{x}_i \in X_i, \boldsymbol{x}_j \in X_j} ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2$

complete linkage → $d(X_i, X_j) = \max_{\boldsymbol{x}_i \in X_i, \boldsymbol{x}_j \in X_j} ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2$

Euclidean distance





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

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)

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

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

$$\mathcal{N}\left(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}\right) \sim \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})^{T}\right]$$

one of the K gaussians

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

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

probability of drawing a point from mixture k

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

GMM parameters

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

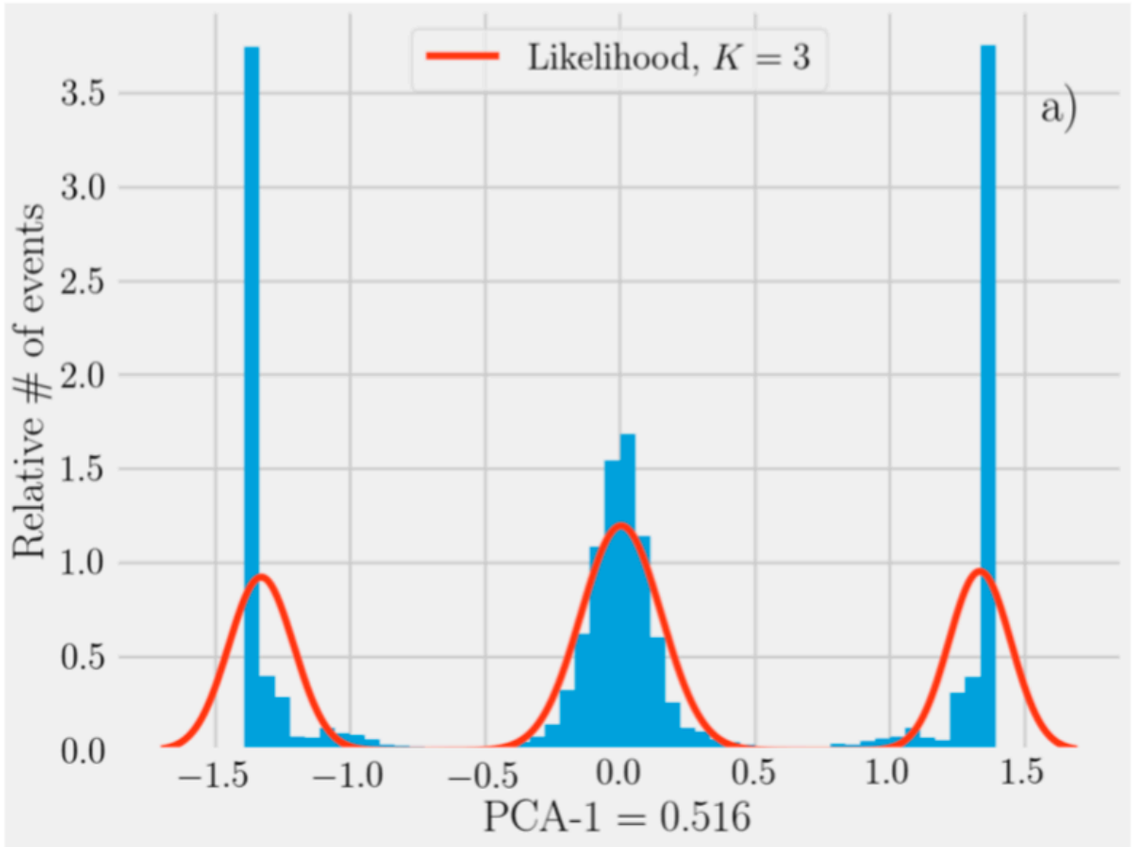
$$\gamma_k(\boldsymbol{x}) \sim \mathcal{N}\left(\boldsymbol{x}\,|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \pi_k$$

model parameters found my maximising likelihood using SGD

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

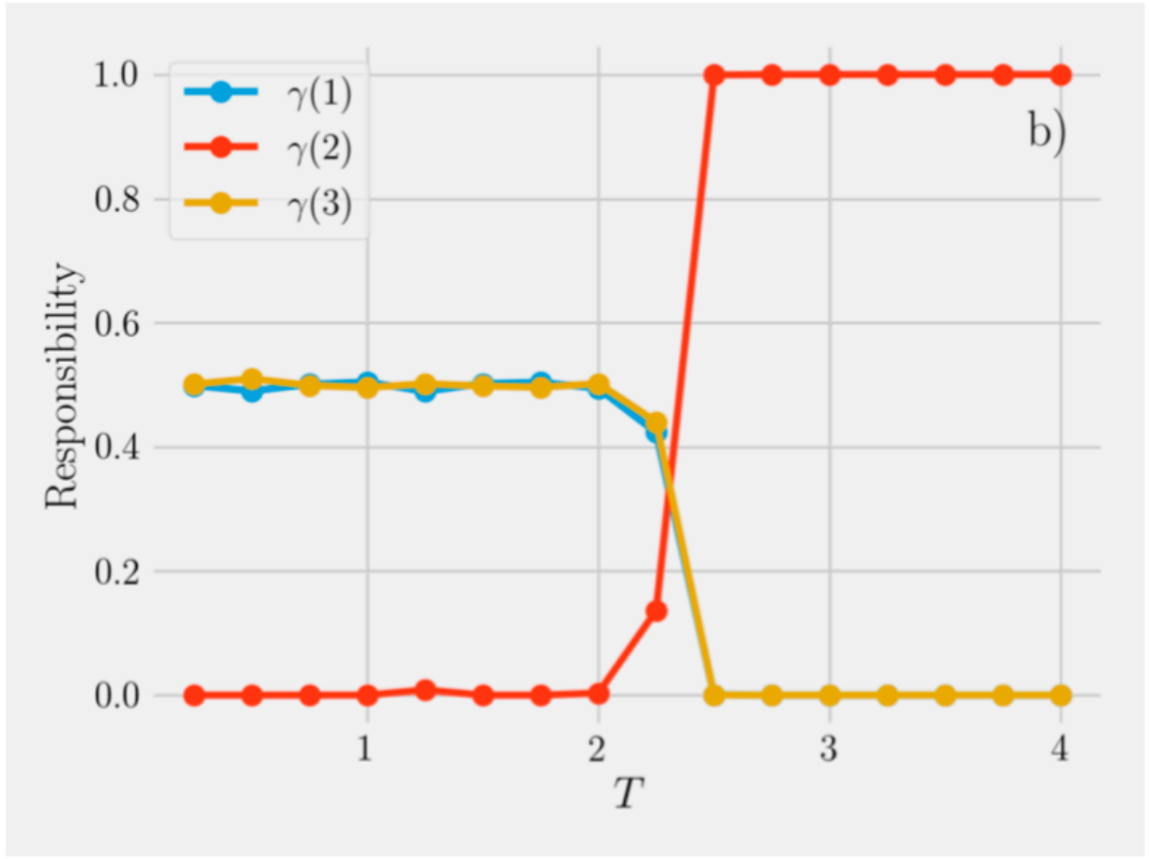
Gaussian Mixture Models

Ising dataset fitted to 3-component GGM



1st principal component (magnetisation)

Probability of being on each phase



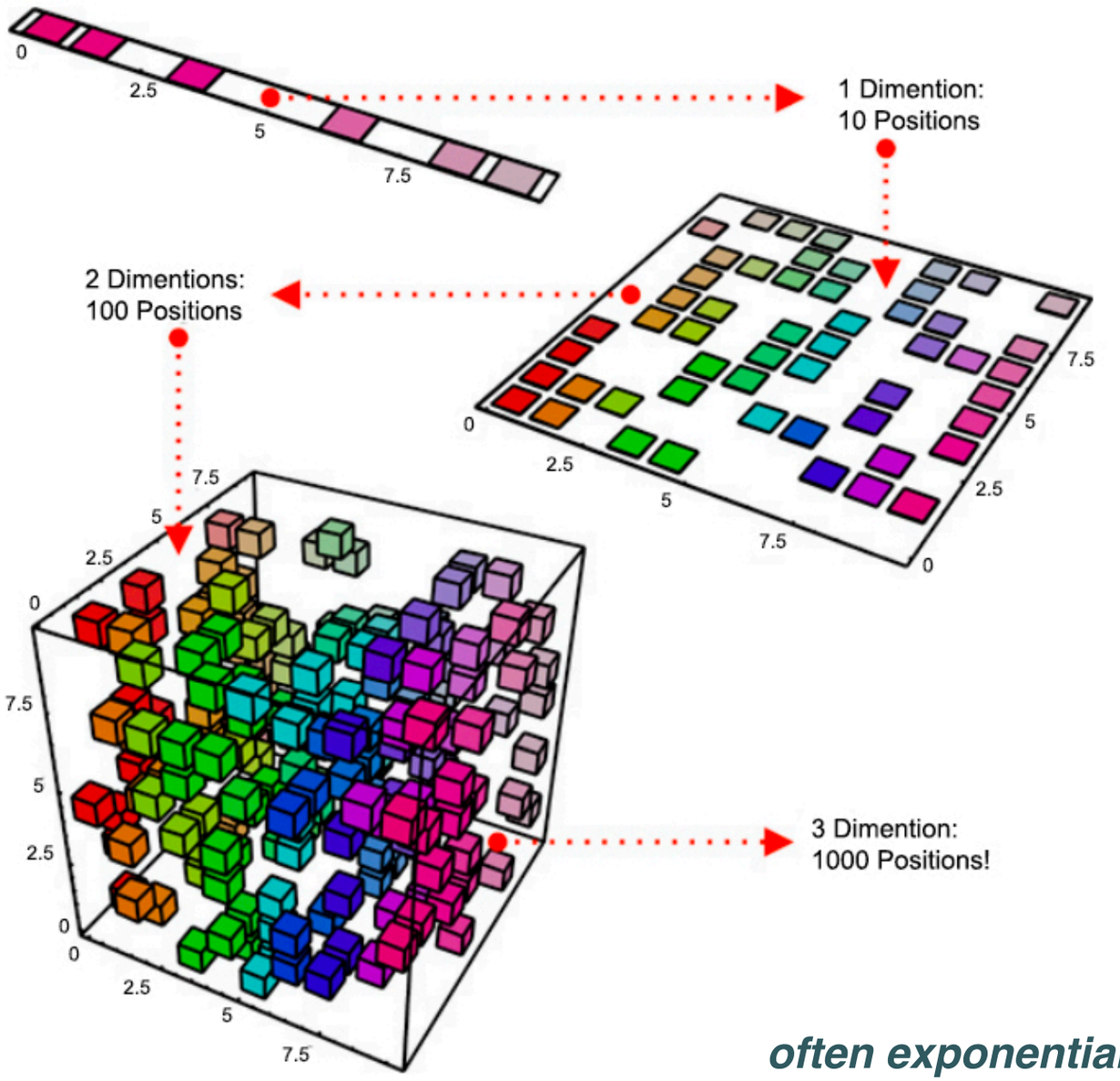
probability coincide at critical point

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!**



often exponential growth of complexity

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 that 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 that done: many pitfalls associated to **high-dimensionality datasets**

Q: if you draw samples at random from a n-dim space, which region will be sampled more often?

tip: think about the 2D case to being with

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 that done: many pitfalls associated to **high-dimensionality datasets**

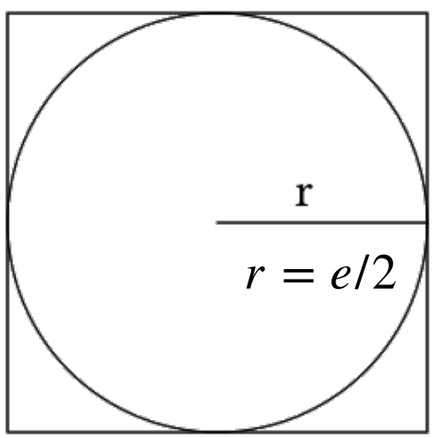
- *High-dimensional data lives near the edge of the sample space*

Dimensional reduction

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

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

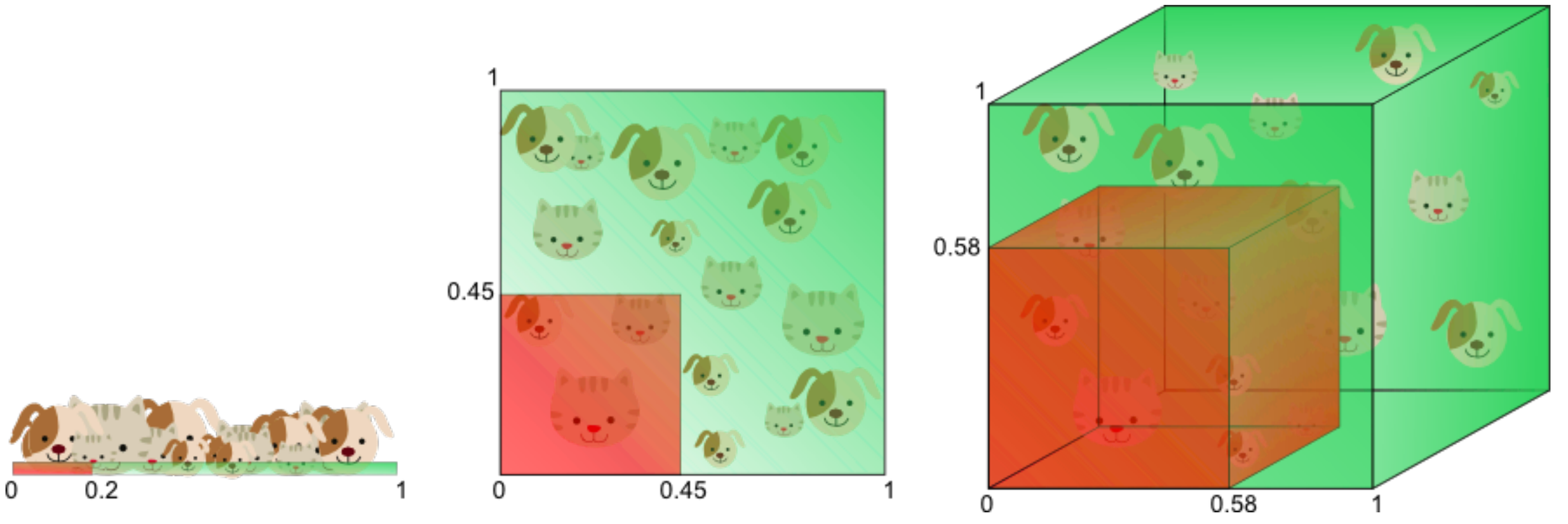
consider a *D*-dimensional sphere *S* of radius *e/2* entered at origin



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

$$P\left(\boldsymbol{x}_i \in \mathcal{S}\right) \simeq \frac{\pi^{D/2}(e/2)^D/\Gamma(D/2+1)}{e^D} = \simeq \frac{\pi^{D/2}}{2^D D^D} \to 0$$

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



Dimensional reduction

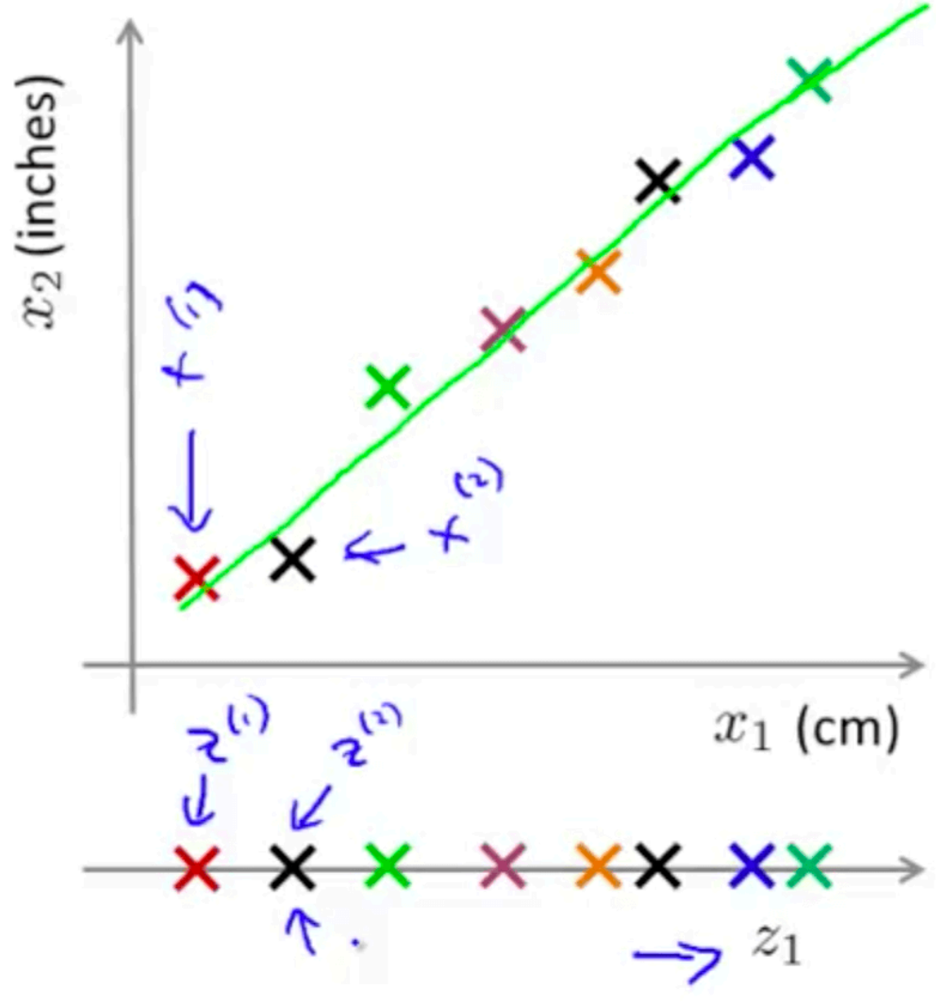
- *High-dimensional data lives near the edge of the sample space*
- *We need to conserve information on original pair-wise distances or similarities when transforming to the latent space*

Q: is it always justified to project n-dimensional data into a m-dimensional subspace (m < n) ?

Dimensional reduction

- *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*

Q: is it always justified to project n-dimensional data into a m-dimensional subspace (m < n) ?



Reduce data from 2D to 1D

$$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$$

$$x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$$

$$\vdots$$

$$x^{(m)} \rightarrow z^{(m)}$$

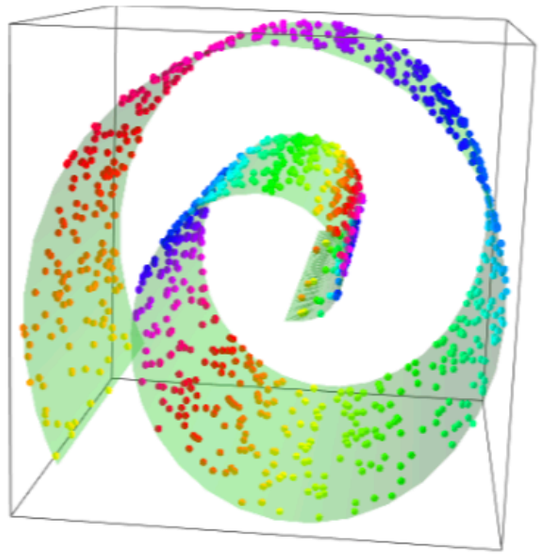
Here I can reduce from 2D to 1D1! Why dimensionality reduction works in this case?

Dimensional reduction

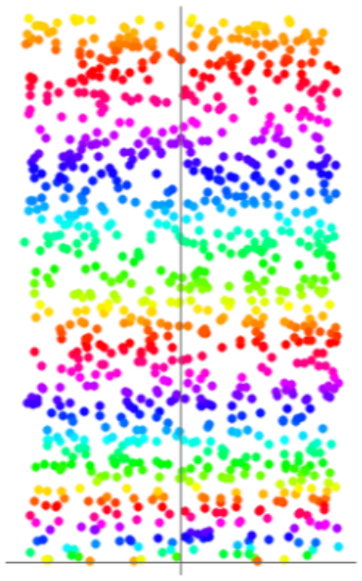
- *High-dimensional data lives near the edge of the sample space*
- *We need to conserve information on original pair-wise distances or similarities when transforming to the latent space*

the minimum number of parameters needed to capture the original patters is the **intrinsic dimensionally of the data**

a) ***original data space: 3D***



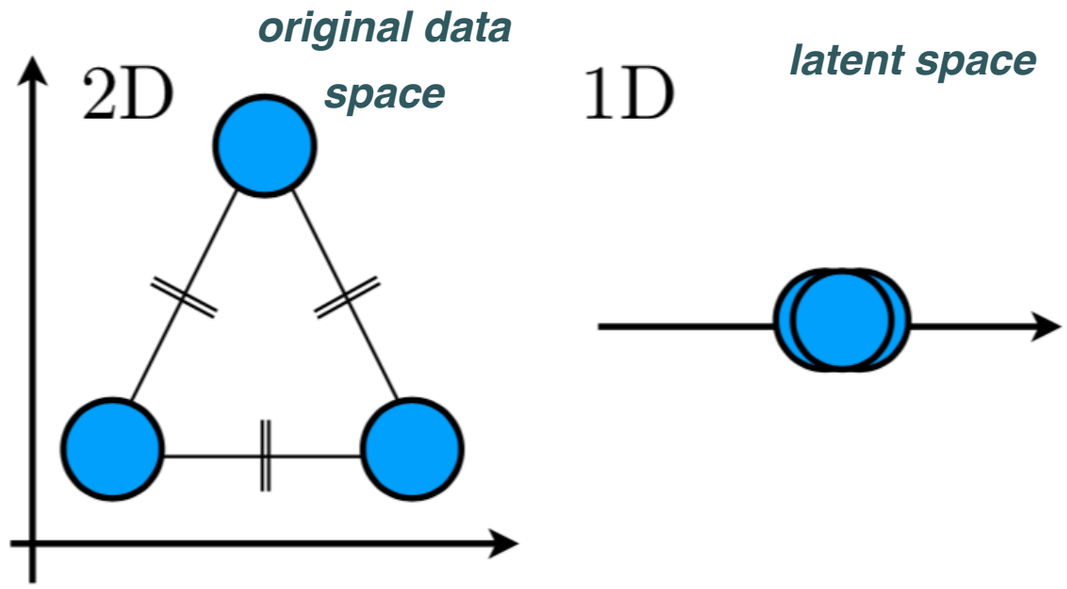
b) ***latent space: 2D***



Dimensional reduction

- *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*
- *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**

there are two main classes of **dimensional reduction methods**

linear methods, based on linear transformations

e.g. Principal Component Analysis (PCA)

non-linear methods, based on non-linear transformations

e.g. Stochastic Neighbour Embedding (SNE)

Principal Component Analysis

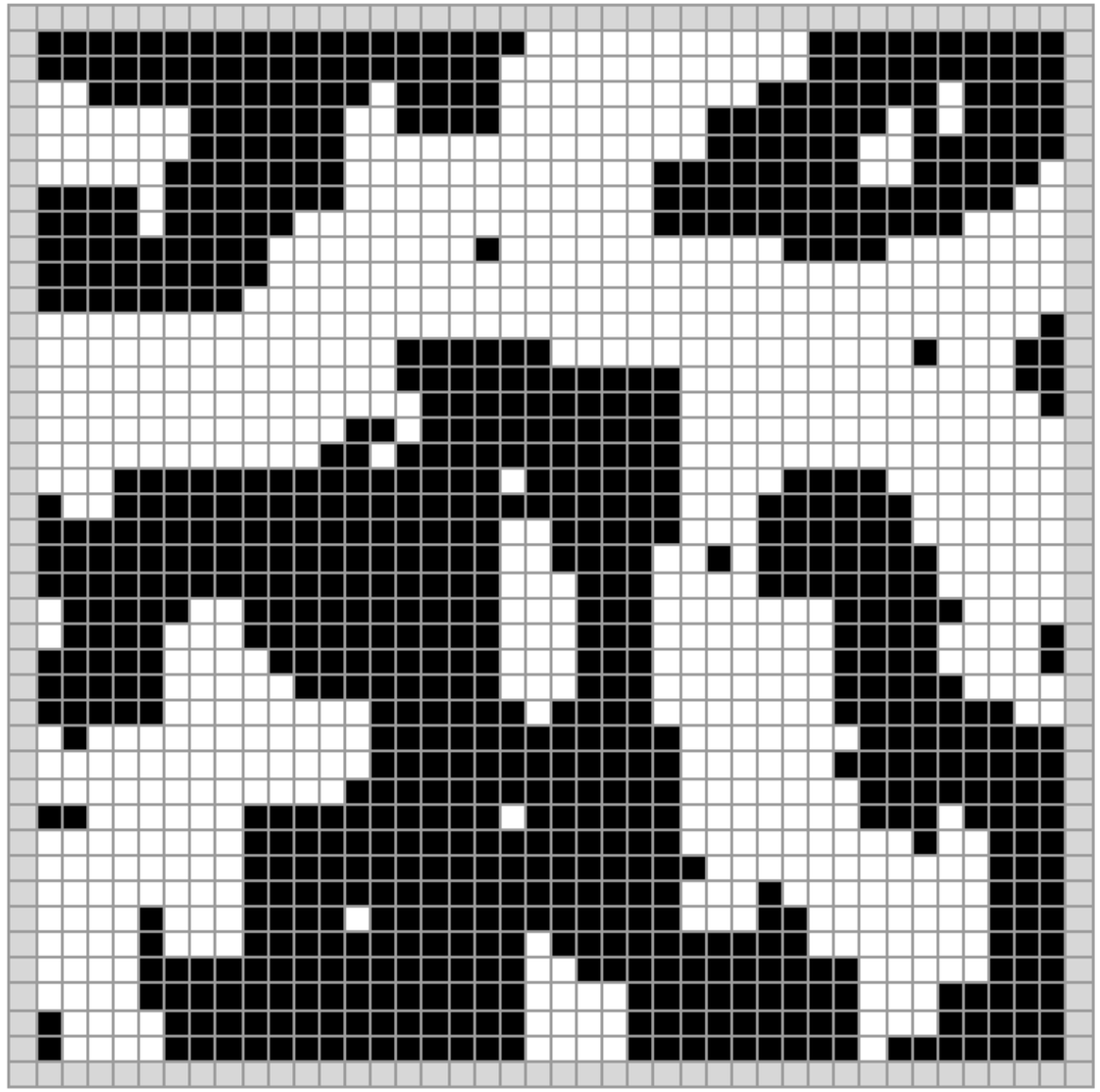
PCA projections often capture the **large-scale structure** of high-dimensional datasets

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/The2DIsingModelMonteCarloSimulationUsingTheMetropolisAlgorit/

Principal Component Analysis

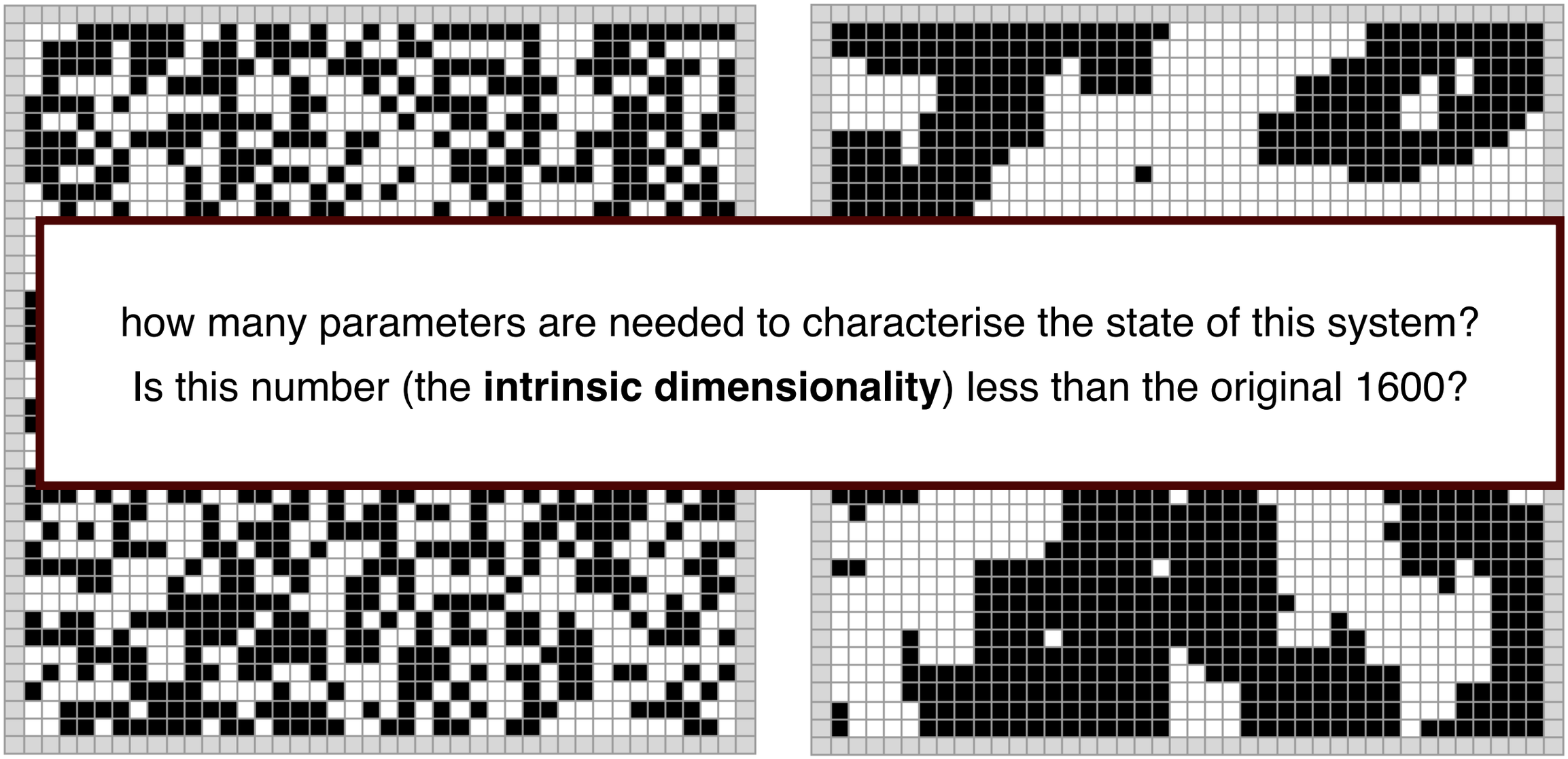
PCA projections often capture the **large-scale structure** of high-dimensional datasets

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/The2DIsingModelMonteCarloSimulationUsingTheMetropolisAlgorit/

Principal Component Analysis

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

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

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

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

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

where rows are the data points and columns are features

$$\boldsymbol{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n,1} & x_{n,2}\,. & \cdots & x_{n,p} \end{pmatrix}$$

data points

features

Principal Component Analysis

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

$$\boldsymbol{\Sigma}(X) = \frac{1}{n-1} X^T X \quad \rightarrow \quad \underset{\substack{\textit{p-dimensional} \\ \textit{in space of features}}}{\boldsymbol{\Sigma}_{lm}} = \frac{1}{n-1} \underset{\textit{sum over data points}}{\sum_{i=1}^{n}} X_{li} X_{im}$$

from where we see that $\boldsymbol{\Sigma}_{lm}$ 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

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$$

columns are eigenvectors of A

diagonal matrix of eigenvalues

columns are eigenvectors of A

Principal Component Analysis

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

$$\boldsymbol{\Sigma}(X) = \frac{1}{n-1} X^T X \quad \rightarrow \quad \underset{\substack{\textit{p-dimensional} \\ \textit{in space of features}}}{\boldsymbol{\Sigma}_{lm}} = \frac{1}{n-1} \sum_{i=1}^{n} X_{li} X_{im}$$

sum over data points

from where we see that $\boldsymbol{\Sigma}_{lm}$ 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

$$X = USV^T$$

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

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

U,V are unitary matrices

diagonal matrix with eigenvalues in decreasing order along diagonal

columns of $\boldsymbol{V}$ ⟶ principal directions of $\boldsymbol{\Sigma}$

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

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

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

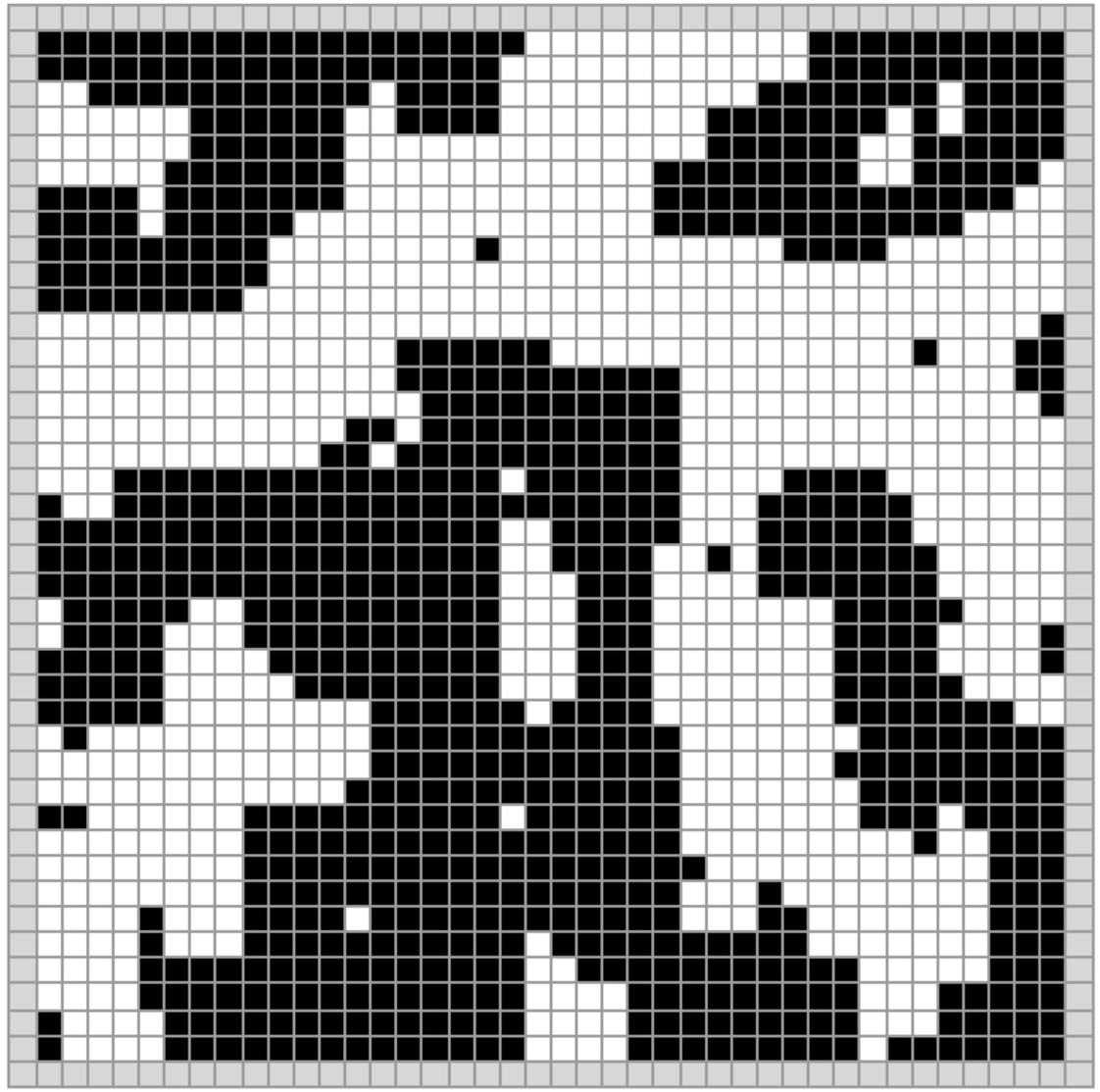
PCA projections often capture the **large-scale structure** of high-dimensional datasets

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/The2DIsingModelMonteCarloSimulationUsingTheMetropolisAlgorit/

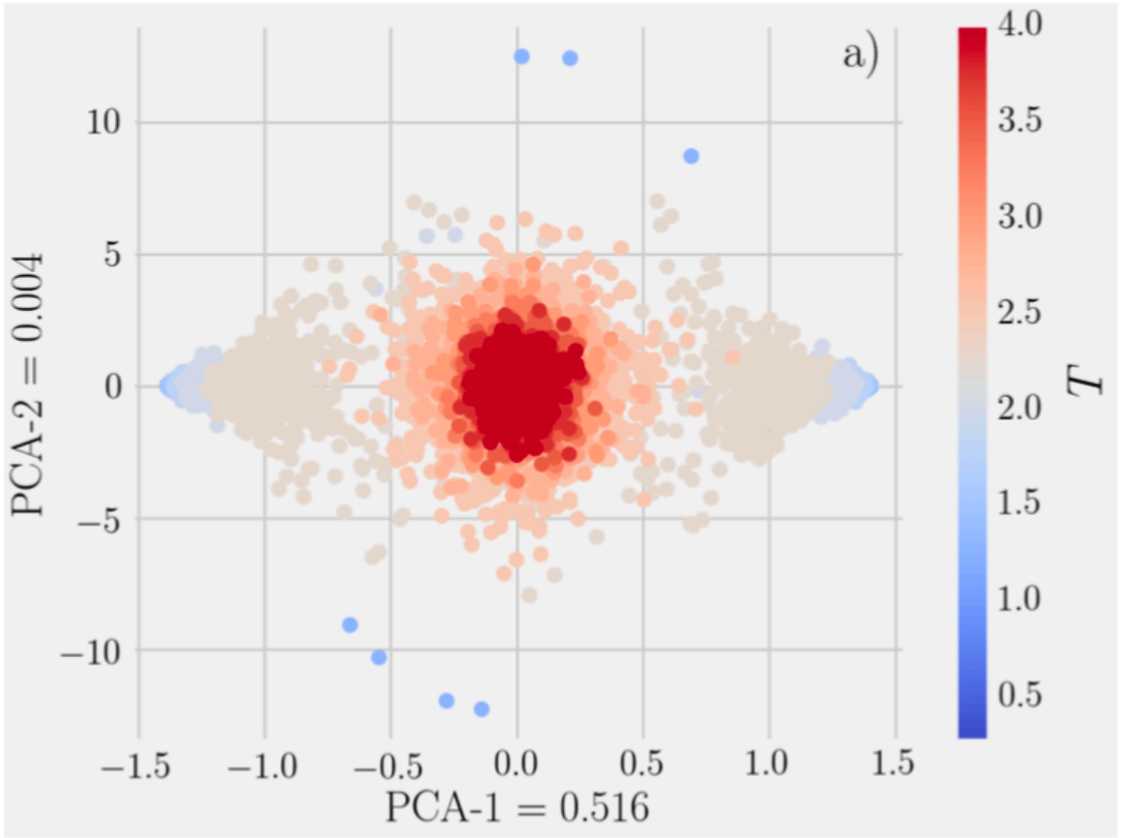
Principal Component Analysis

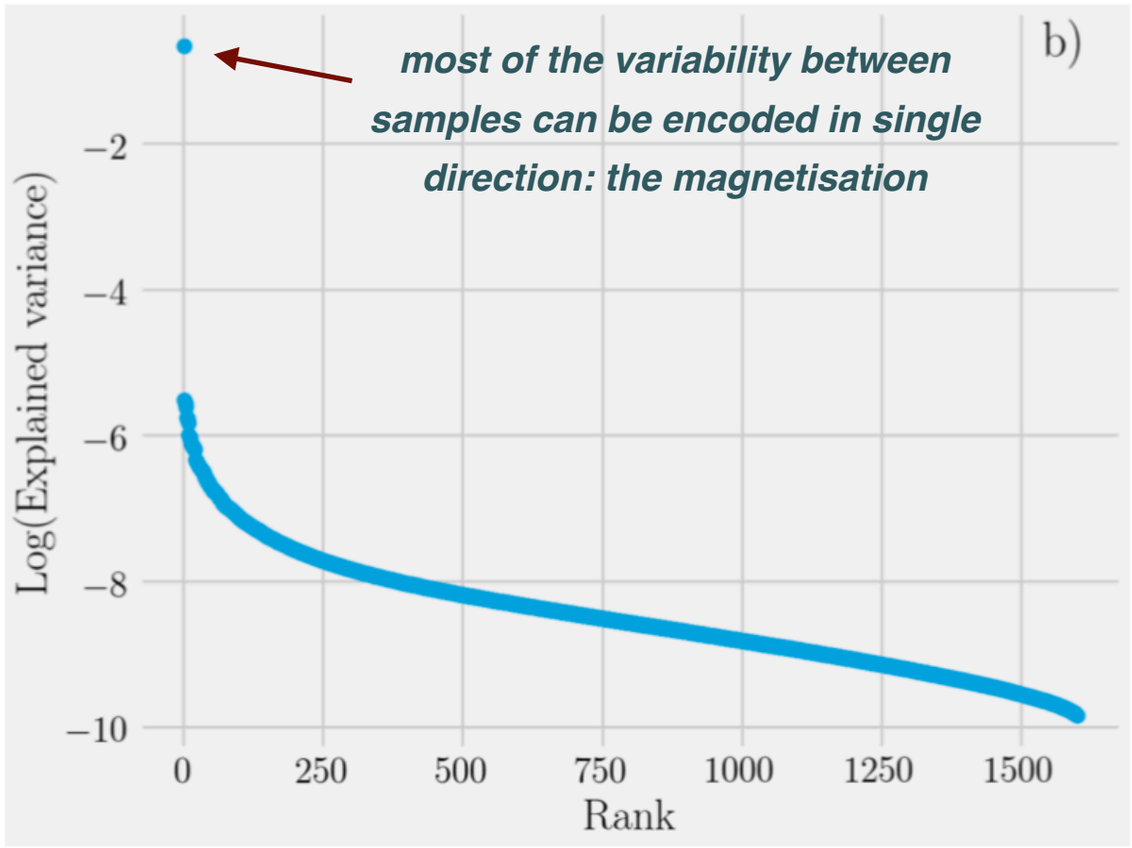
PCA projections often capture the **large-scale structure** of high-dimensional datasets

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



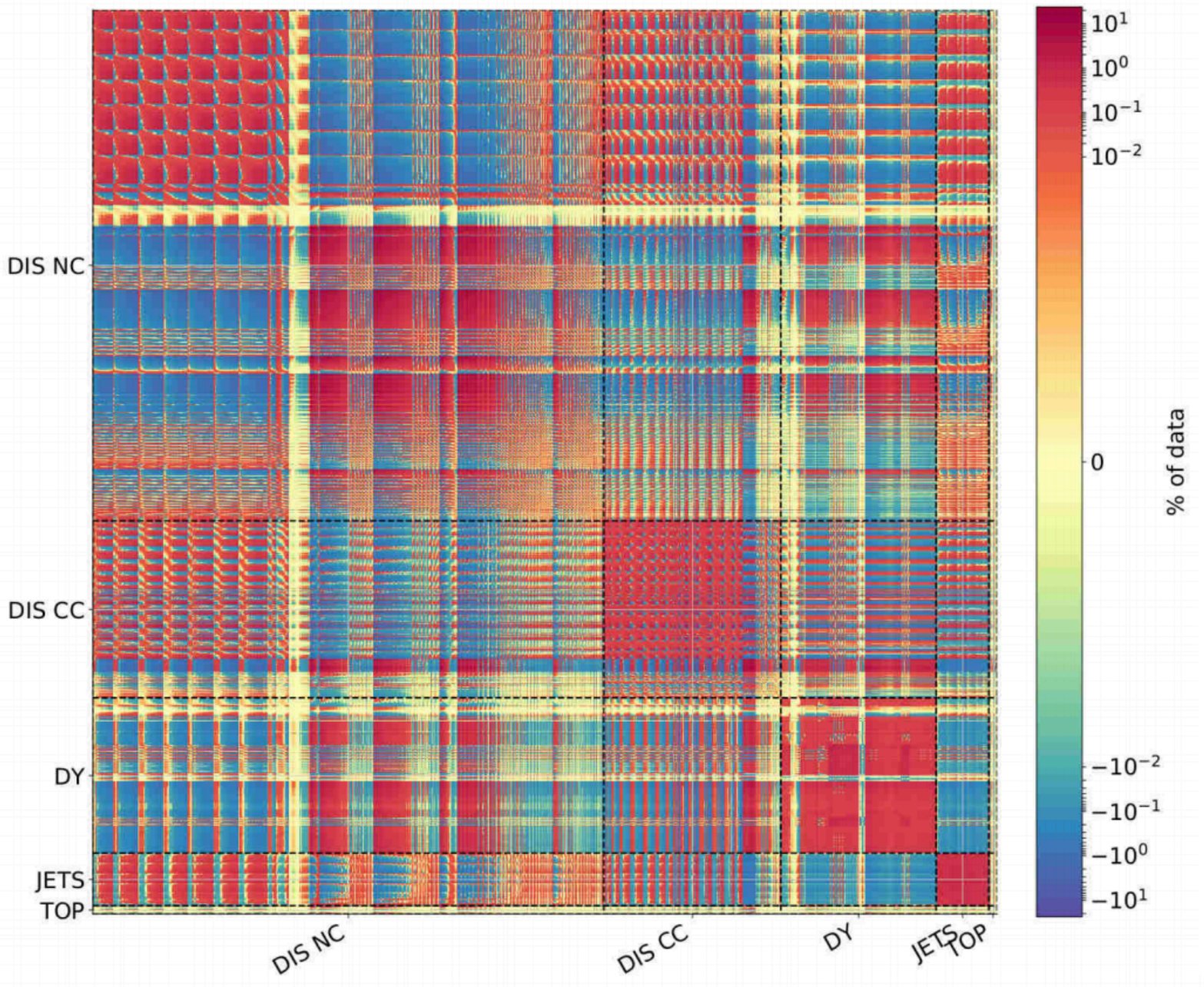


The first principal component accounts for > 50% of total variability!

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

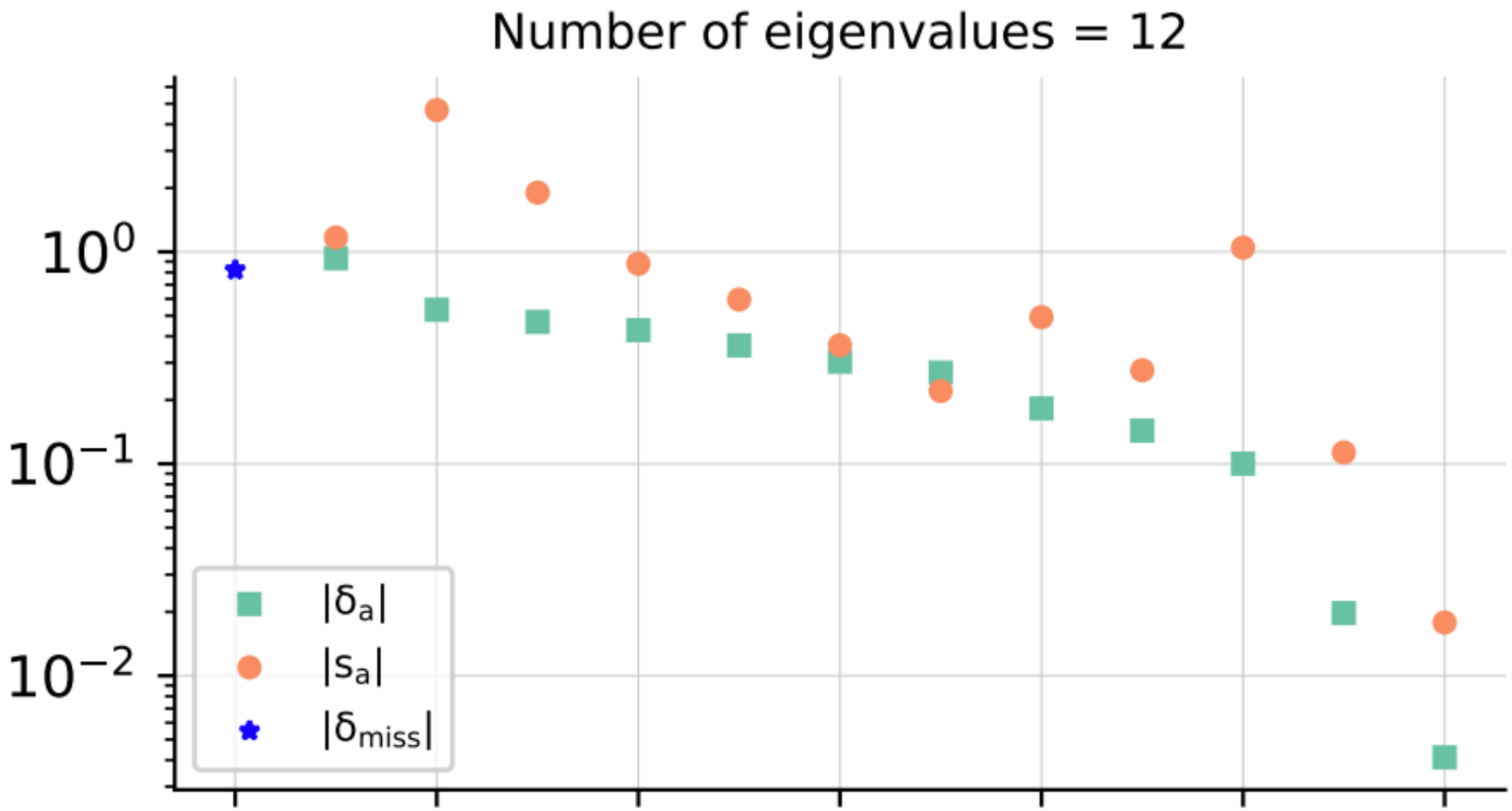
assume that some gives you the **covariance matrix of theory errors** of a global PDF fit (3000-dimensional space!). What are the **relevant components**?



Process Type	Dataset	Reference	$N_{\rm dat}$	$N_{\rm dat}$ (total)
DIS NC	NMC	[28,29]	134	1593
	SLAC	[30]	12	
	BCDMS	[31,32]	530	
	HERA σ^{p}_{NC}	[36]	886	
	HERA σ^{c}_{NC}	[37]	31	
DIS CC	NuTeV dimuon	[33,34]	41	552
	CHORUS	[35]	430	
	HERA σ^{p}_{CC}	[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^{ll}_T, M_{ll})	[46]	44	
	ATLAS Z p_T 8 TeV (p^{ll}_T, 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^{ll}_T, M_{ll})	[55]	28	
	LHCb Z 940 pb	[60]	9	
	LHCb $Z \to ee$ 2 fb	[61]	17	
	LHCb $W,Z \to \mu$ 7 TeV	[62]	29	
	LHCb $W,Z \to \mu$ 8 TeV	[63]	30	
	CDF Z rap	[38]	29	
	D0 Z rap	[39]	28	
	D0 $W \to e\nu$ asy	[40]	8	
	D0 $W \to \mu\nu$ asy	[41]	9	
JET	ATLAS jets 2011 7 TeV	[47]	31	164
	CMS jets 7 TeV 2011	[56]	133	
TOP	ATLAS $\sigma^{\rm top}_{tt}$	[48,49]	3	26
	ATLAS $t\bar{t}$ rap	[50]	10	
	CMS $\sigma^{\rm top}_{tt}$	[57,58]	3	
	CMS $t\bar{t}$ rap	[59]	10	
Total			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**?



no knowledge of underlying physics required

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
- ☑ On 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

Data Space

$$p_{i|j} = \frac{\exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 / 2\sigma_i\right)}{\sum_{k \neq i} \exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_k||^2 / 2\sigma_i\right)} \qquad \boldsymbol{x}_i \in \mathbb{R}^p$$

probability that j is neighbour of I

model parameter

t-SNE

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

Data Space

$$p_{i|j} = \frac{\exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma_i\right)}{\sum_{k\neq i}\exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_k||^2/2\sigma_i\right)} \qquad \boldsymbol{x}_i \in \mathbb{R}^p$$

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

Latent Space

$$q_{i|j} = \frac{\left(1+||\boldsymbol{y}_i - \boldsymbol{y}_j||^2\right)^{-1}}{\sum_{k\neq i}\left(1+||\boldsymbol{y}_i - \boldsymbol{y}_k||^2\right)^{-1}} \qquad \boldsymbol{y}_i \in \mathbb{R}^{p'}, \quad p' < p$$

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

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

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(**x**)* and *q(**x**)* plays an important role in machine learning applications

$$D_{KL}\left(p\,||\,q\right) = \int d\boldsymbol{x}\, p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \qquad D_{KL}\left(q\,||\,p\right) = \int d\boldsymbol{x}\, q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}$$

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

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

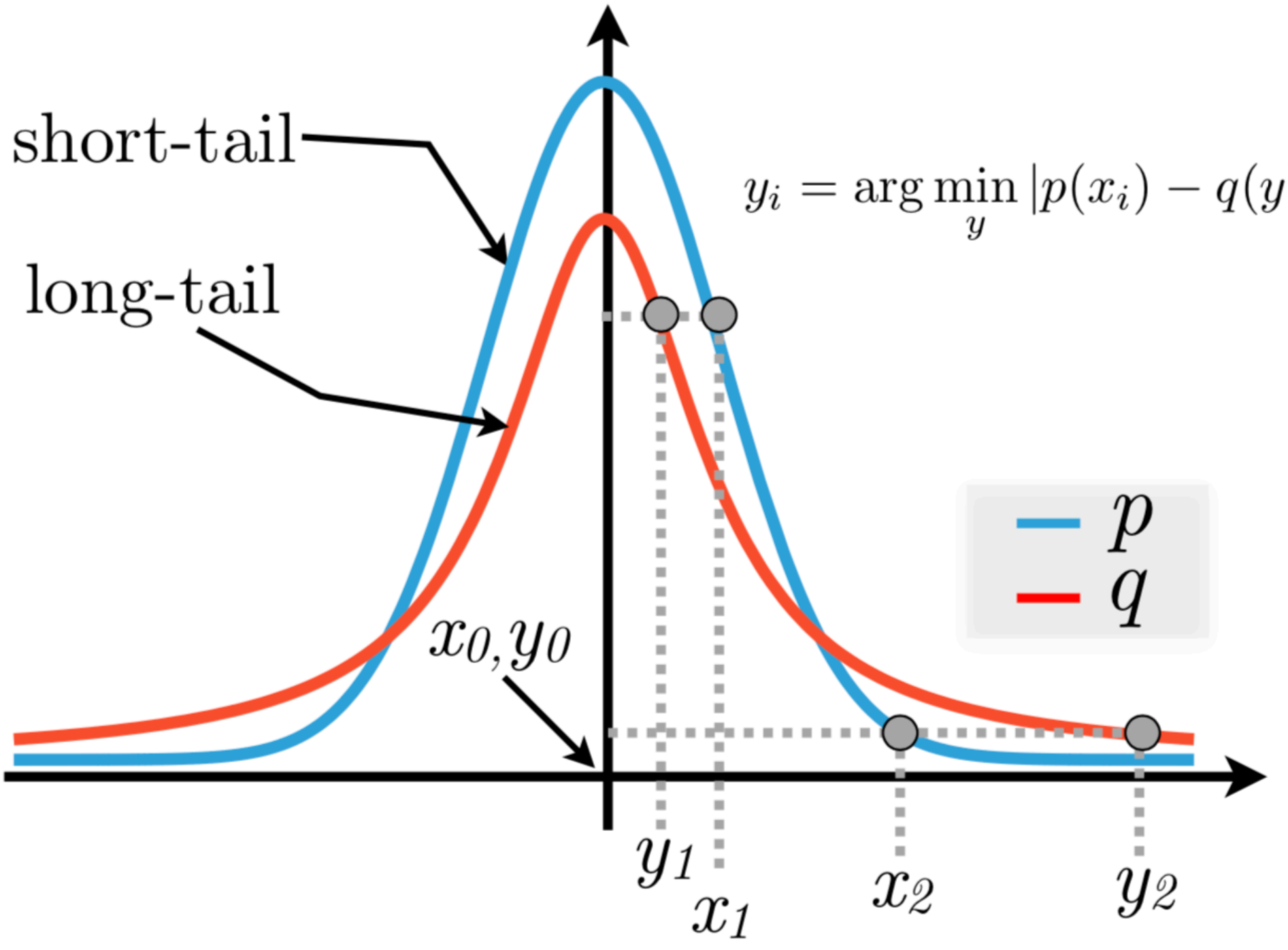
Jensen-Shannon divergence

The KL-divergence is **positive-definite**, and only vanishes when *p(**x**)*= q*(**x**)*

$$D_{KL}\left(p\,||\,q\right) \geq 0$$

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

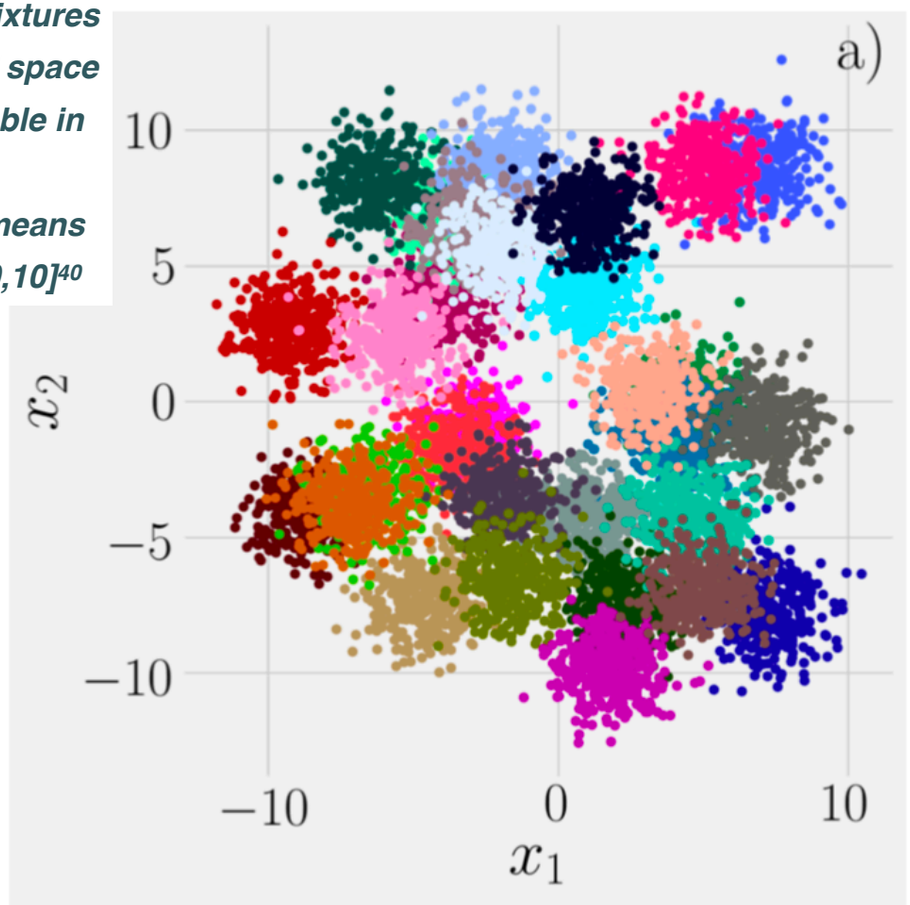
t-SNE



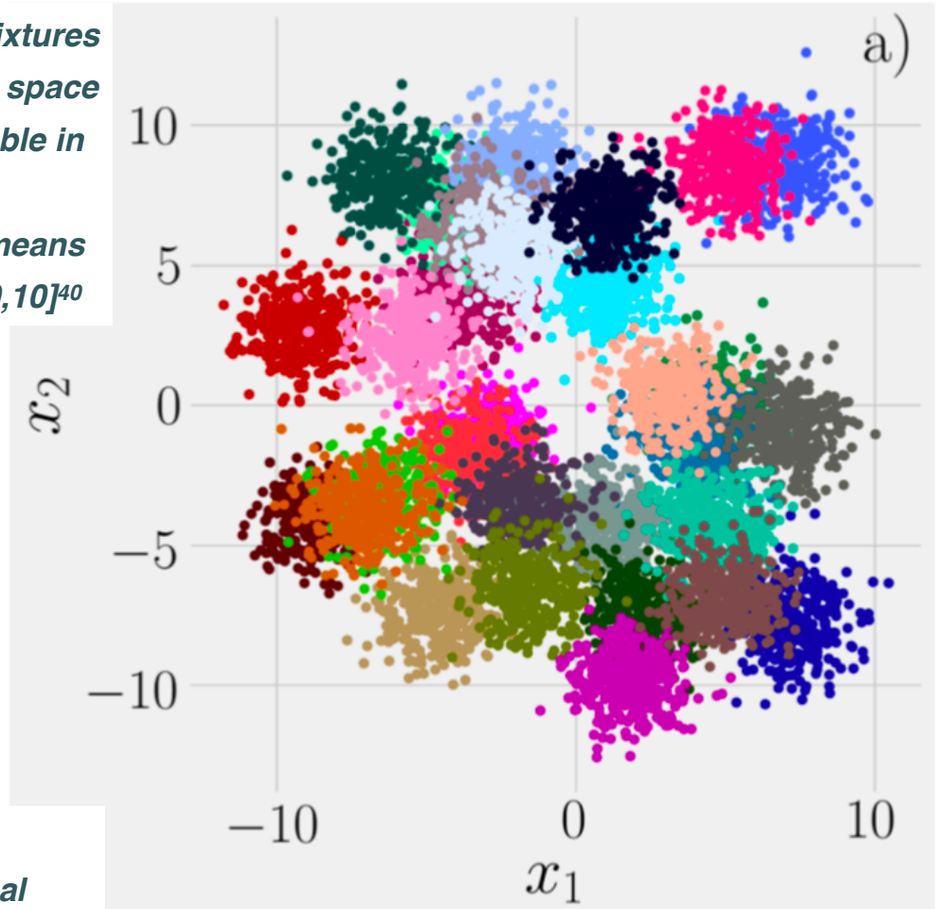
Note that *q* is a **long-tail distribution** (Cauchy): this preserves short distance information while repelling two points that are far apart in the original space

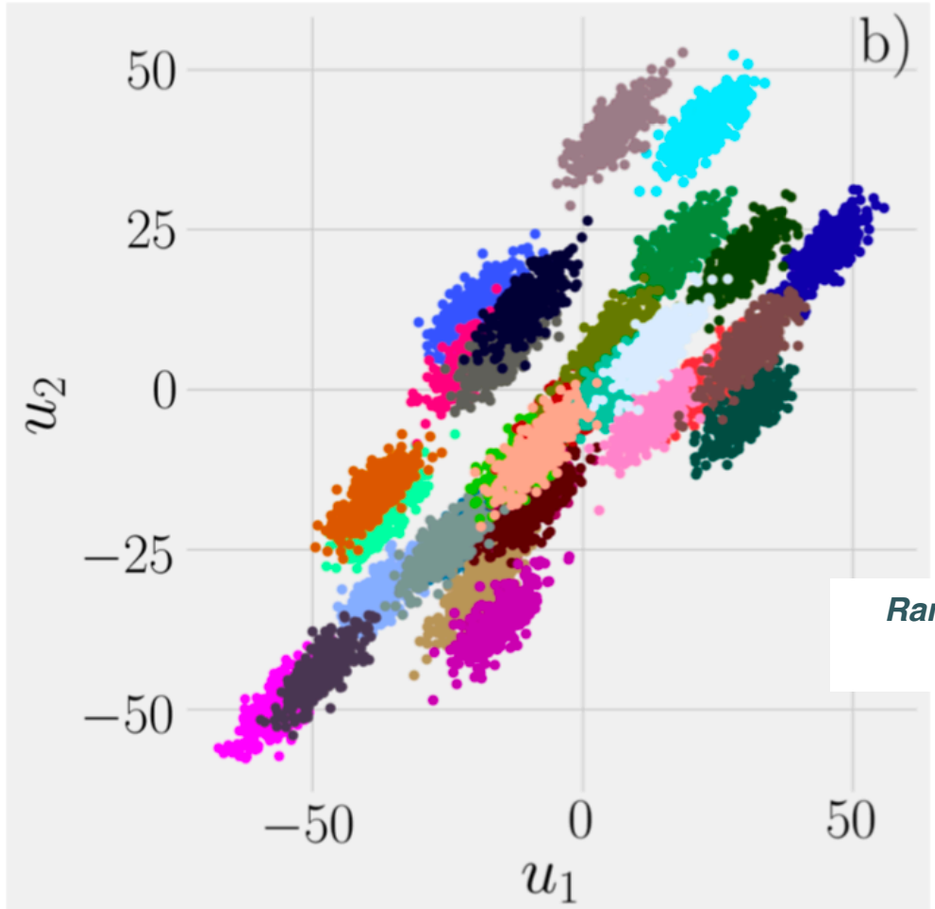
K=30 Gaussian mixtures in 40-dimensional space (labels not available in real case!)

Same variance, means at random in $[-10,10]^{40}$



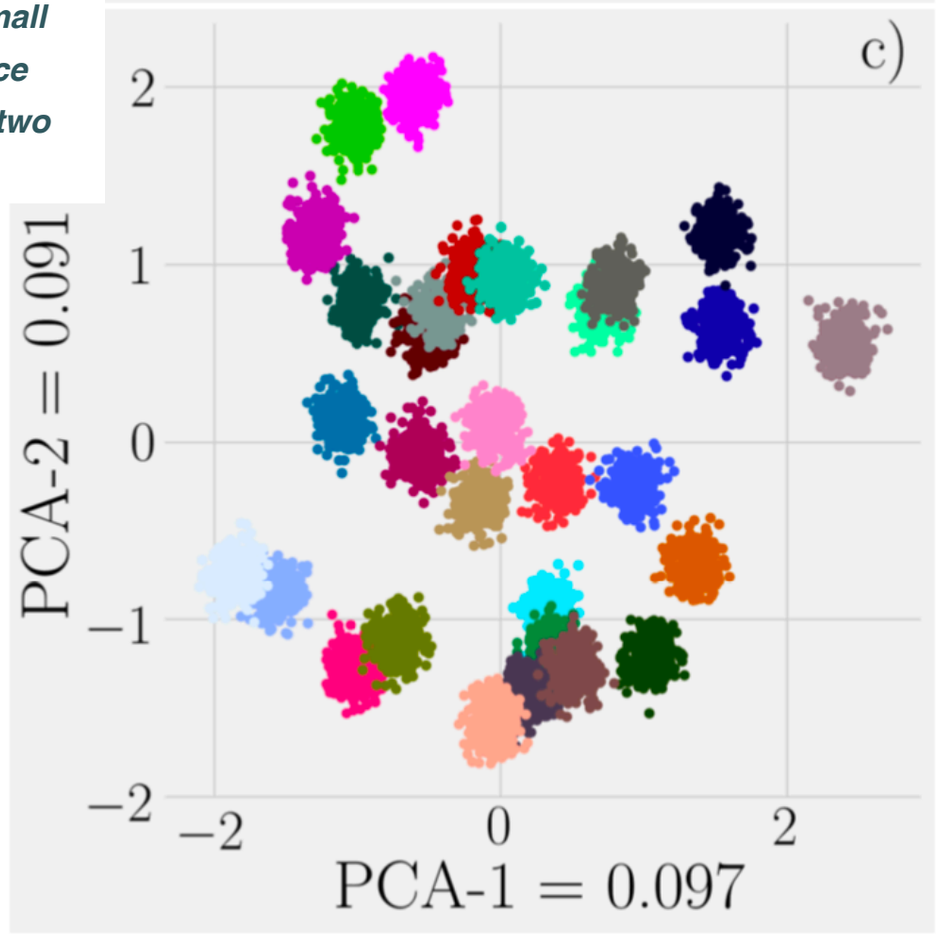
K=30 Gaussian mixtures in 40-dimensional space (labels not available in real case!)
Same variance, means at random in $[-10,10]^{40}$

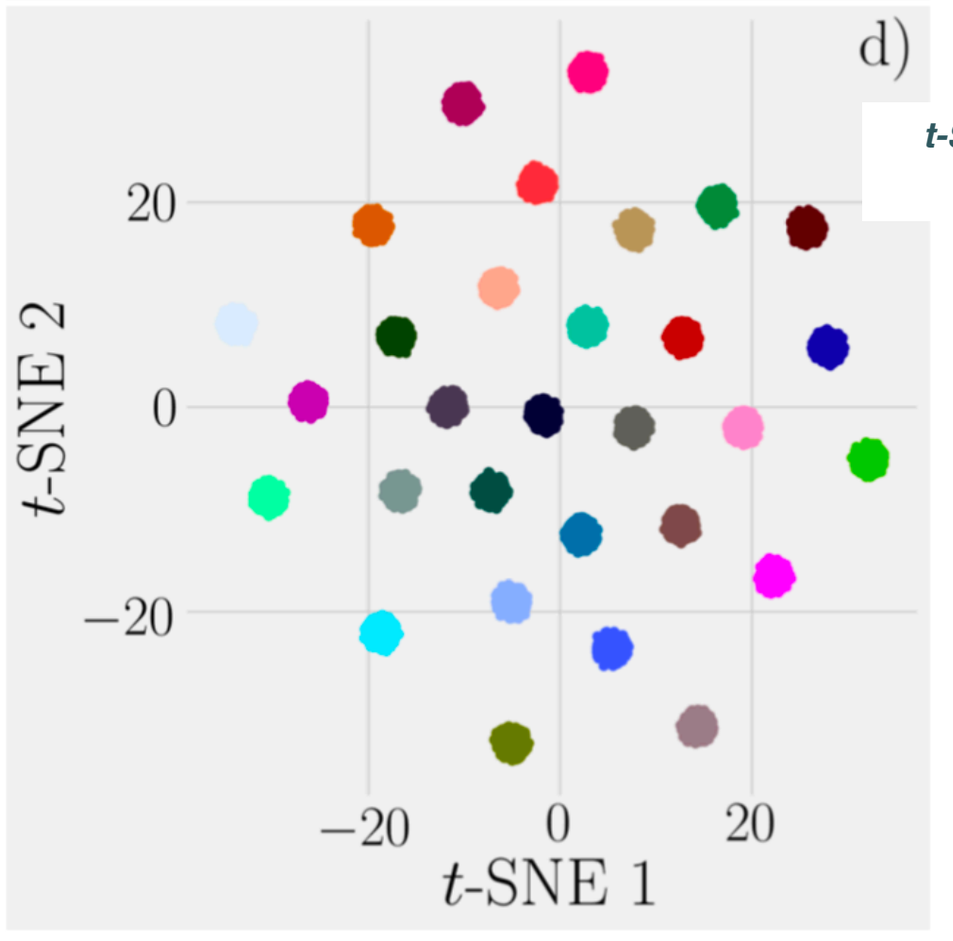




Random projection to 2D space

PCA projection (linear dimensional reduction): only small fraction of variance accounted by first two components





t-SNE embedding to 2D latent space

Tutorial #4

Two-dimensional Ising Model

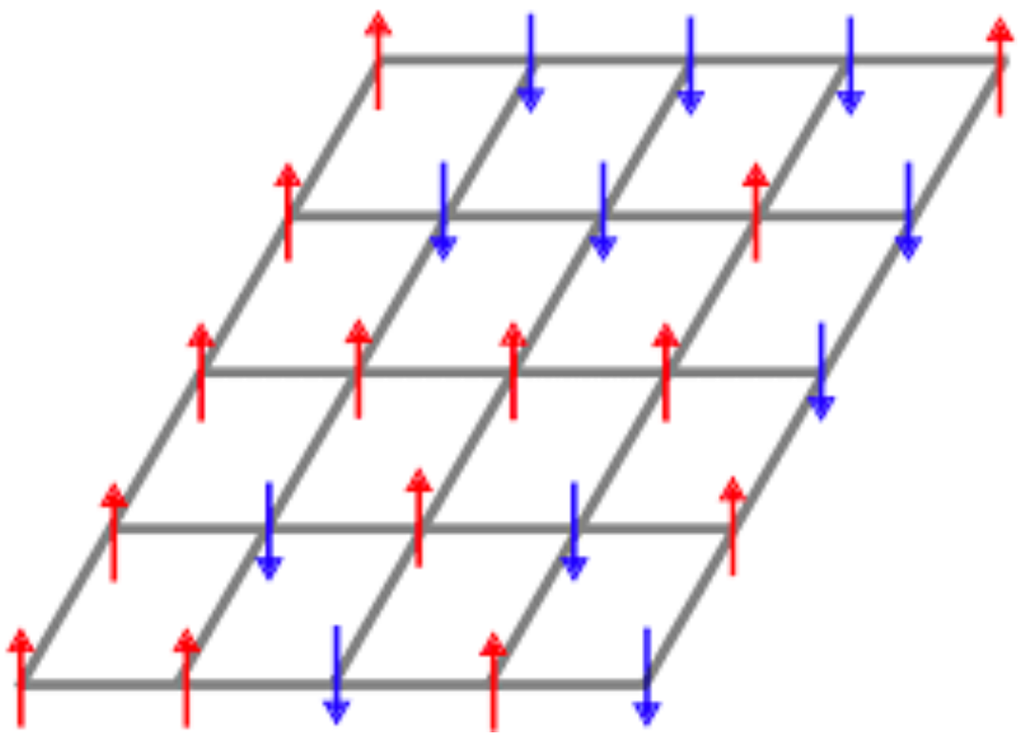
☑ Model a ferromagnetic solid by a **discrete lattice of spins**, with Hamiltonian

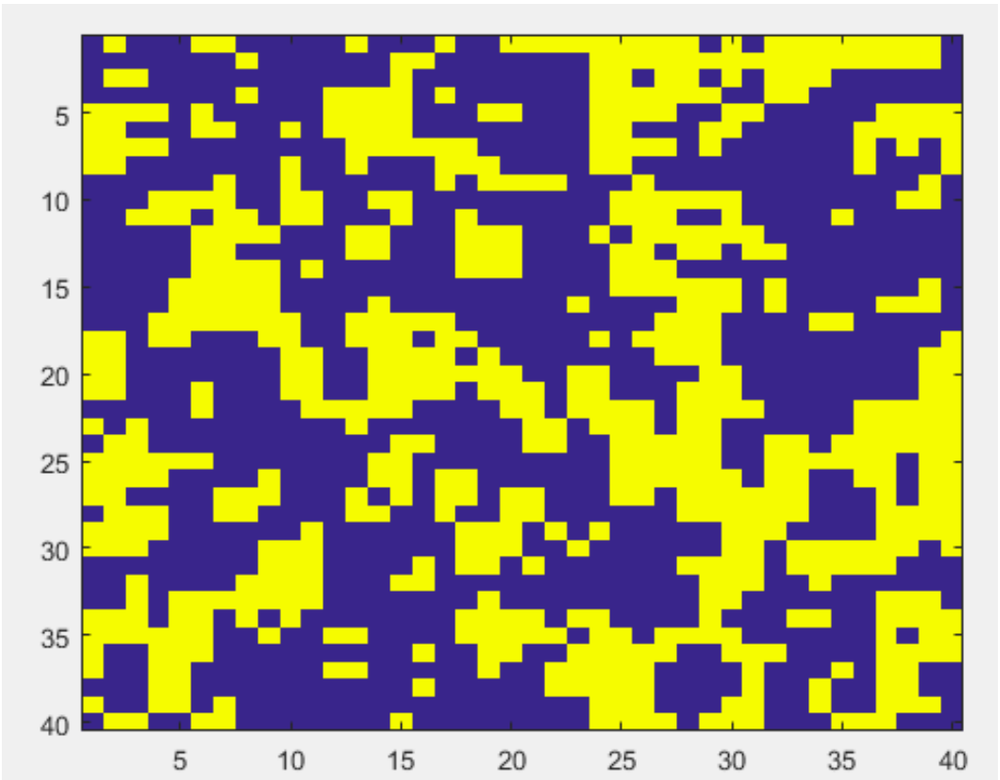
$$H\left(\{\sigma\}\right) = -\sum_{\langle i,j\rangle} J_{ij}\sigma_i\sigma_j - \mu\sum_{j} h_j\sigma_j$$

interaction between nearest-neighbour spins ***interaction with external magnetic field***

☑ We will try to understand the **underlying physical law** without actually solving the system, rather by using Monte Carlo generated samples

☑ Deploy **unsupervised ML methods**: PCA, K-means clustering, t-SNE





Summary

☑ Unsupervised machine learning provides useful approaches to **identify features** in unlabelled data

☑ These features can be used e.g. to group ``similar'' data points into clusters, or to detect ``anomalies'' that do not fit the general patterns

☑ In general the dimensionality of **latent space** (intrinsic dimensionality) << data dimensionality. Can use dimensionality reduction methods to gain insight on the problem

☑ Working directly in the latent space offers many advantages, both practical and conceptual

☑ PCA and t-SNE are two popular methods in ML for **dimensionality reduction**