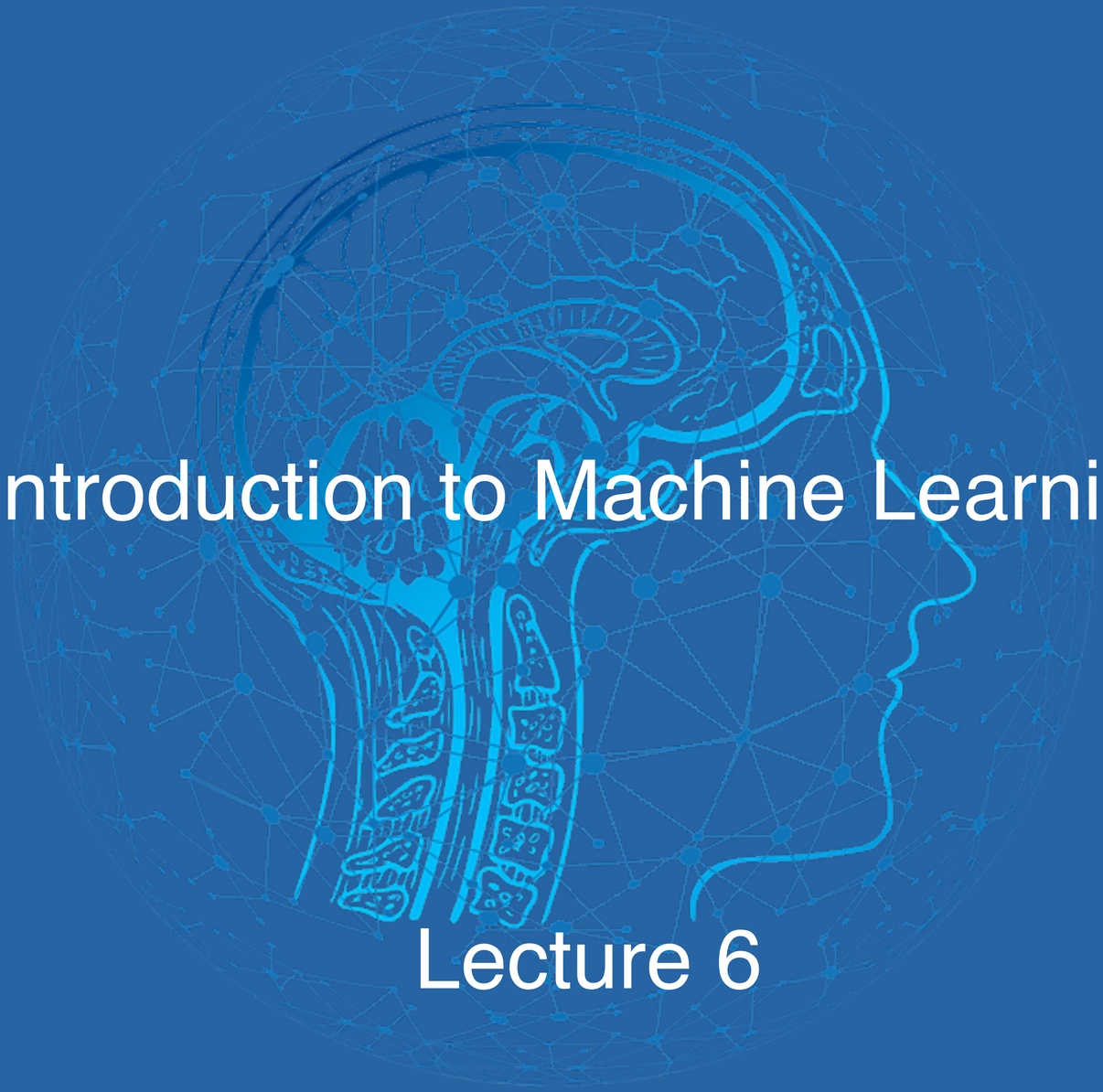
Imperial College London



Introduction to Machine Learning

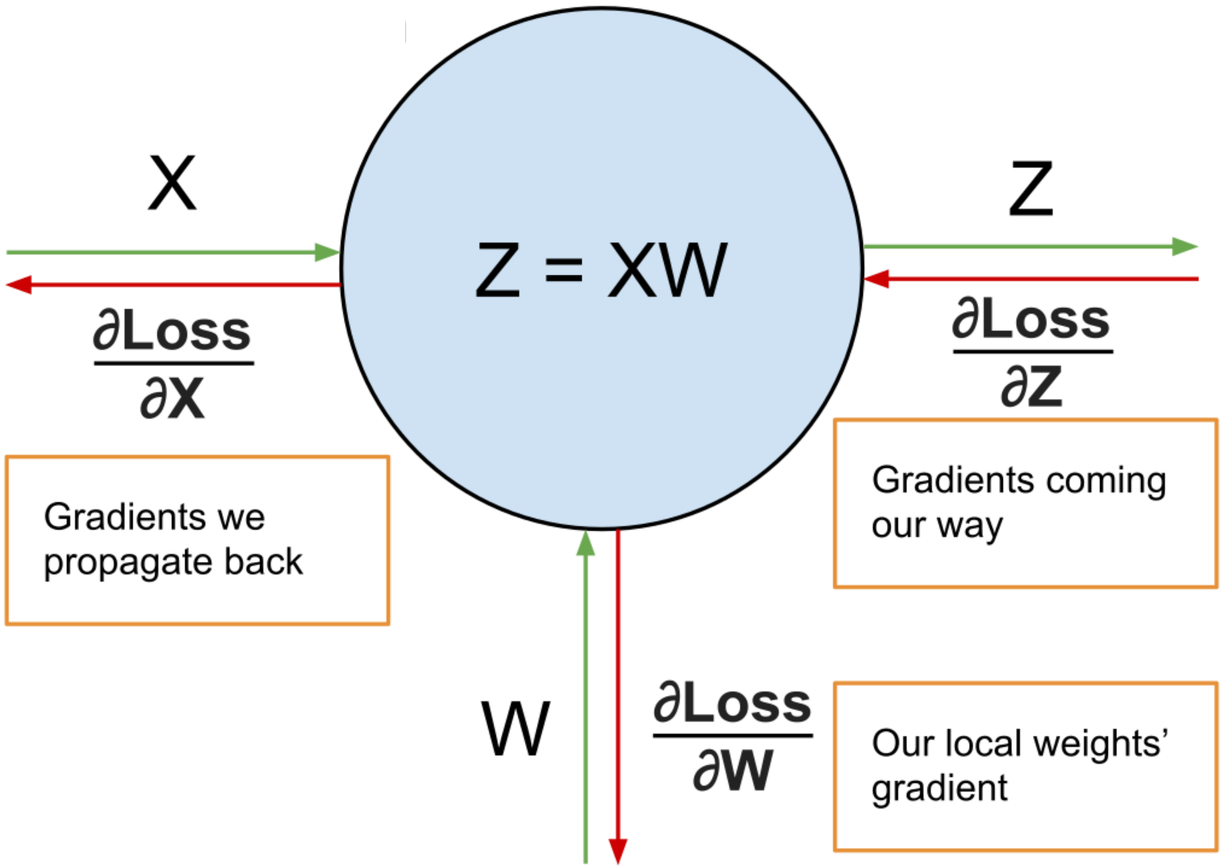
Lecture 6

Antoine Cully & Marek Rei & Josiah Wang

In the previous lecture...

Backpropagation

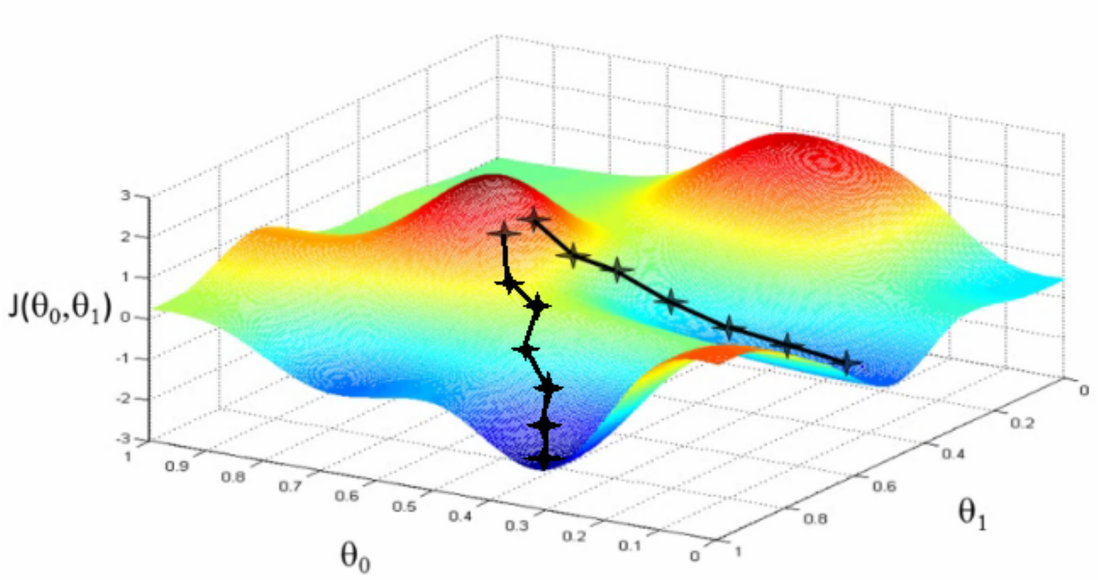
Backpropagation: Efficiently propagating the gradients backwards through the network layers



Gradient descent

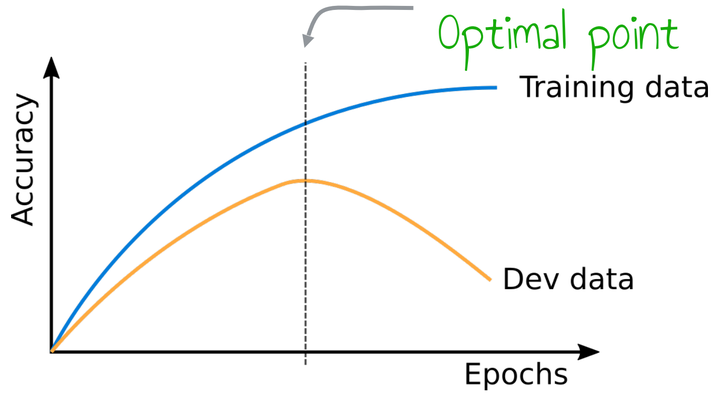
Gradient descent: Repeatedly updating parameters by taking small steps in the negative direction of the partial derivative, so the model gets better at predicting our data.

$$W = W - \alpha \frac{\partial L}{\partial W}$$



Avoiding overfitting in neural networks

Early stopping:

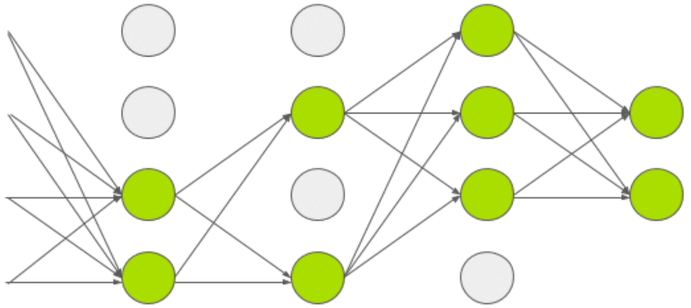


Regularisation (L1 and L2):

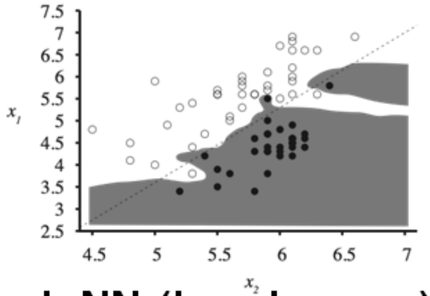
$$J(\theta) = Loss(y, \hat{y}) + \lambda \sum_{w} w^2$$

$$J(\theta) = Loss(y, \hat{y}) + \lambda \sum_{w} |w|$$

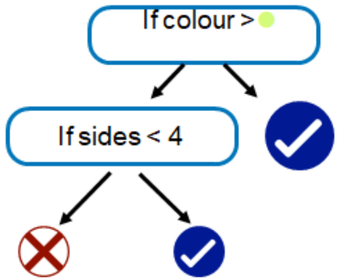
Dropout:



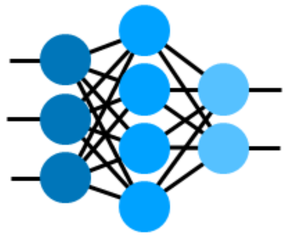
What models we learnt so far?



k-NN (lazy learner)



Decision Trees (eager learner)



Neural Networks (eager learner)

Approach family:

Supervised Learning

Tasks:

- **Classification**
- **Regression**

Today...

Course plan

	Lecture	Lecturer
Week 2	Introduction to ML	*Josiah*
Week 3	Instance-based Learning + Decision Trees	*Antoine*
Week 4	Machine Learning Evaluation	*Marek*
Week 5	Artificial Neural Networks I	*Marek*
Week 6	Artificial Neural Networks II	*Marek*
Week 7	Unsupervised Learning	*Antoine*
Week 8	Genetic Algorithms	*Antoine*

Today's lecture

9 videos:

1. This short intro

Clustering:

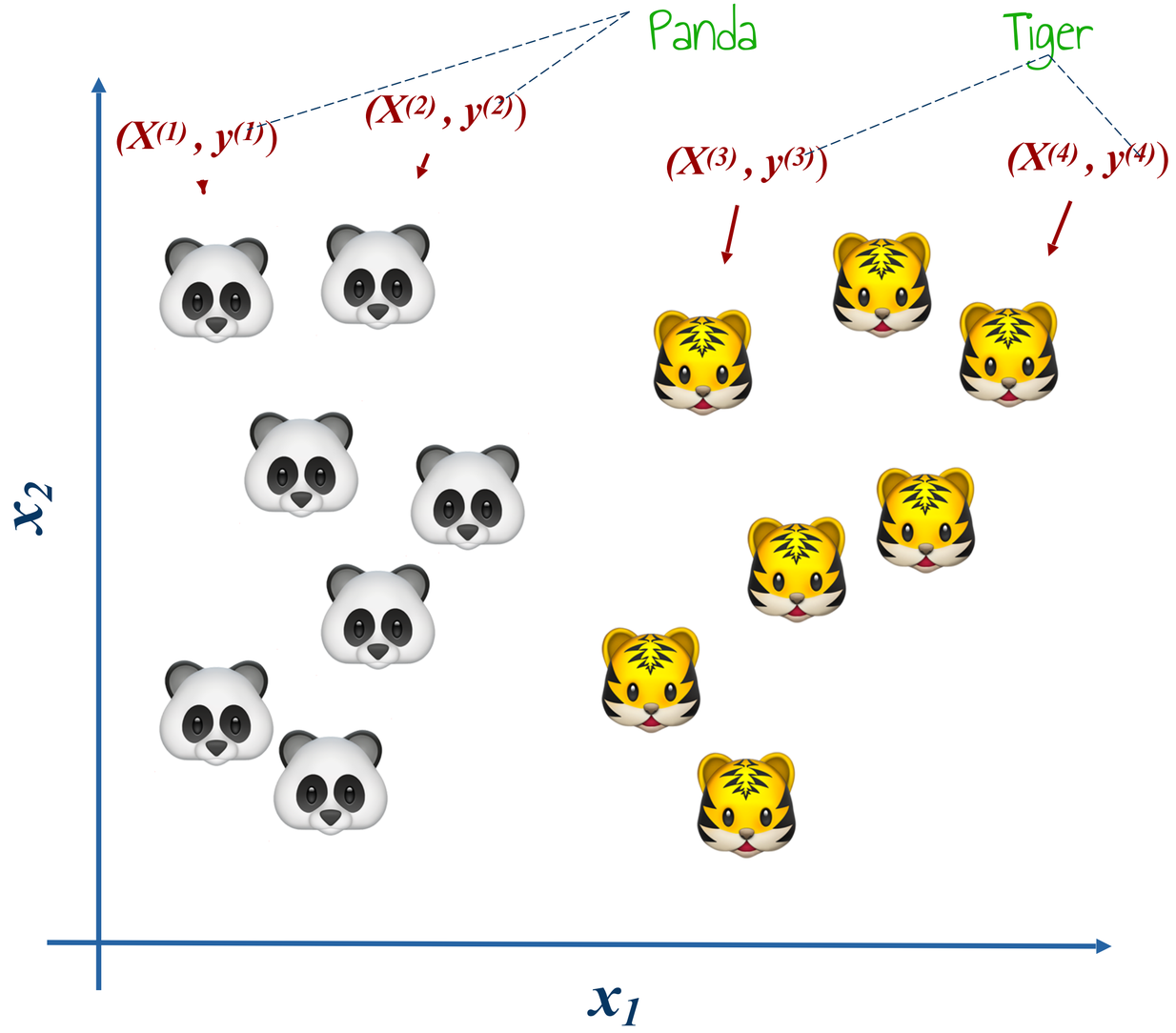
2. General concept of Clustering
3. Clustering with K-means
4. K-means: hyper-parameter tuning, strengths and weaknesses

Probability Density Estimation:

5. General concepts
6. Non-parametric approaches
7. Parametric approaches
8. GMM-EM
9. GMM-EM: hyper-parameter tuning, strengths and weaknesses

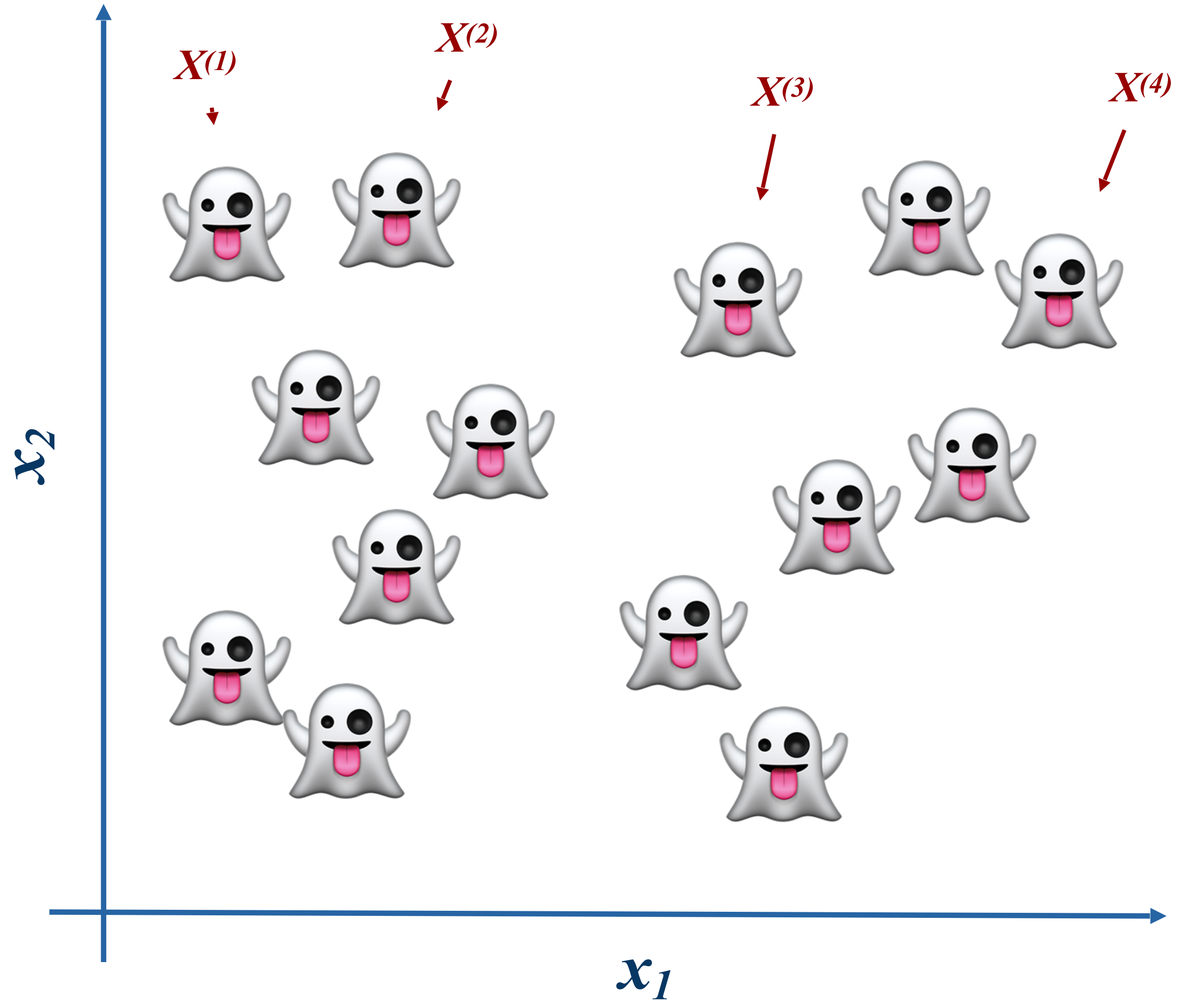
Unsupervised Learning

Supervised learning



Training data: $\{(X^{(1)}, y^{(1)}), (X^{(2)}, y^{(2)}), \ldots, (X^{(N)}, y^{(N)})\}$

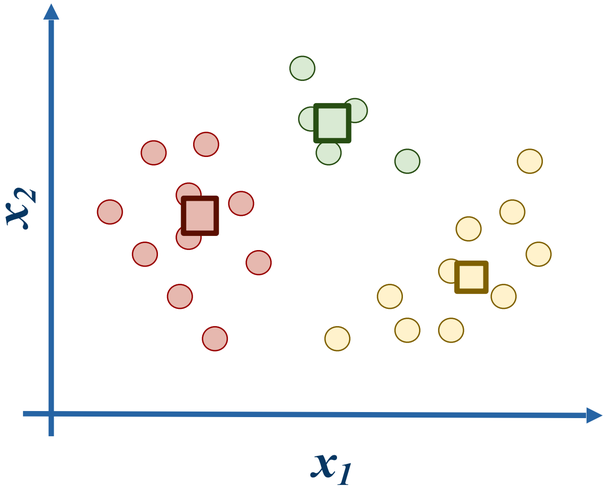
Unsupervised learning



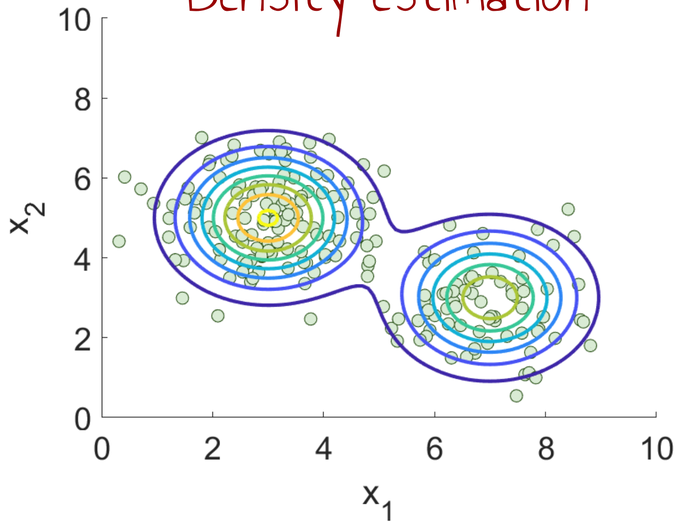
Training data: $\{X^{(1)}, X^{(2)}, \ldots, X^{(N)}\}$

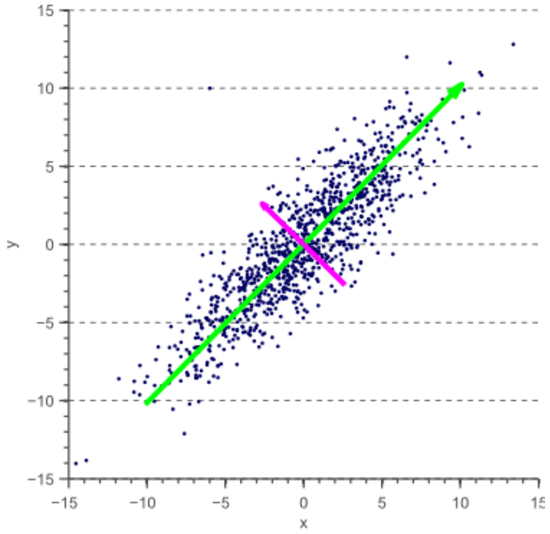
Examples of Unsupervised Learning

Clustering



Density estimation





Dimensionality reduction

- Principal Component Analysis
- t-SNE
- Autoencoder
- etc.

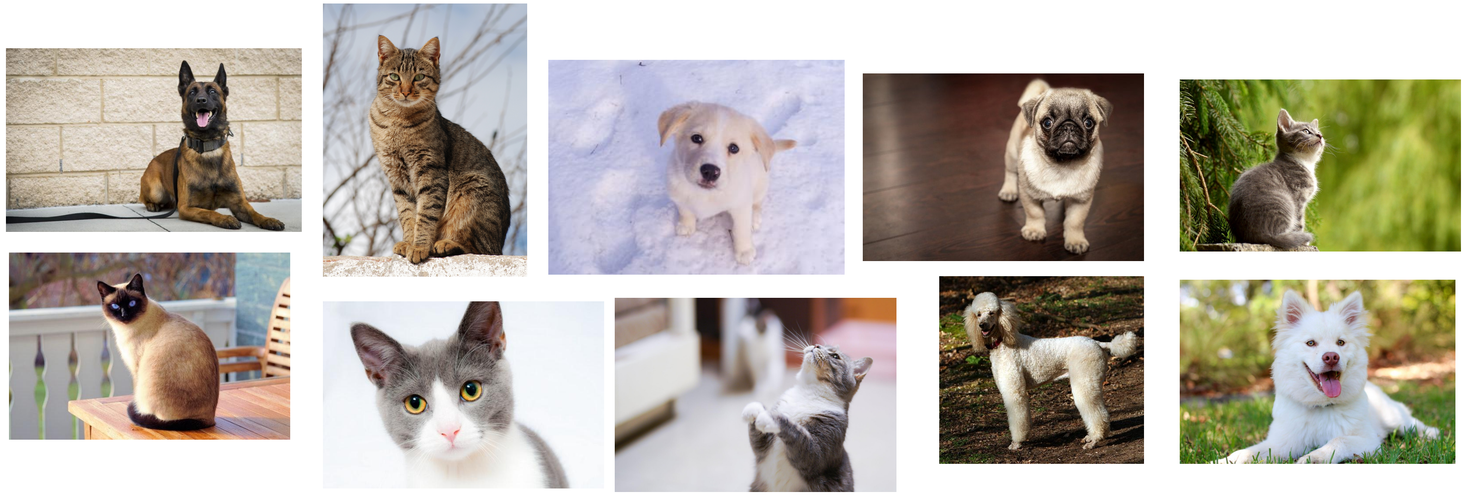
To be continued...
(Clustering)

Clustering

What is a cluster?

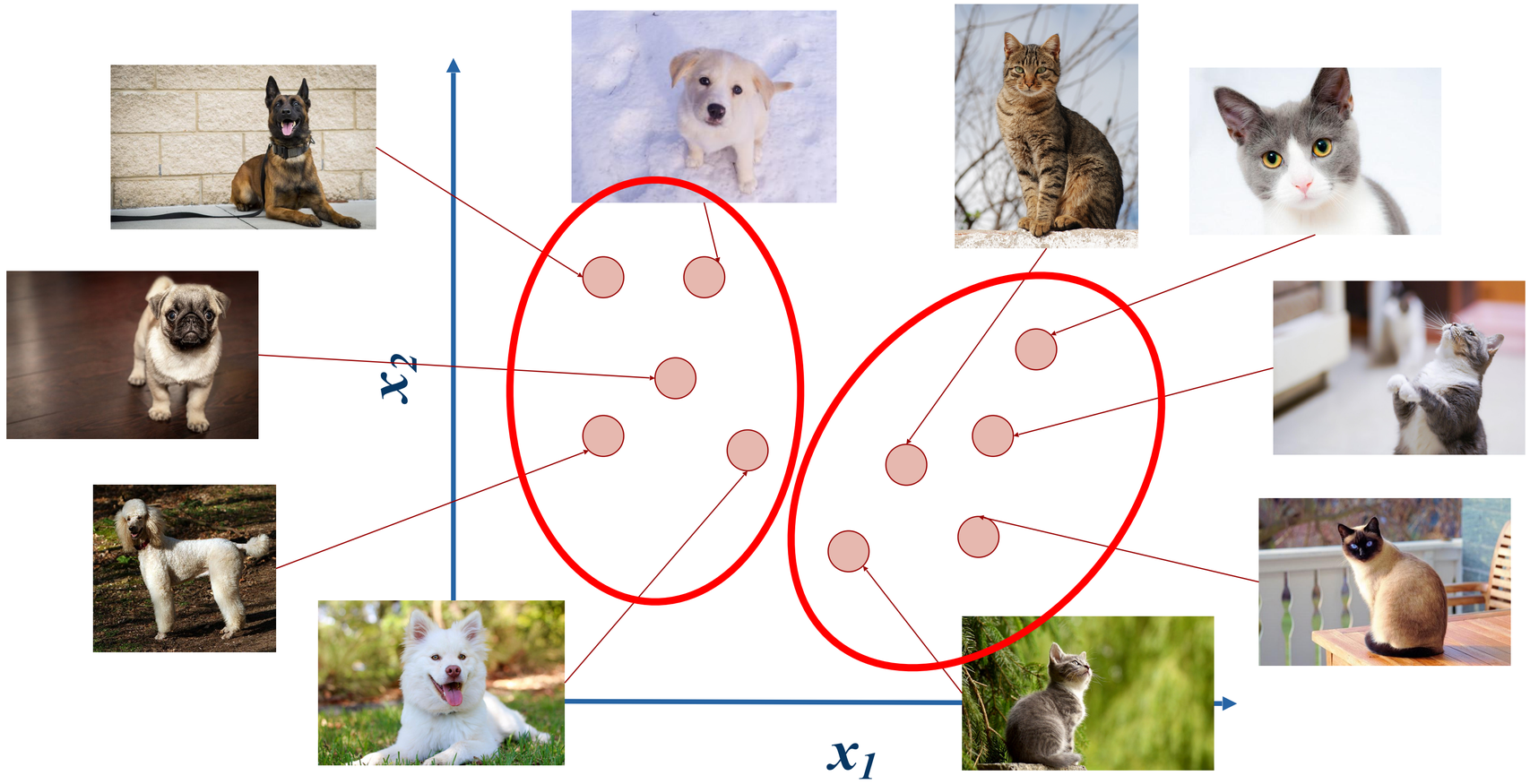
Low intra-cluster variance

- A cluster is a set of instances that are ‘similar’ to each other, and ‘dissimilar’ to instances in other groups/clusters.



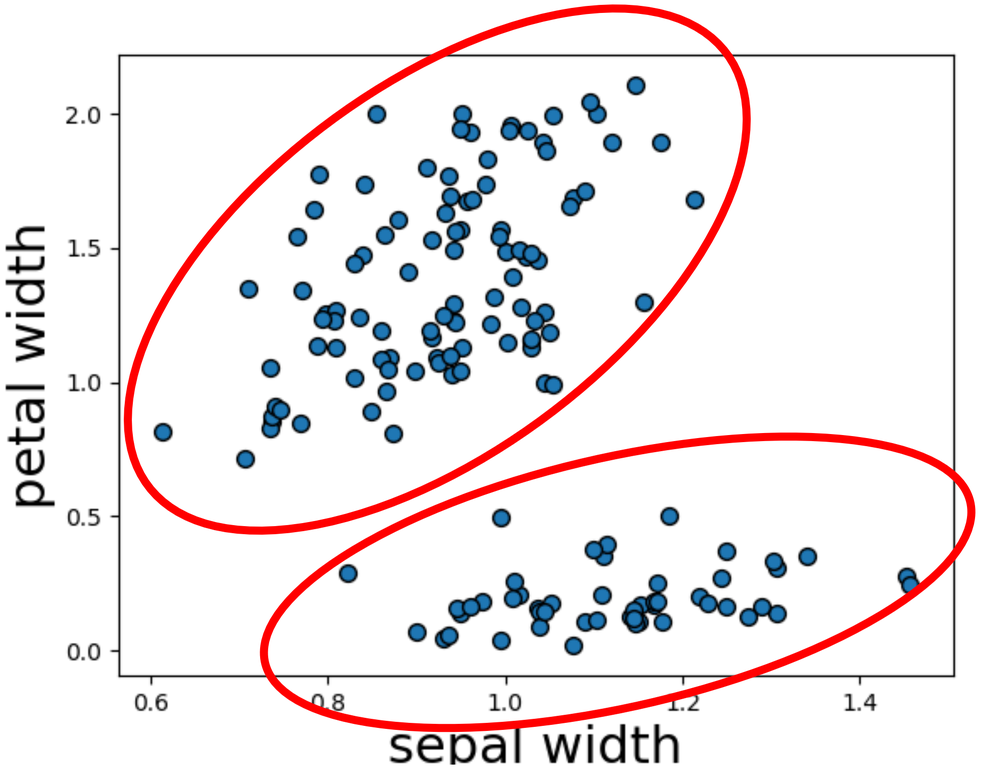
What is clustering?

- The task of grouping instances (in some feature space) together such that instances in the same group are more similar to each other than to other instances in other groups



Another example of clustering…

- You collected lots of sample flowers. They look similar yet different at the same time.
- Are there actually several species in there? Or are they all from the same species?



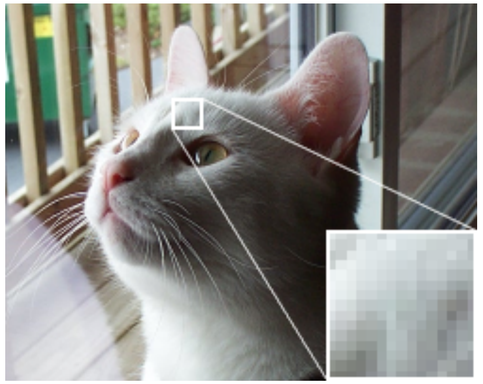
How to do this automatically?

Applications of Clustering

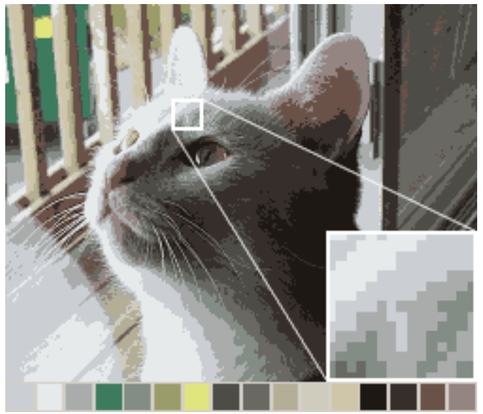
Vector Quantisation

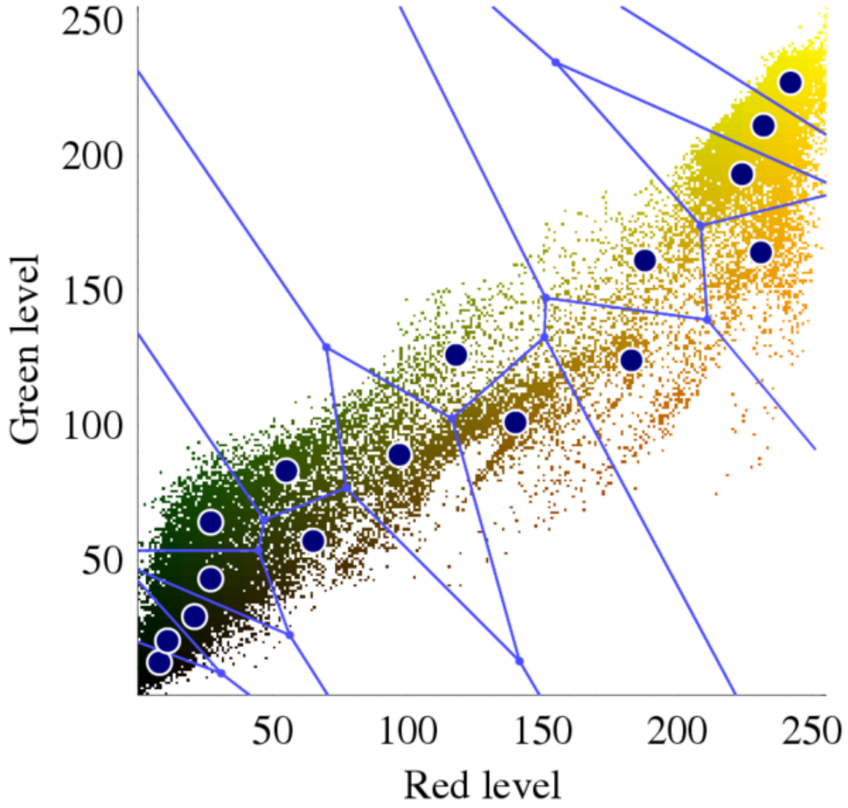
Group continuous features into prototypes

24bits color encoding



16 different colors (4 bits)



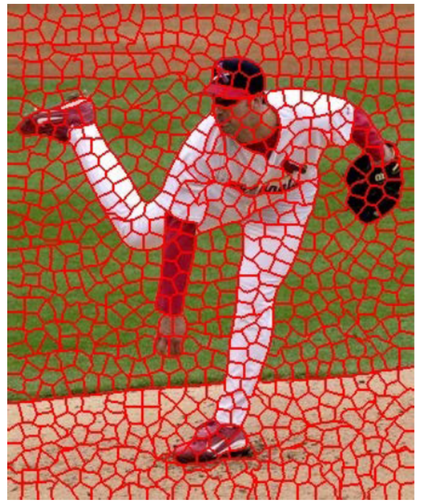


From Jamelan and Dcoetzee

Applications of Clustering

Vector Quantisation

Group continuous features into prototypes



Superpixels

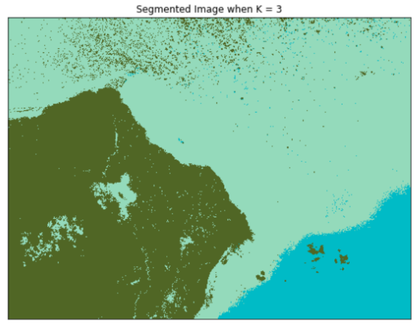
Group similar-looking pixels as intermediate stage of processing

"Bag of visual words": extract patches, map to dictionary of 'words', represent images as histograms of 'words'

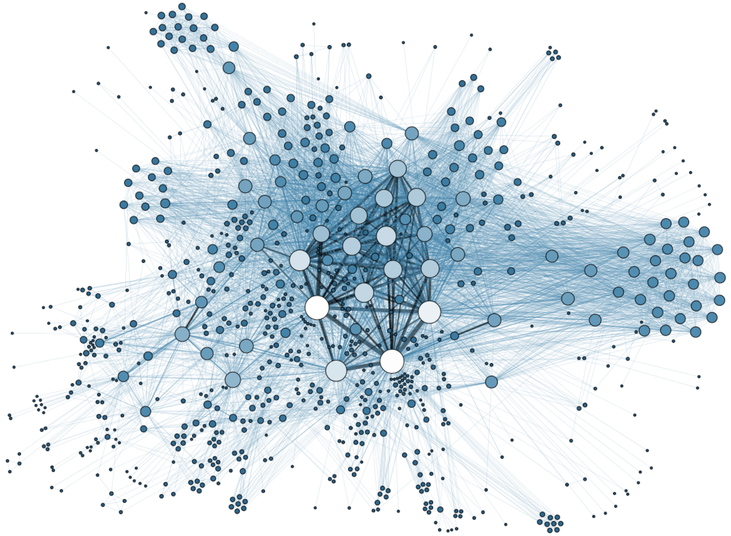
http://slazebni.cs.illinois.edu/spring19/lec20_recognition_intro.pdf
http://slazebni.cs.illinois.edu/spring19/lec24_segmentation.pdf

Applications of Clustering

Image segmentation



Social network analysis



Market segmentation

https://towardsdatascience.com/introduction-to-image-segmentation-with-k-means-clustering-83fd0a9e2fc3

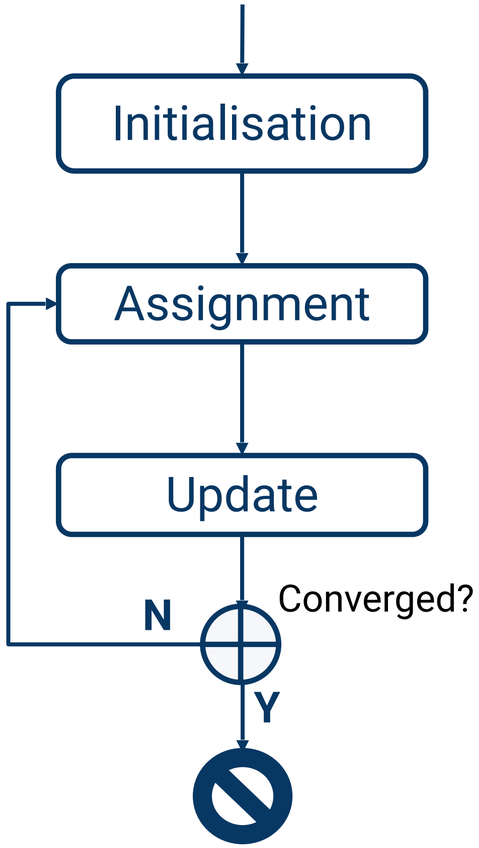
https://www.nutshell.com/blog/market-segmentation

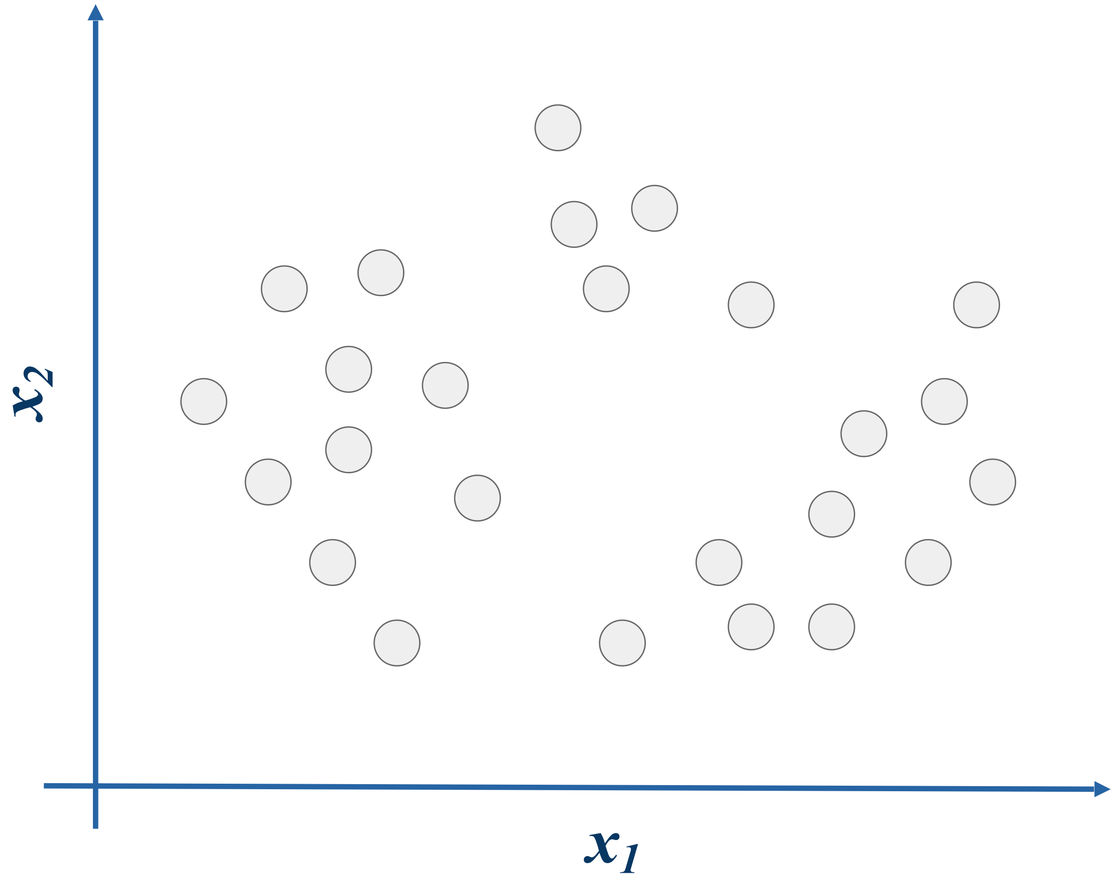
https://commons.wikimedia.org/wiki/File:Social_Network_Analysis_Visualization.png/

To be continued...
(K-means)

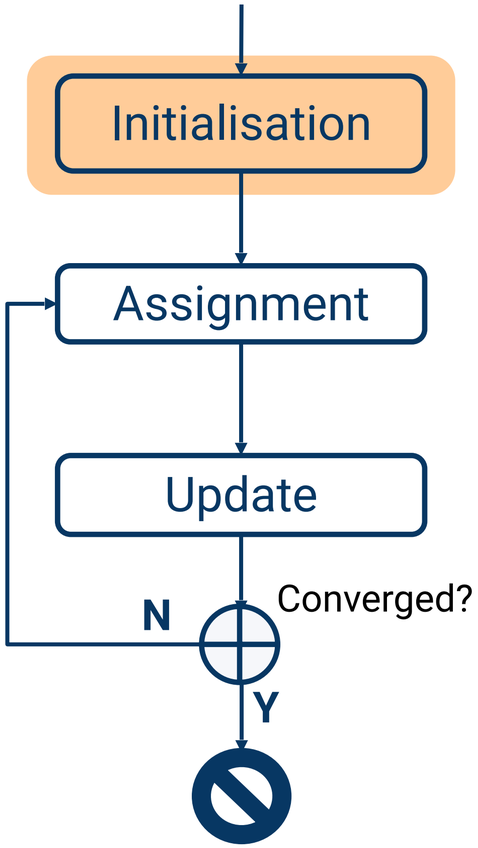
Clustering with K-means

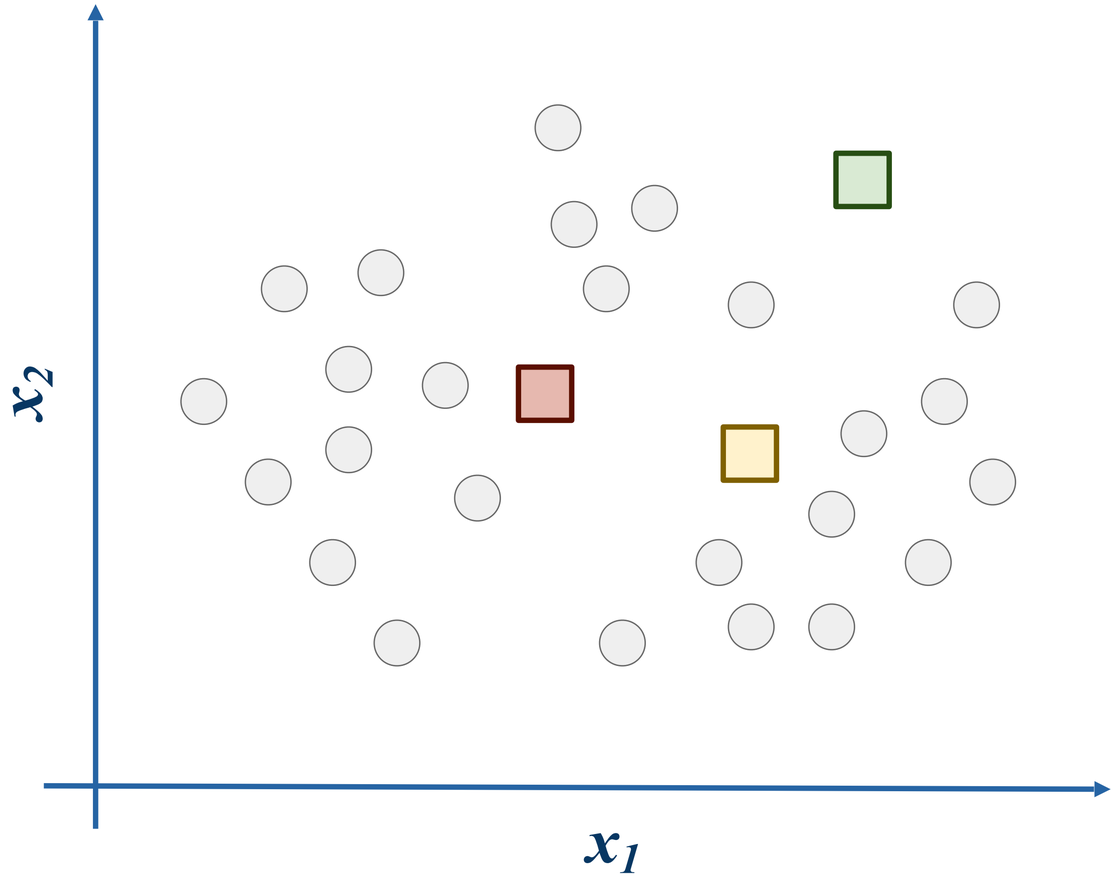
K-Means



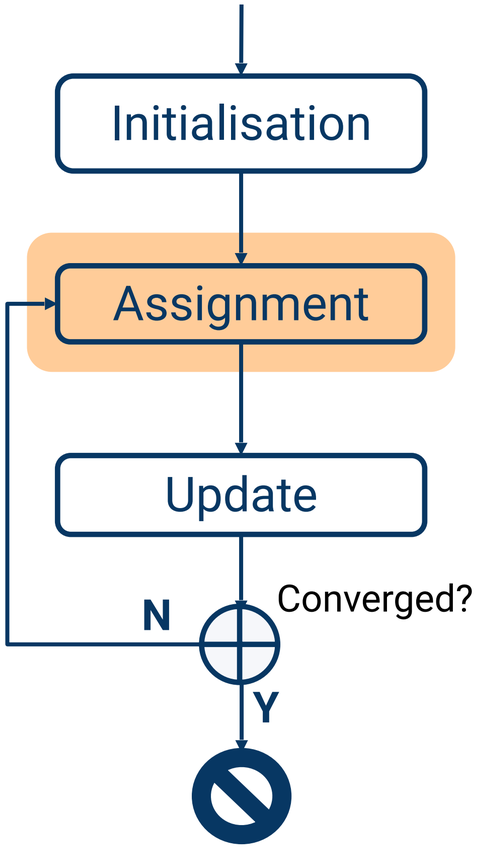


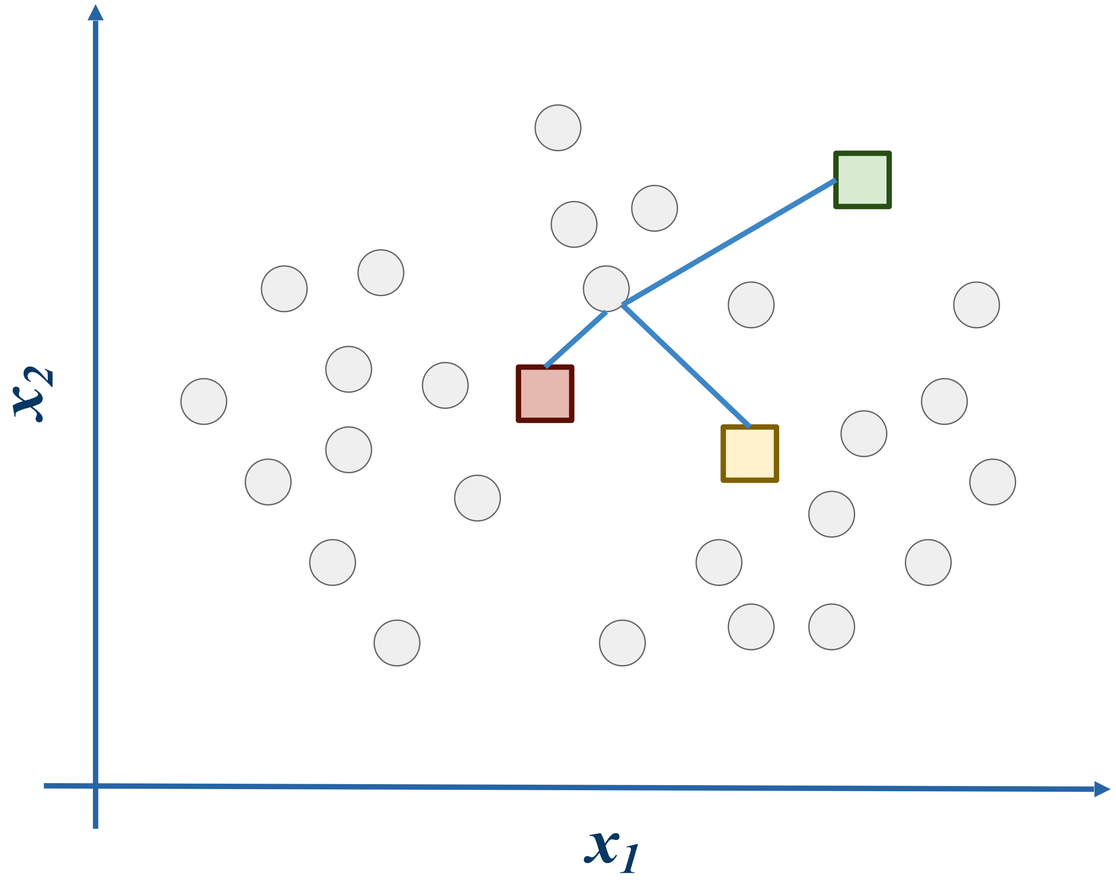
K-Means



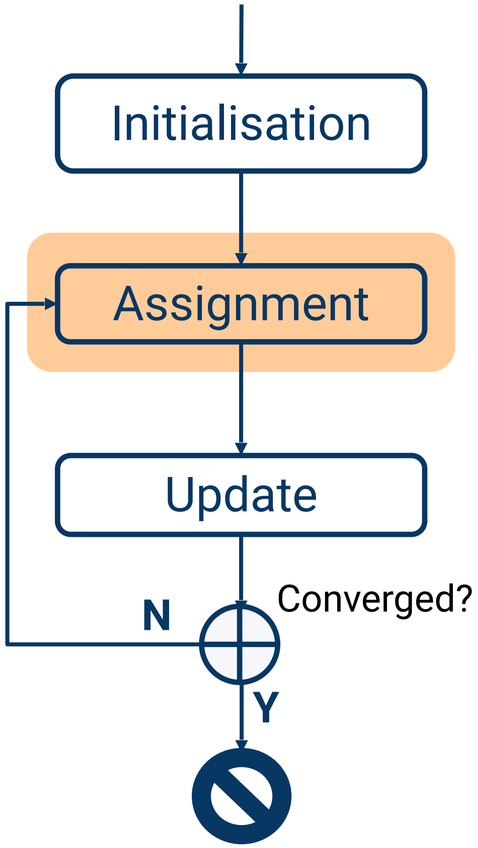


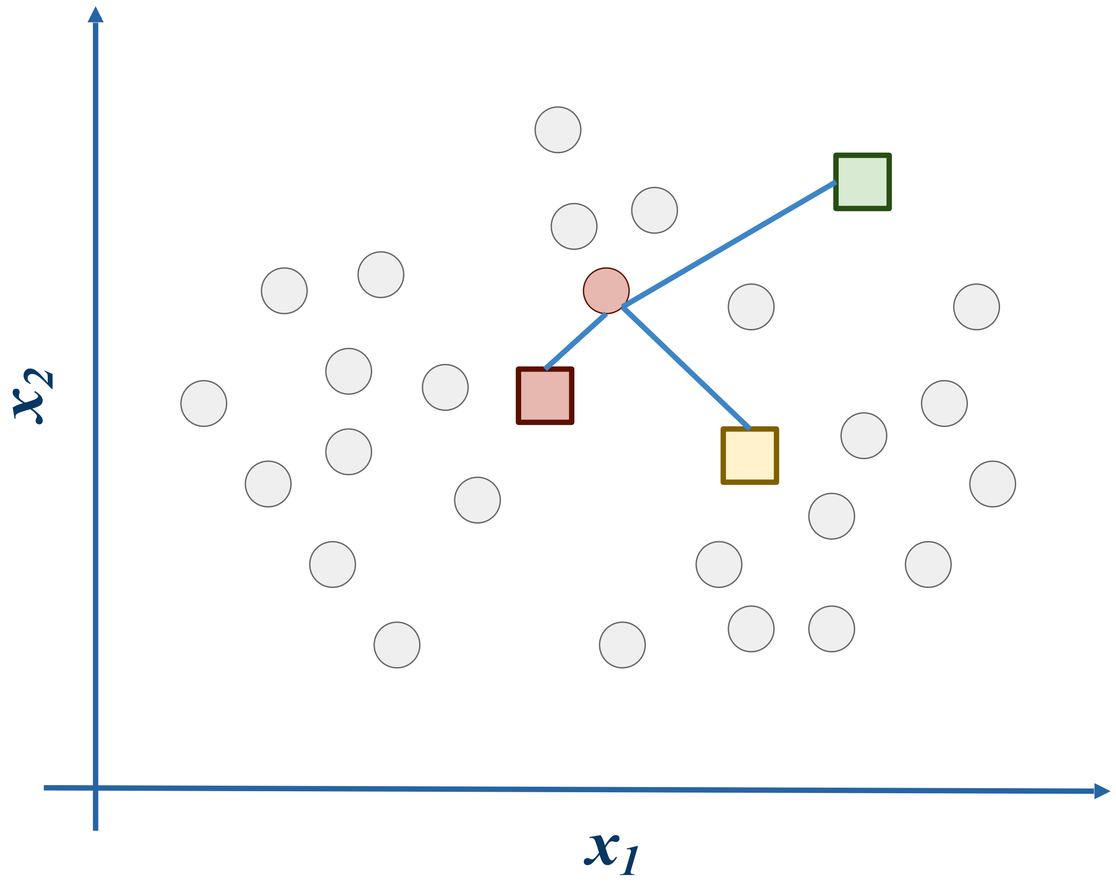
K-Means



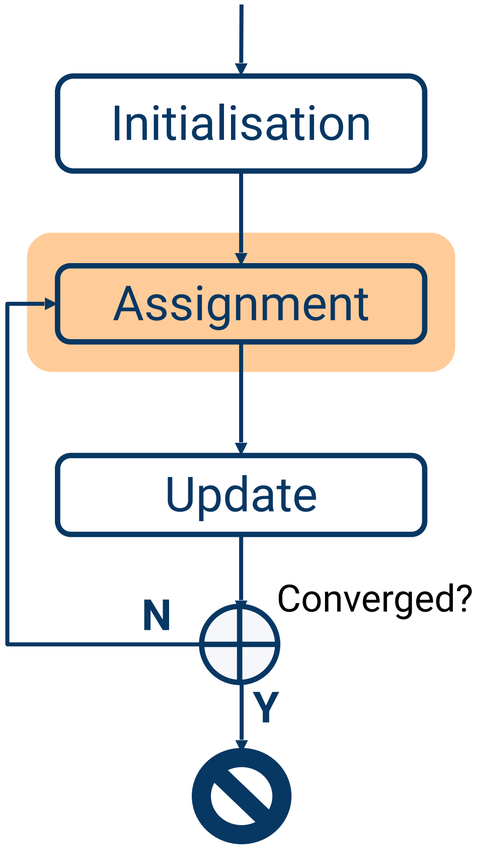


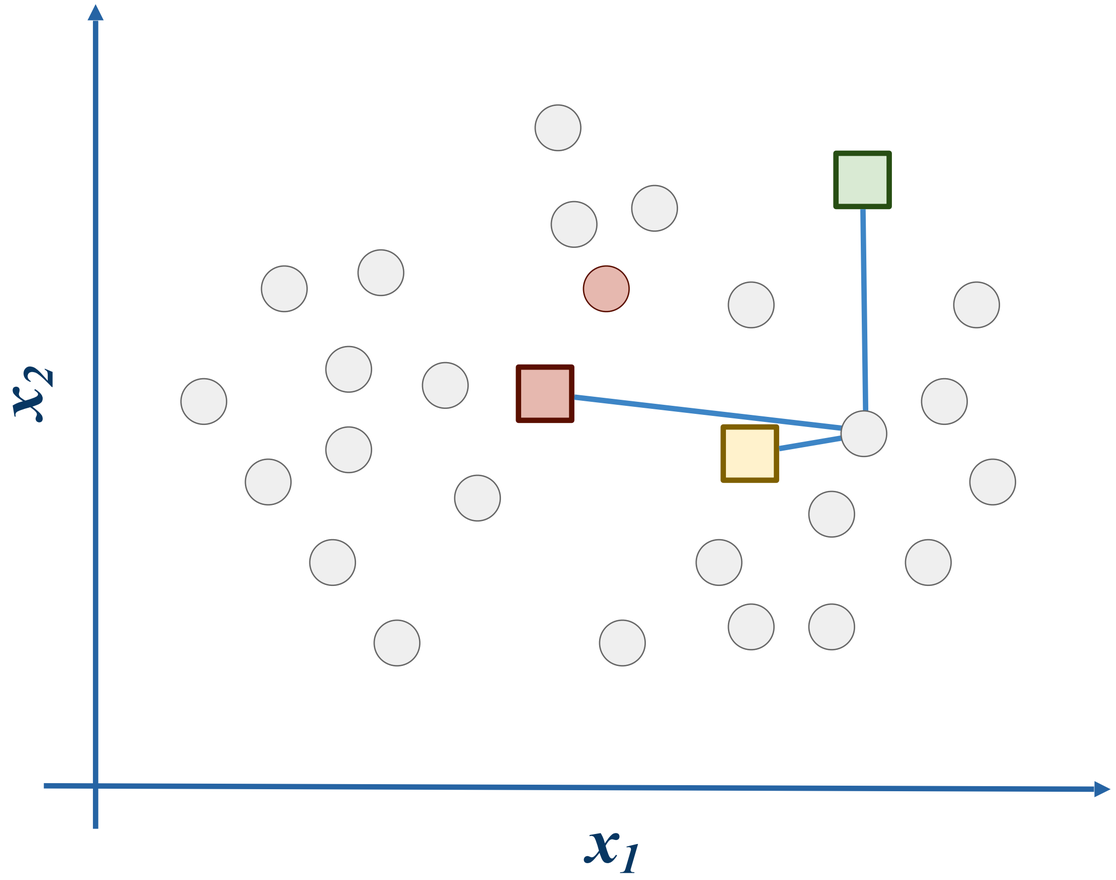
K-Means



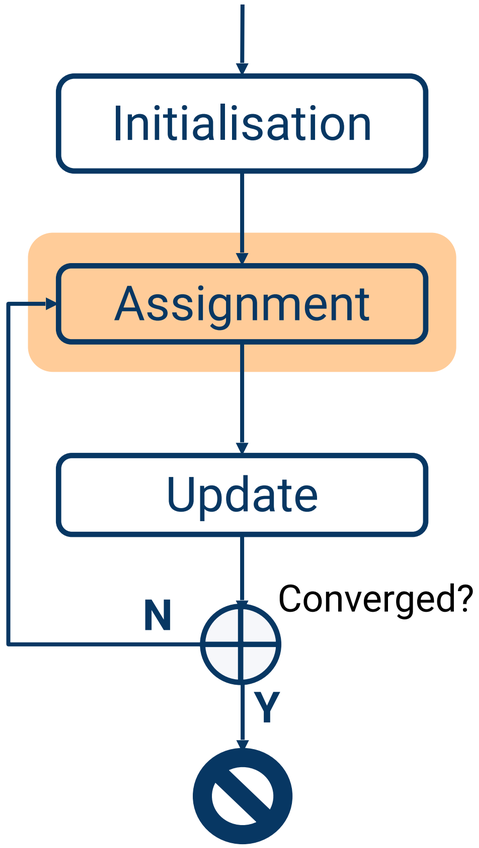


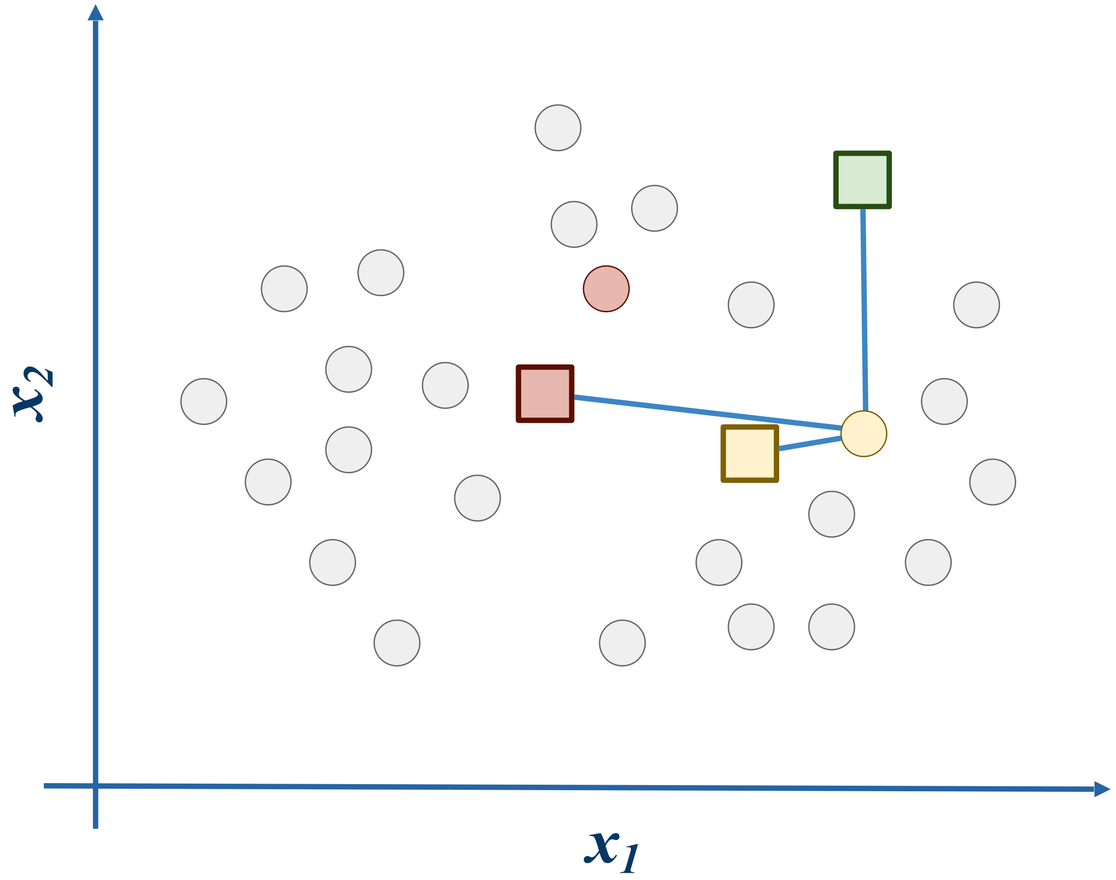
K-Means



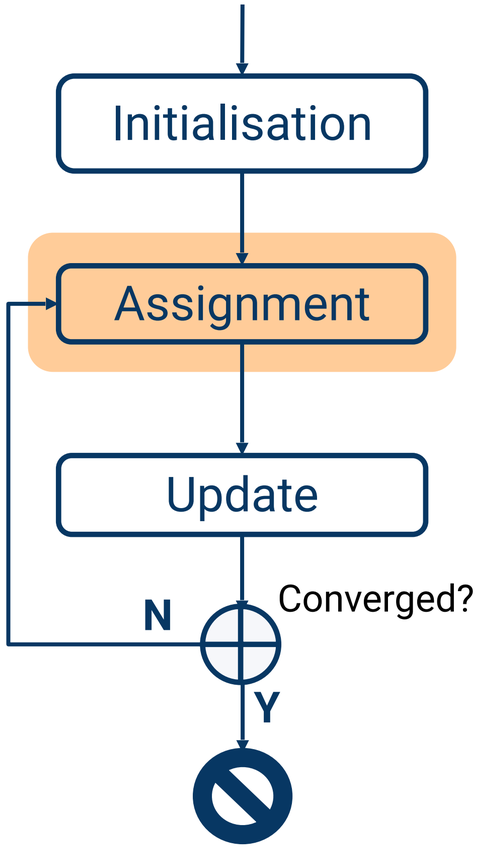


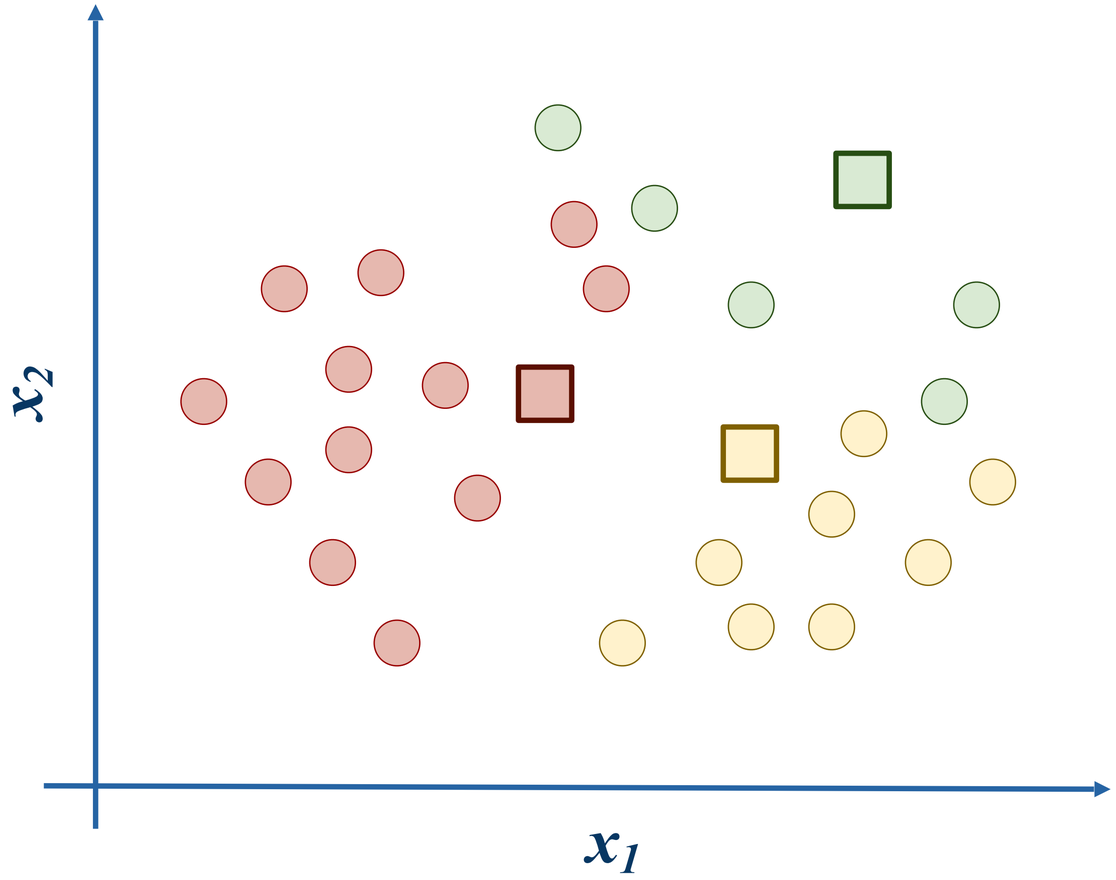
K-Means



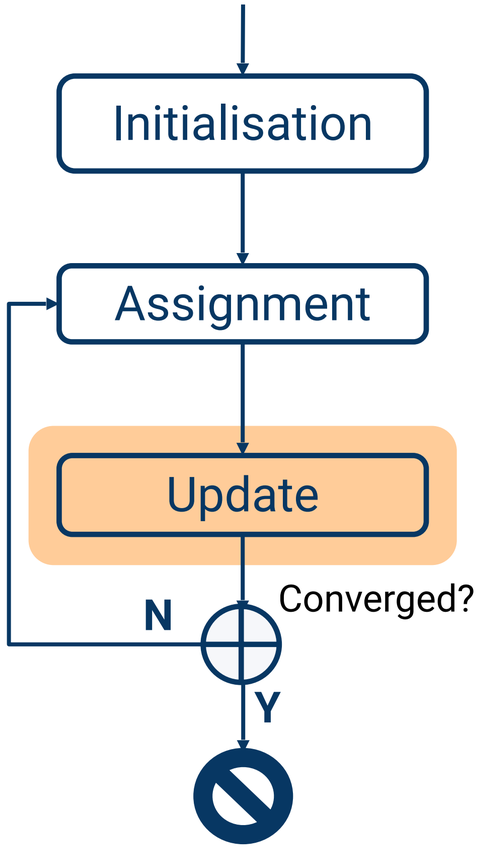


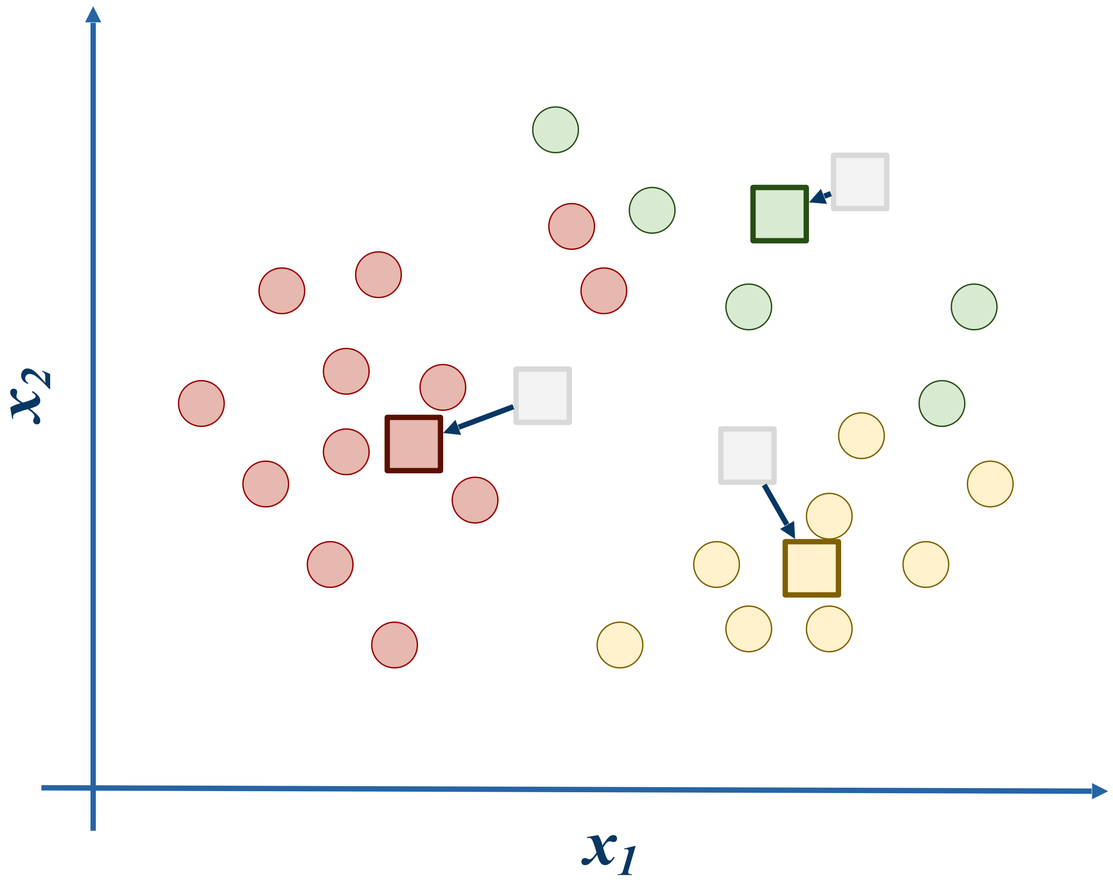
K-Means





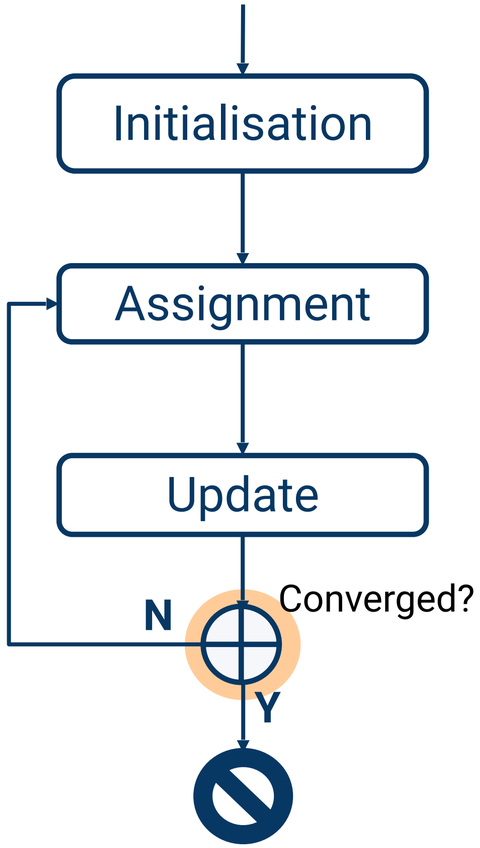
K-Means

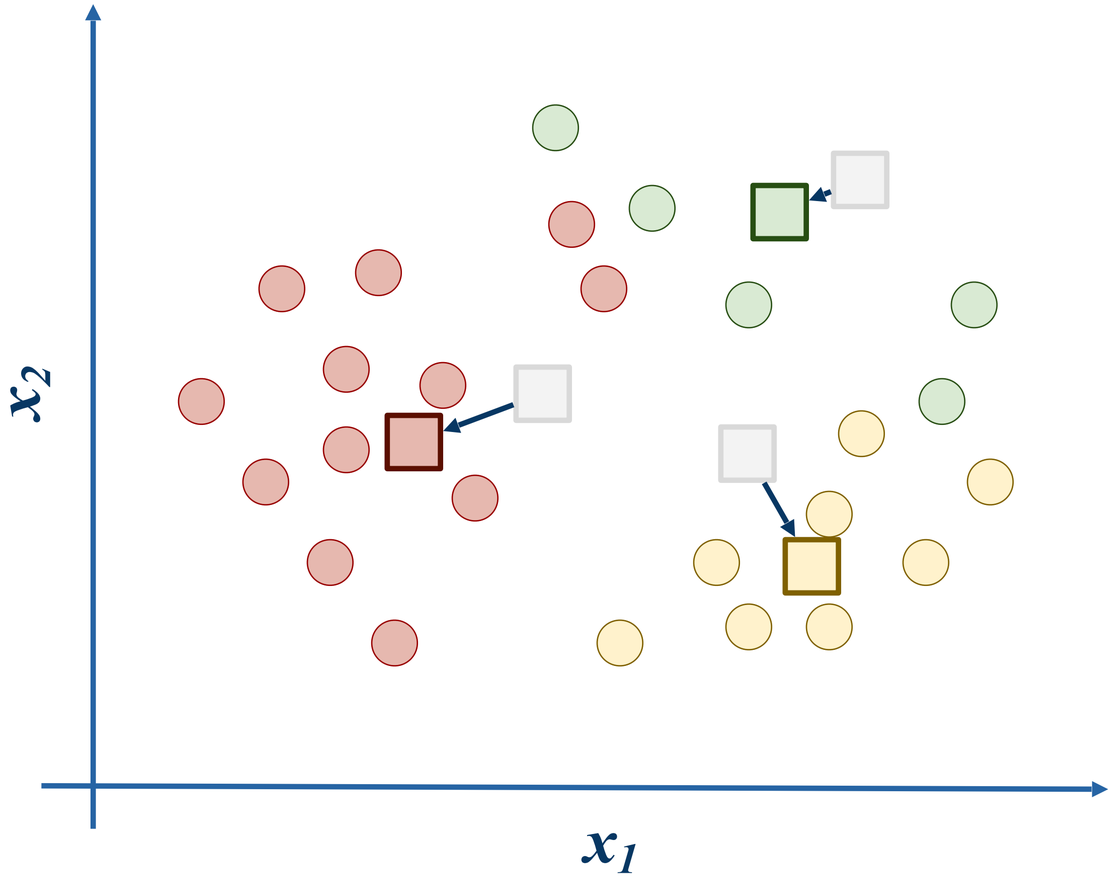




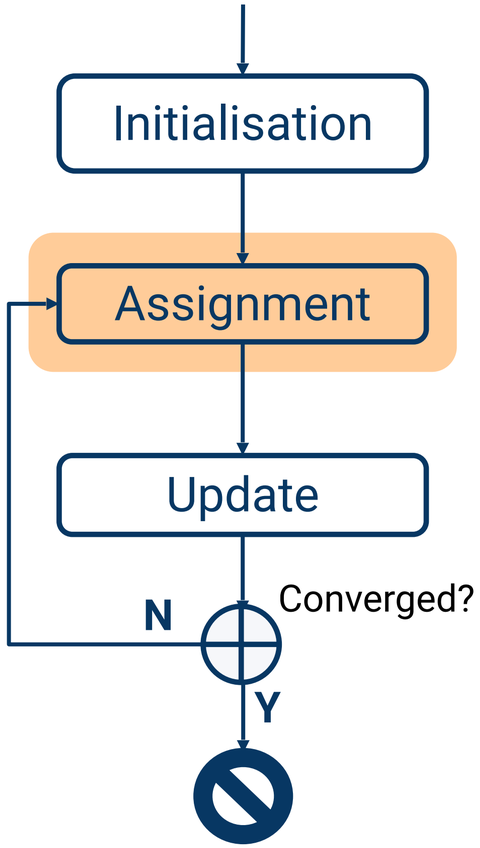
31

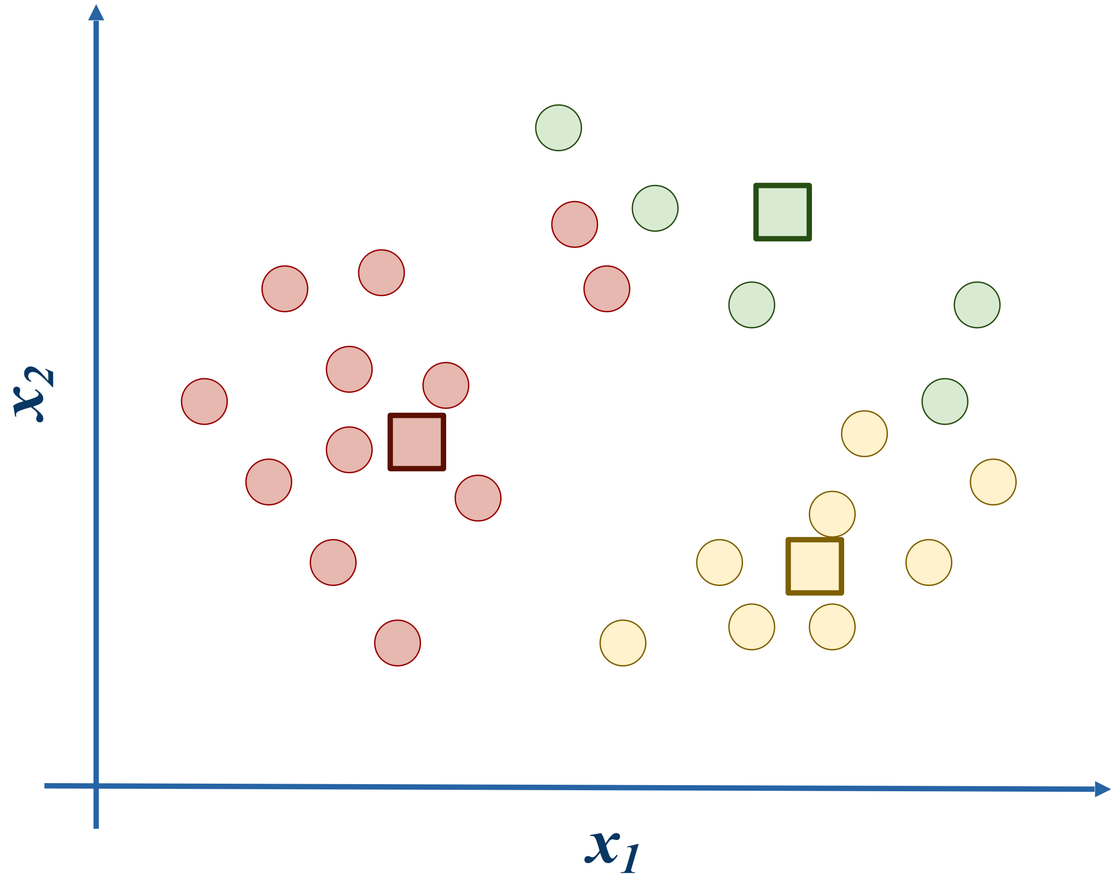
K-Means



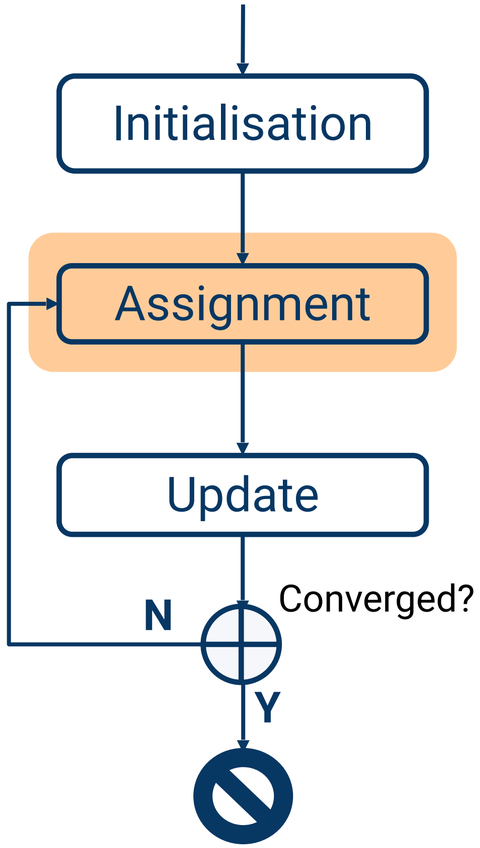


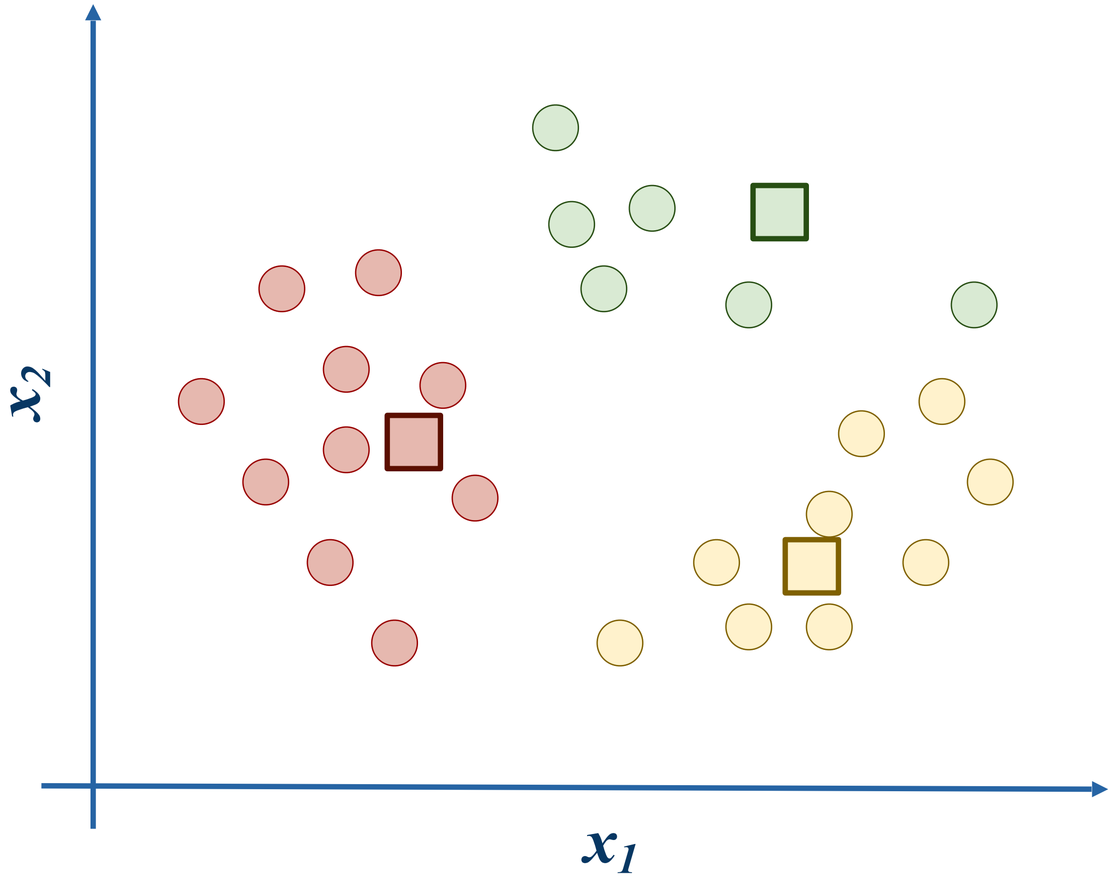
K-Means



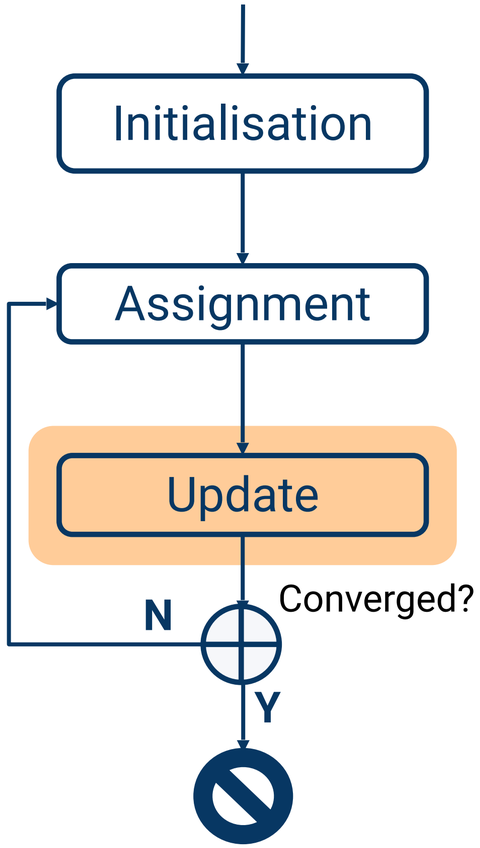


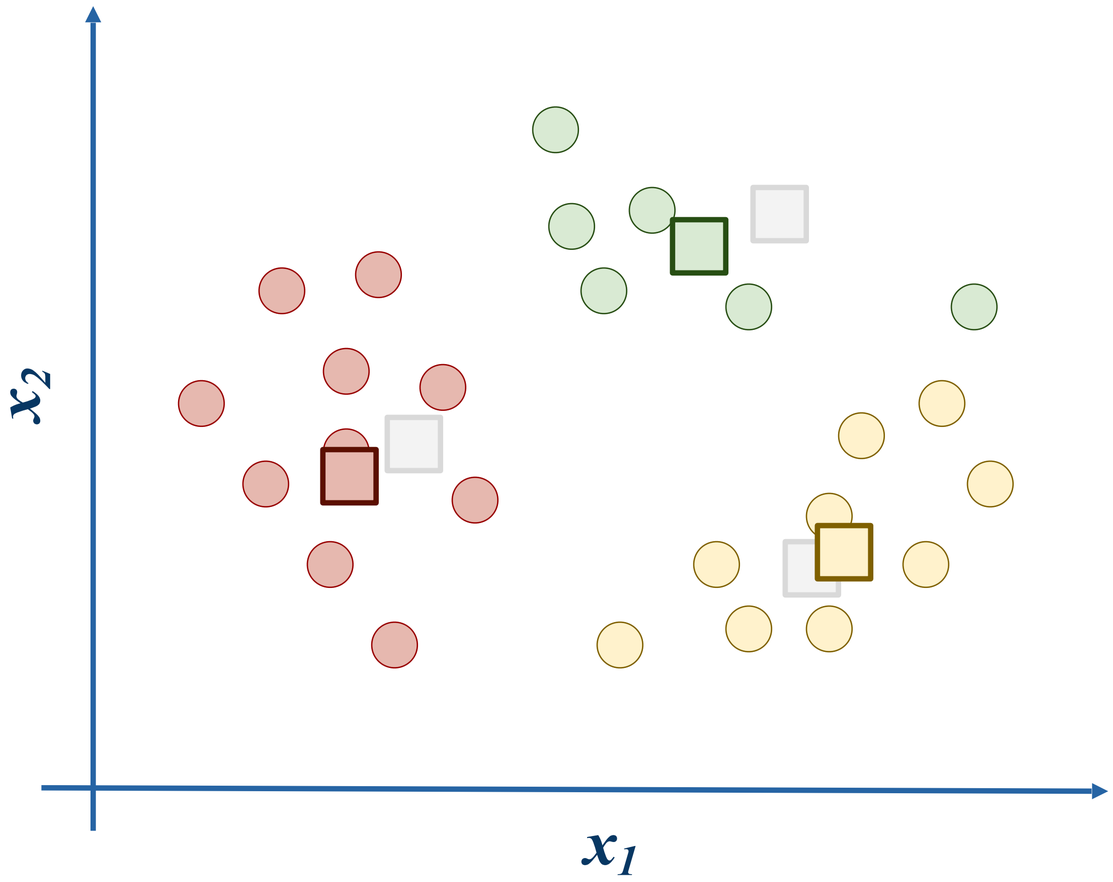
K-Means





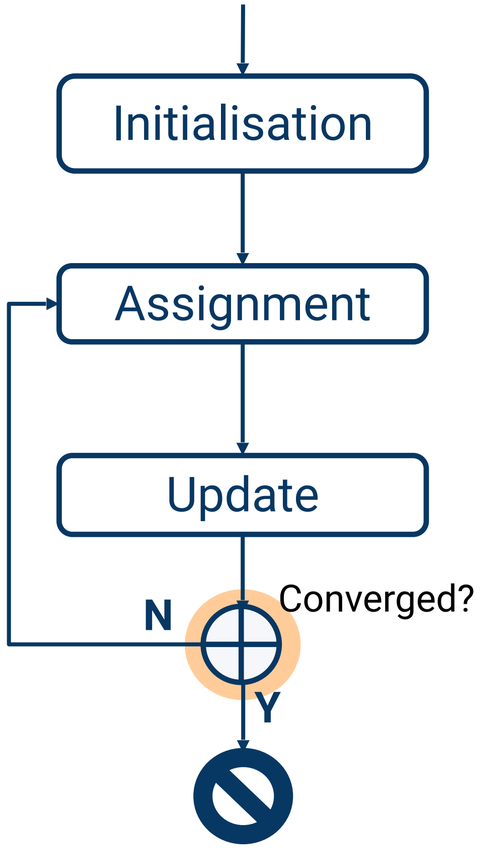
K-Means

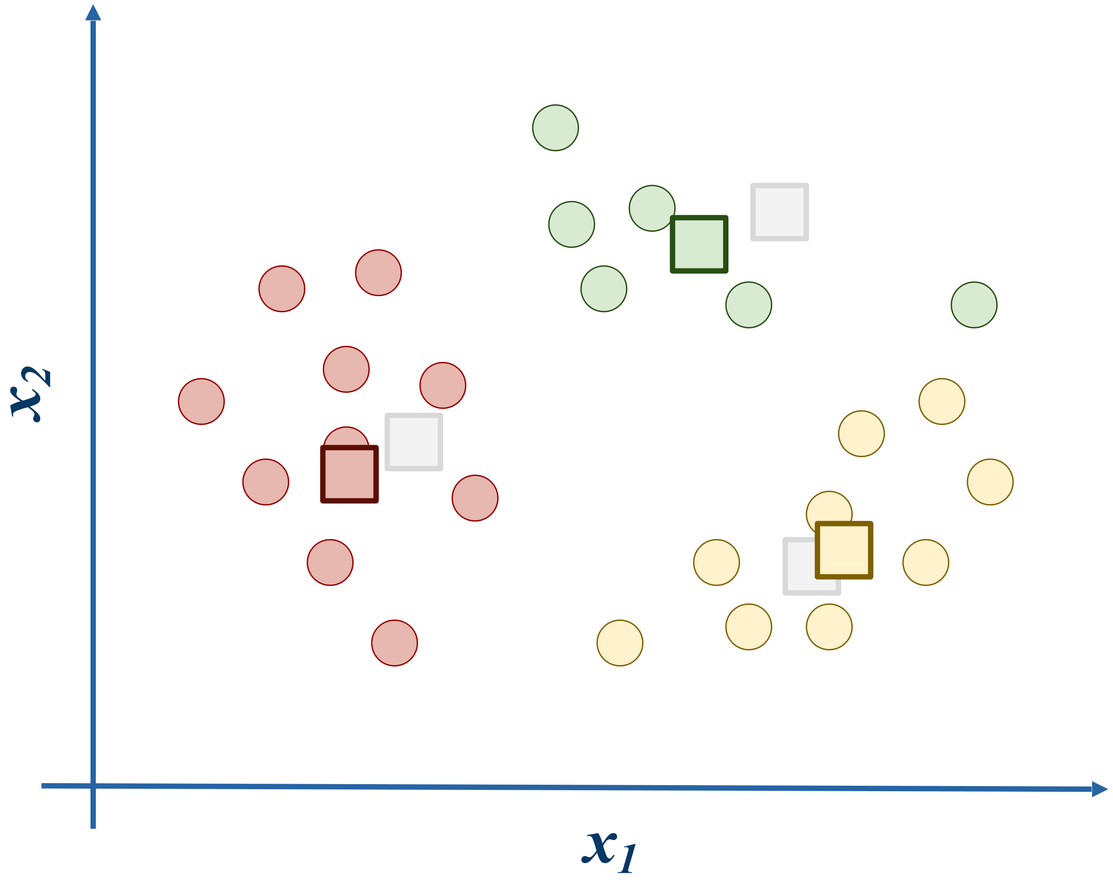




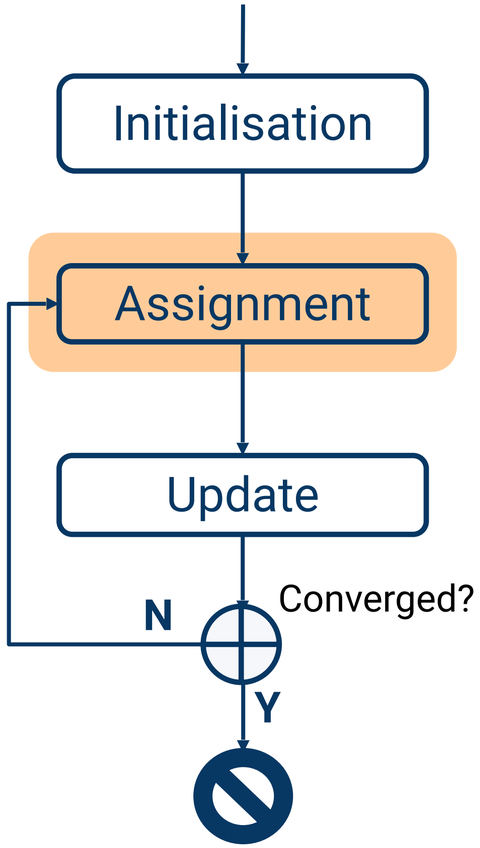
35

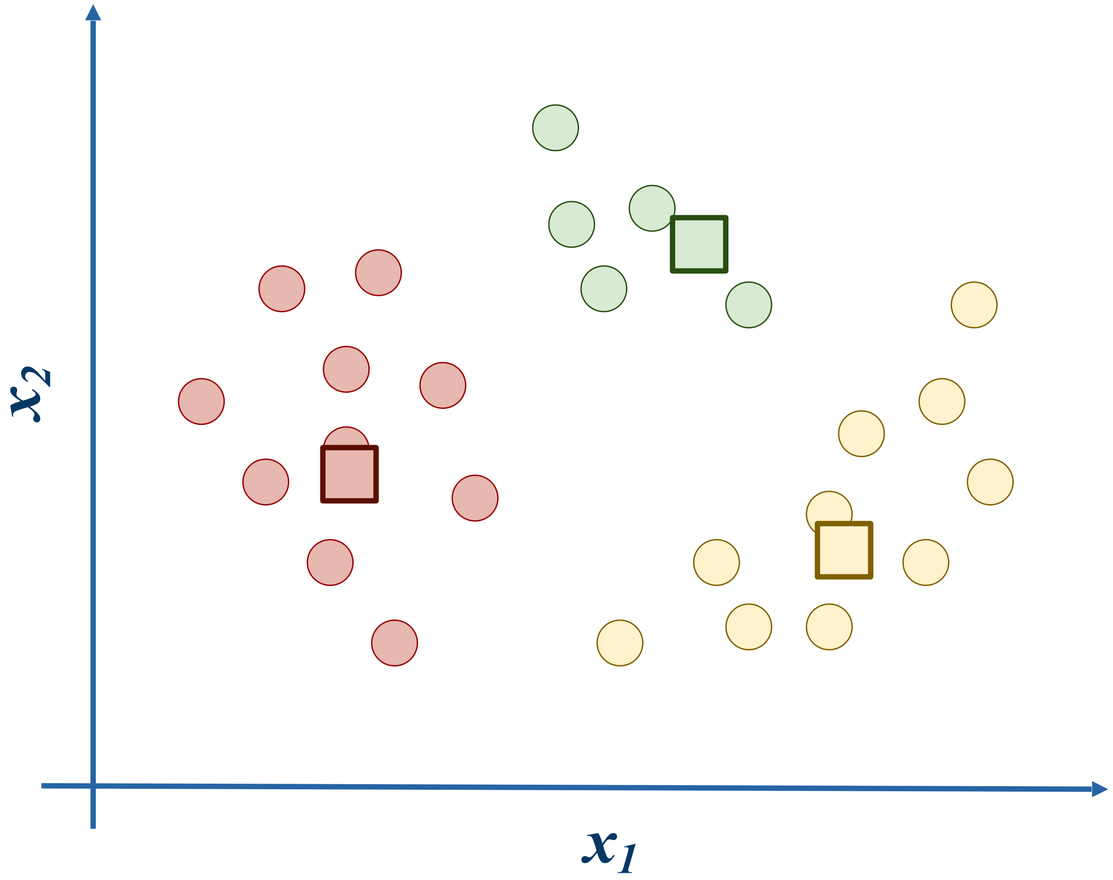
K-Means



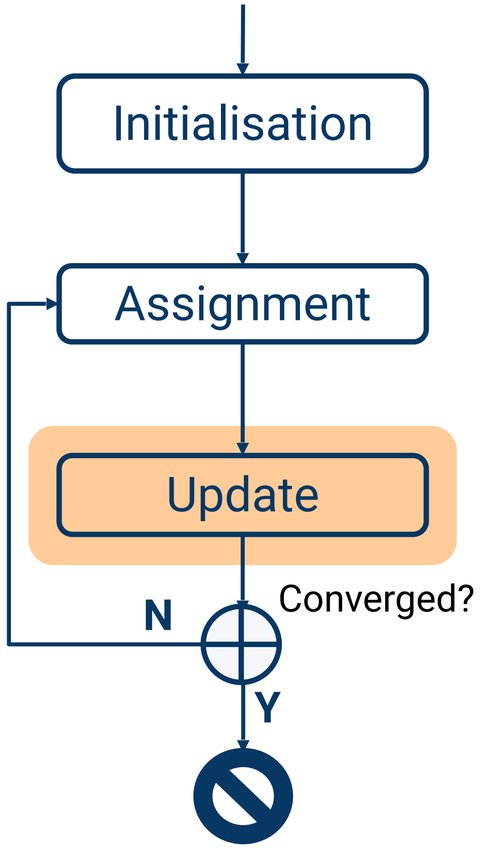


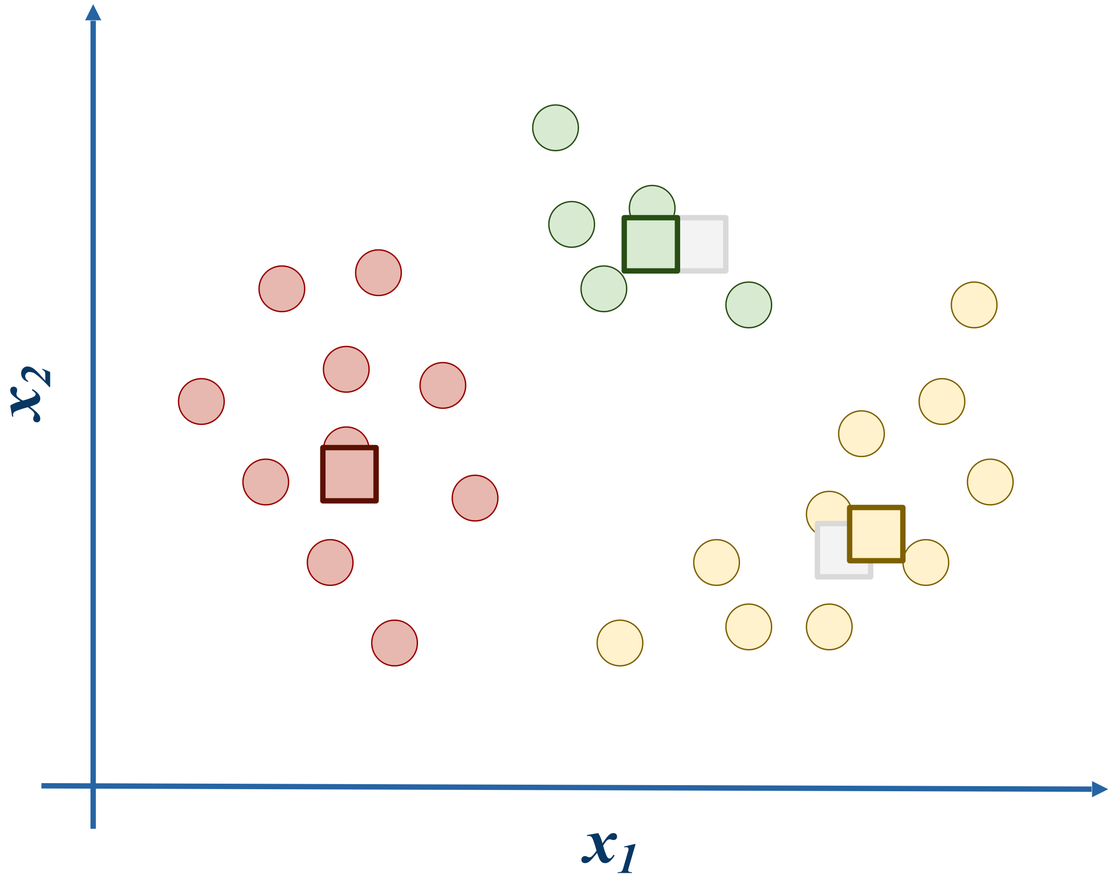
K-Means





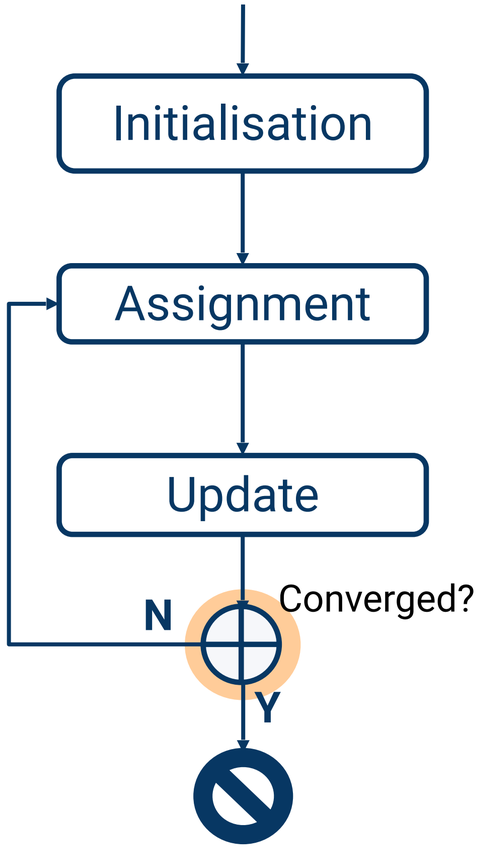
K-Means

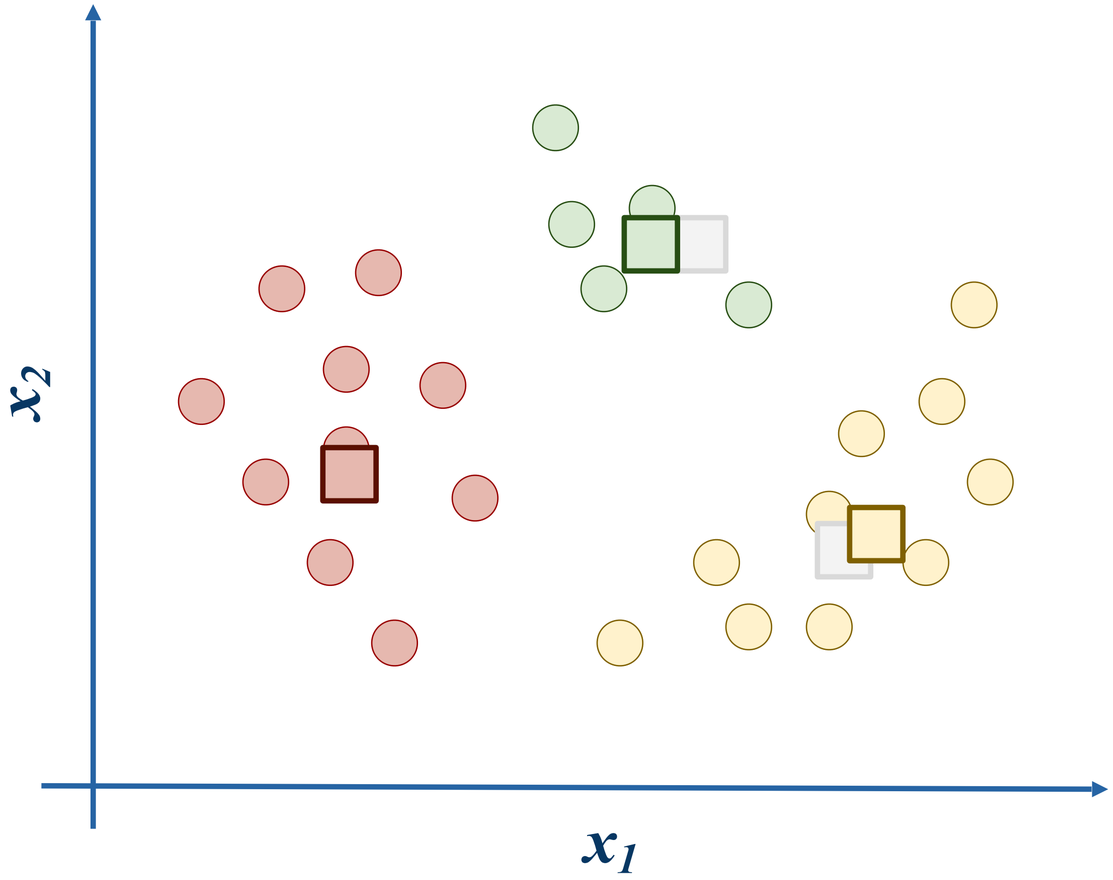




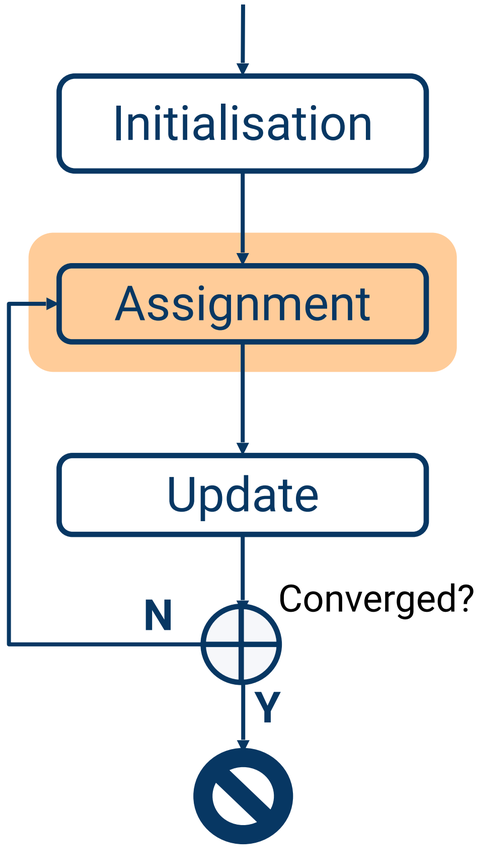
38

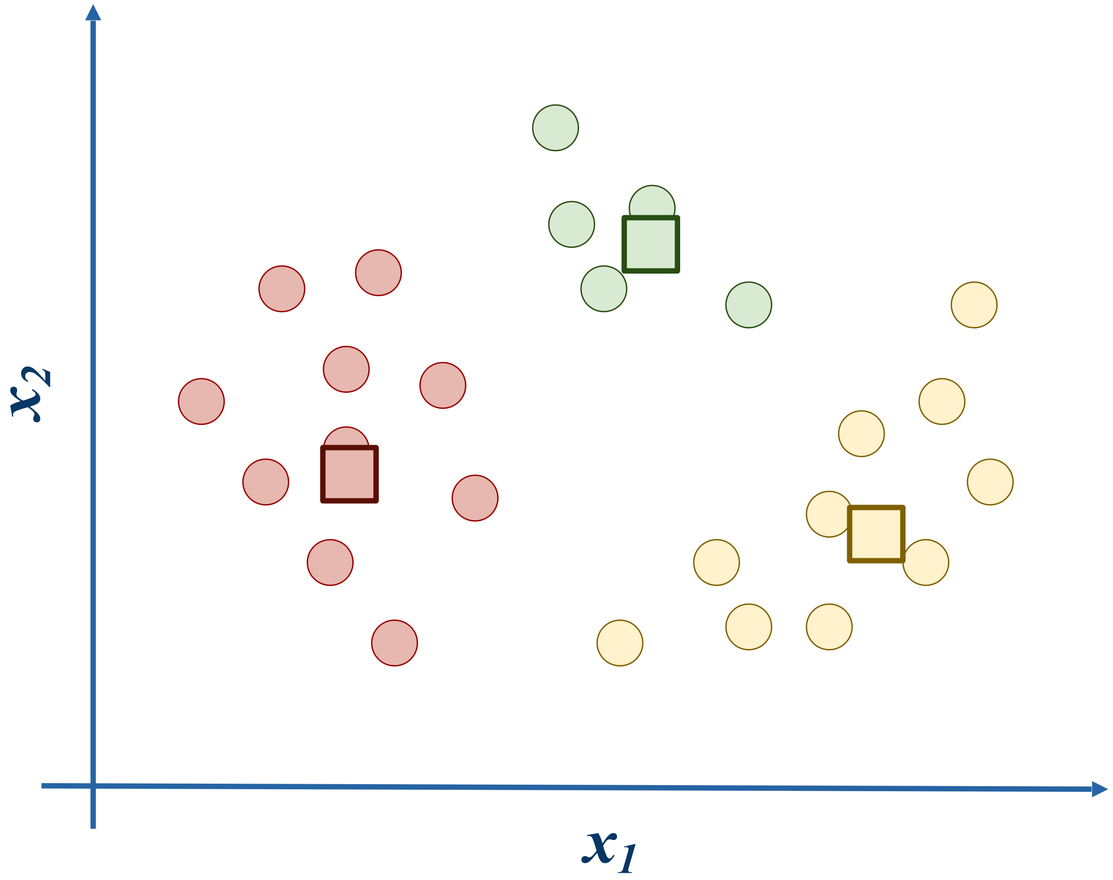
K-Means



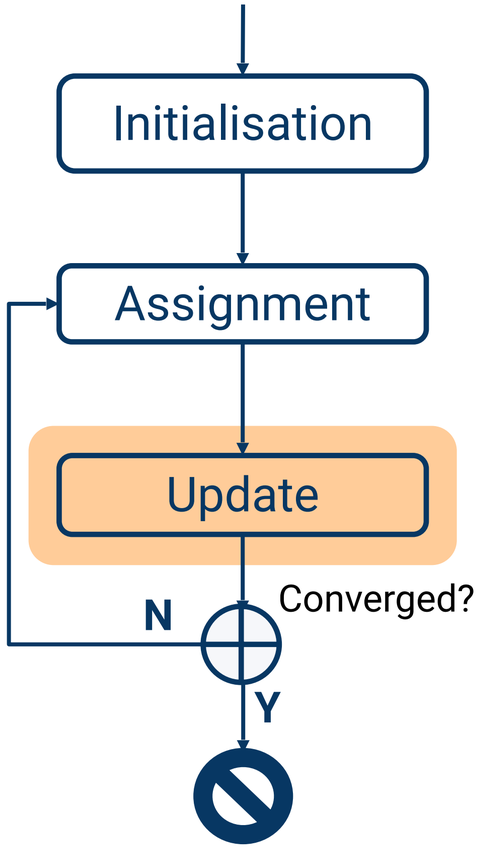


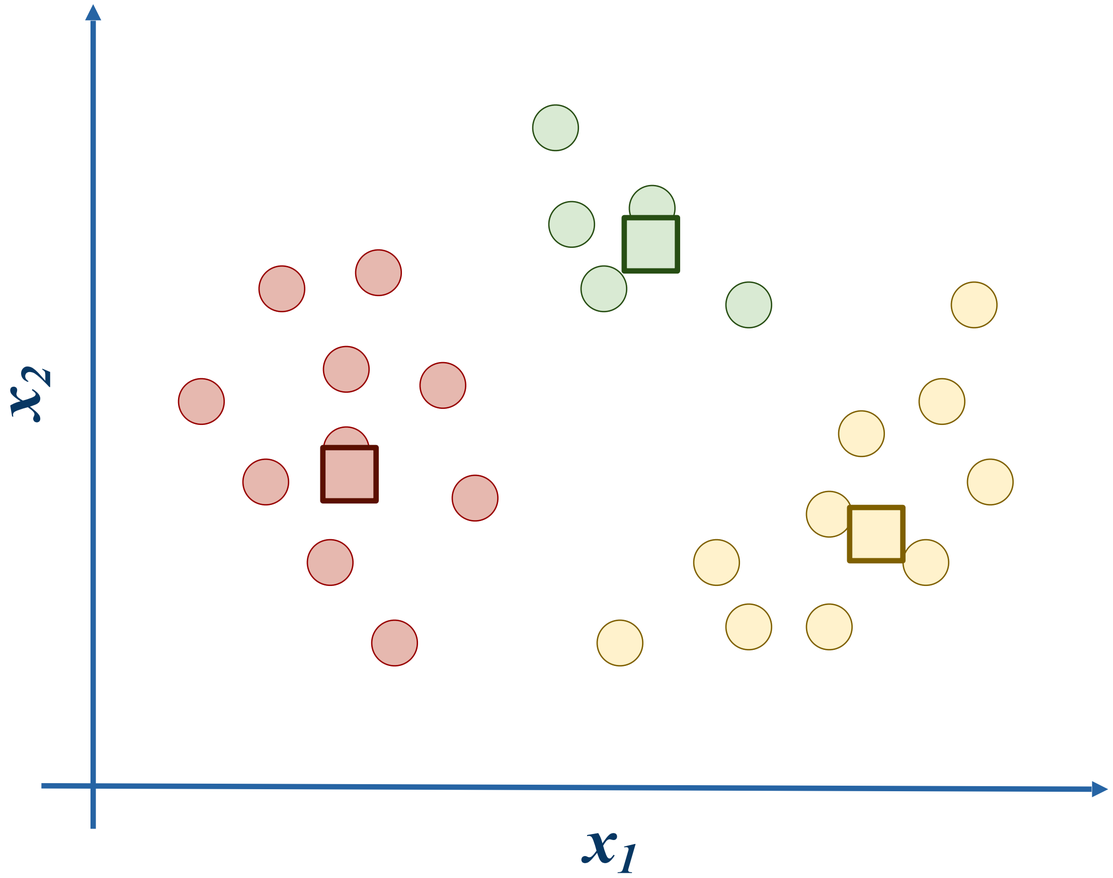
K-Means





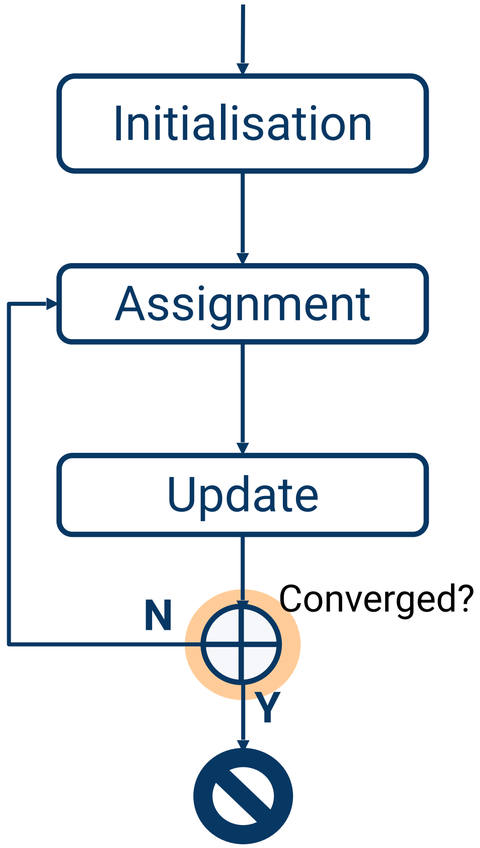
K-Means

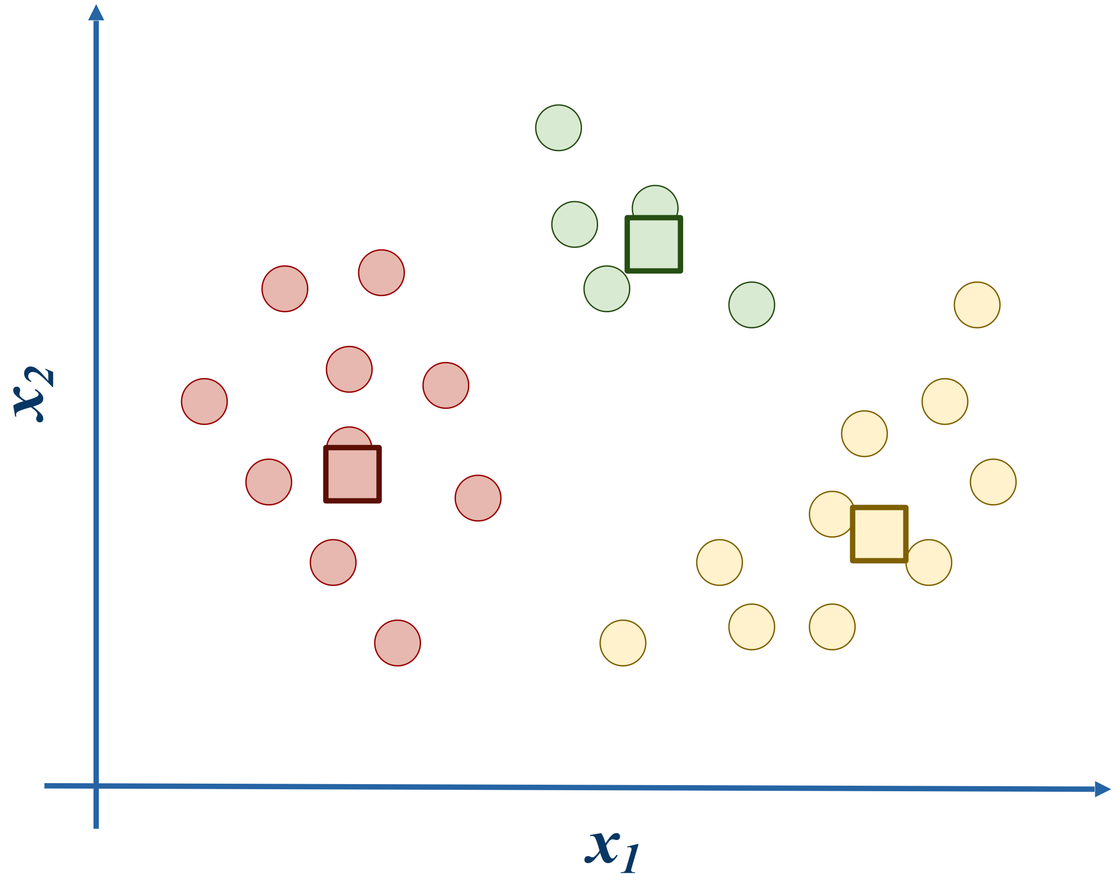




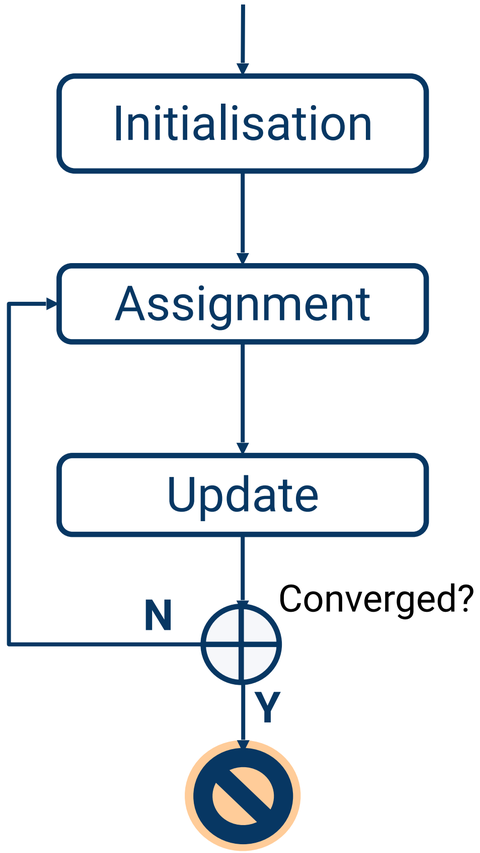
41

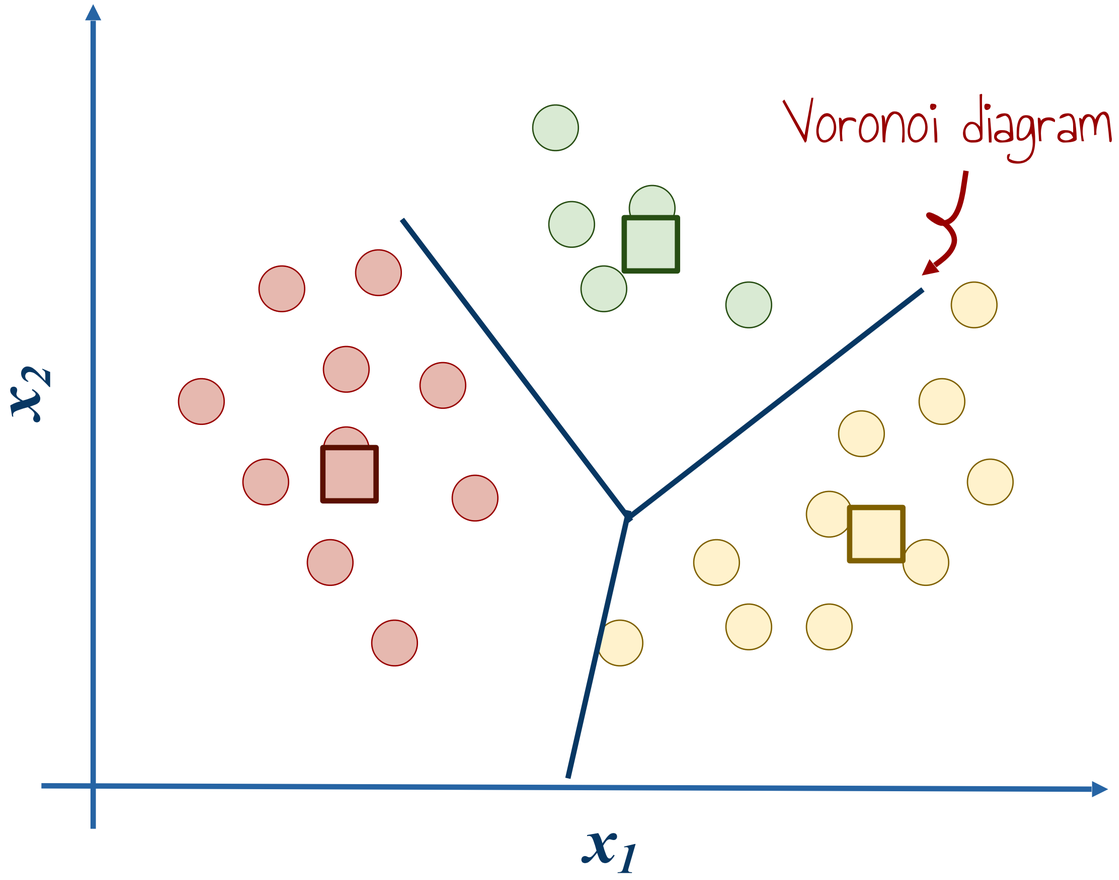
K-Means





K-Means

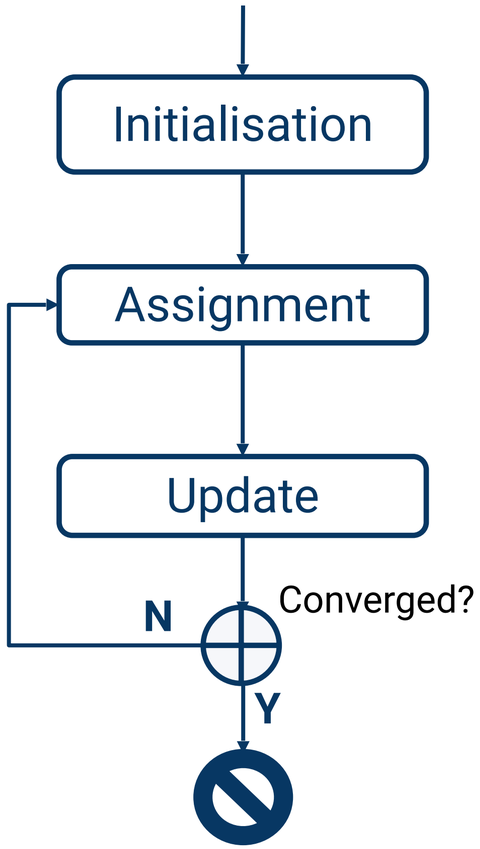

Initialisation
Assignment
Update
Converged?
N
Y


Voronoi diagram
x_2
x_1

K-Means: Summary

Better: select K instances at random.

Avoids empty clusters



Step 1: Initialisation

- Select *K*, generate *K* random cluster centroids

Step 2: Assignment

- Assign each training example to the nearest centroid

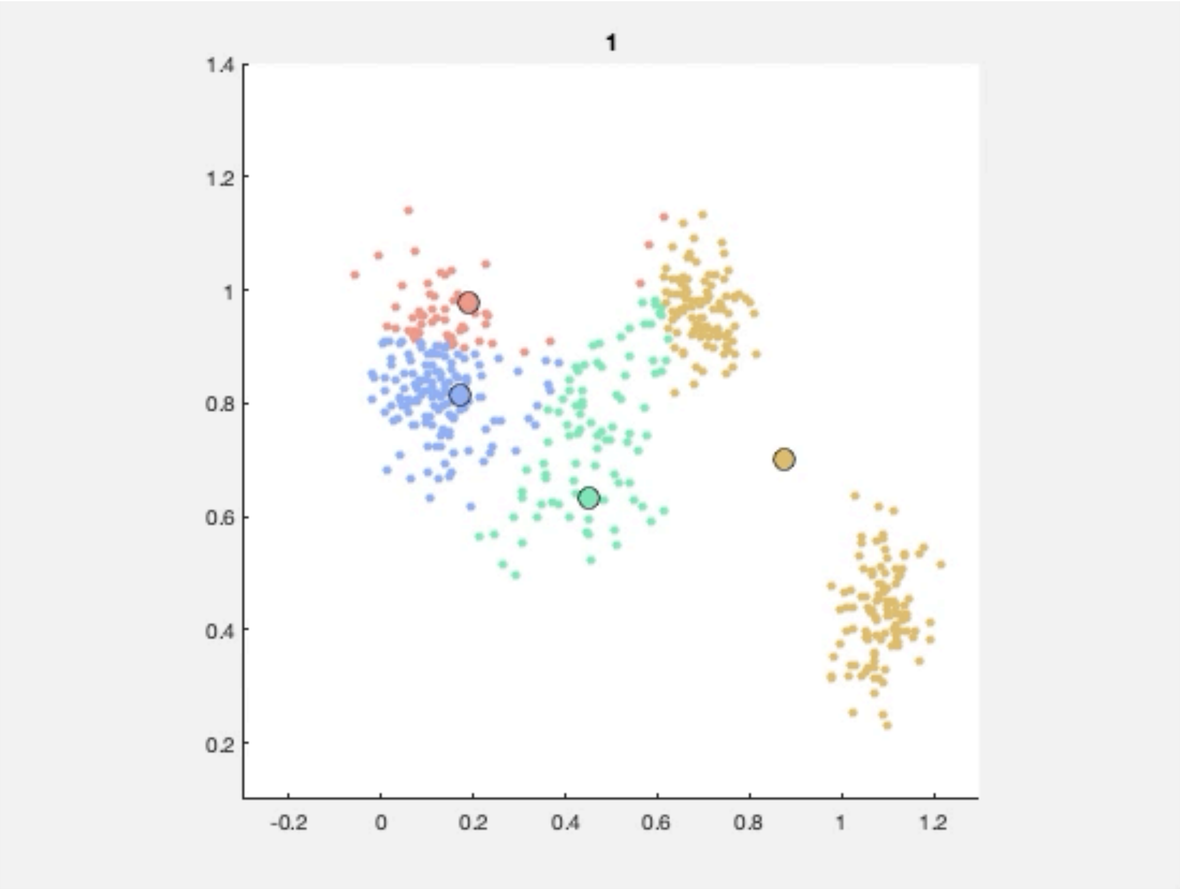
Step 3: Update

- Update position of each centroid by computing the mean position of all examples assigned to it

Step 4: Convergence check

- Stop if position of centroids did not change. Otherwise go to Step 2.

K-means in video



K-Means: More formally

Step 1: Initialisation

Randomly select *K* training examples as cluster centroids

Step 2: Assignment

For each $i \in \{1, \ldots, N\}$

$$c^{(i)} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmin}} ||x^{(i)} - \mu_k||^2$$

To which cluster is training example $x^{(i)}$ assigned?

$c^{(i)} \in \{1, \ldots, K\}$

Distance between training example (i) and cluster mean

Step 3: Update

For each $k \in \{1, \ldots, K\}$

$$\mu_k = \frac{\sum_{i=1}^{N} 1(c^{(i)} = k) \cdot x^{(i)}}{\sum_{i=1}^{N} 1(c^{(i)} = k)}$$

Mean of cluster k

Indicator function: 1(a)

If a is True, return 1

Else return 0

Step 4: Convergence check

If $\forall_k \left| \mu_k^t - \mu_k^{(t-1)} \right| < \epsilon$, stop. Else go to Step 2.

Difference between updated mean and previous mean for each cluster is "insignificant enough"

K-Means: Viewed as a model

K-Means optimisation objective:

Minimize

All training examples

Mean of the cluster to which $x^{(i)}$ is assigned

$$L(c^{(1)}, \ldots, c^{(N)}, \mu_1, \ldots, \mu_K) = \frac{1}{N} \sum_{i=1}^{N} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

Cluster assignments

Cluster means

Minimize the average distance between each training sample to the mean of its assigned cluster

K-Means: Viewed as a model

Step 1: Initialisation

Randomly select *K* training examples as cluster centroids

Step 2: Assignment

For each $i \in \{1, \ldots, N\}$ Minimize $L(\overbrace{c^{(1)}, \ldots, c^{(N)}}^{\text{Minimize L wrt c}}, \overbrace{\mu_1, \ldots, \mu_K}^{\text{Fix } \mu})$

$$c^{(i)} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmin}} ||x^{(i)} - \mu_k||^2$$

Step 3: Update

For each $k \in \{1, \ldots, K\}$ Minimize $L(\overbrace{c^{(1)}, \ldots, c^{(N)}}^{\text{Fix c}}, \overbrace{\mu_1, \ldots, \mu_K}^{\text{Minimize L wrt } \mu})$

$$\mu_k = \frac{\sum_{i=1}^{N} 1(c^{(i)} = k) \cdot x^{(i)}}{\sum_{i=1}^{N} 1(c^{(i)} = k)}$$

Step 4: Convergence check

If $\forall_k \left| \mu_k^t - \mu_k^{(t-1)} \right| < \epsilon$, stop. Else go to Step 2.

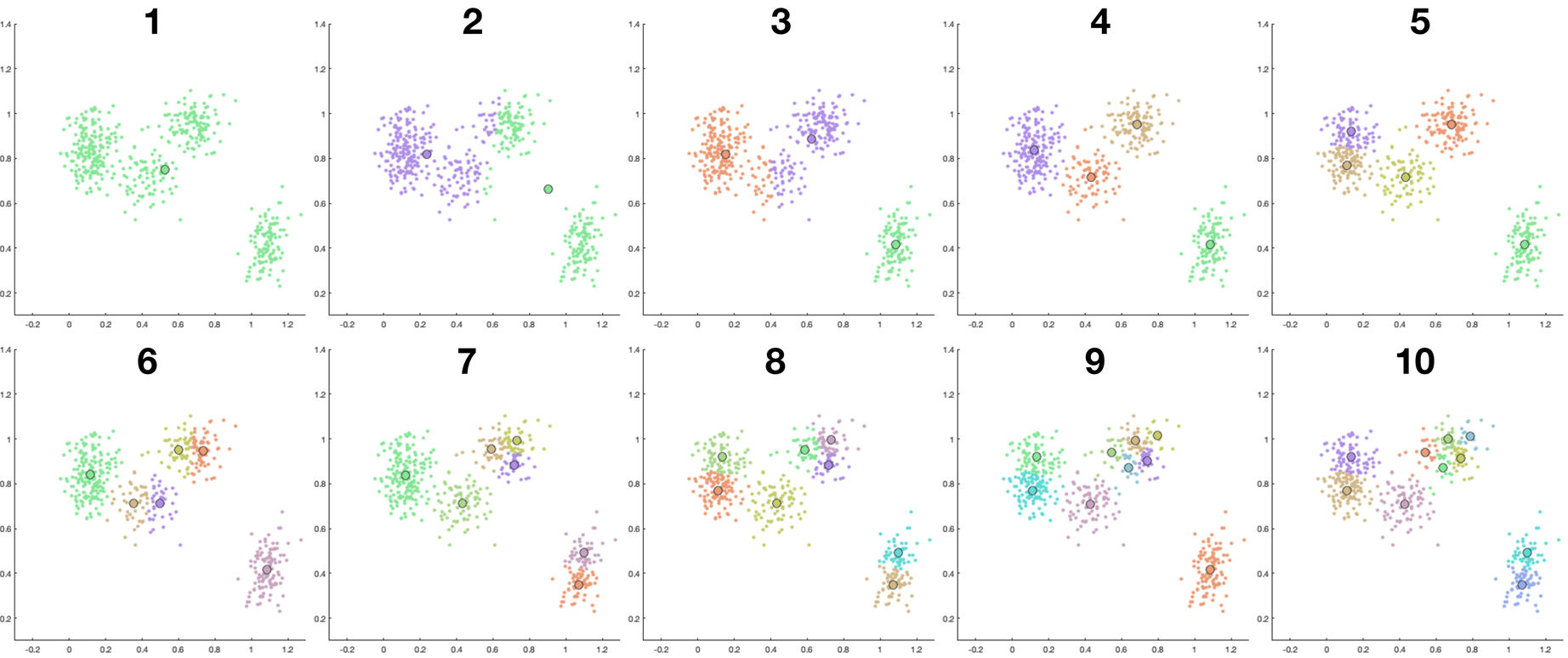
To be continued...
(K-means: Hyper-parameter tuning,
strengths and weaknesses)

K-means:
Hyper-parameter tuning, strengths and weaknesses

*How to select **K**?*

Different values of K will lead to very different results. So picking the best value is important.

Several approaches exist to select K.

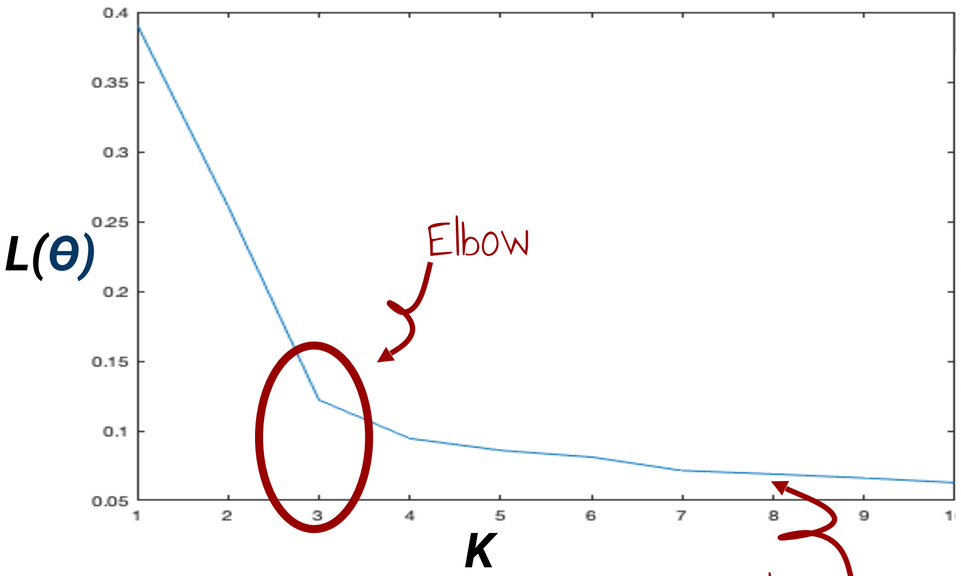


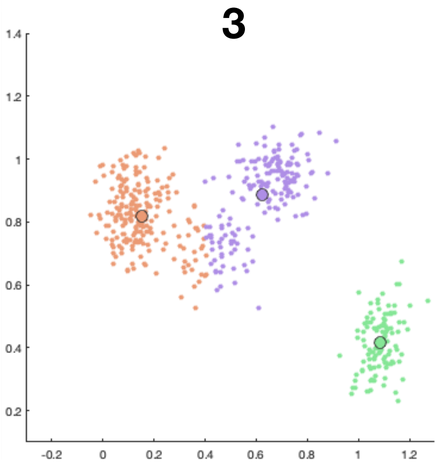
How to select **K**: The elbow method

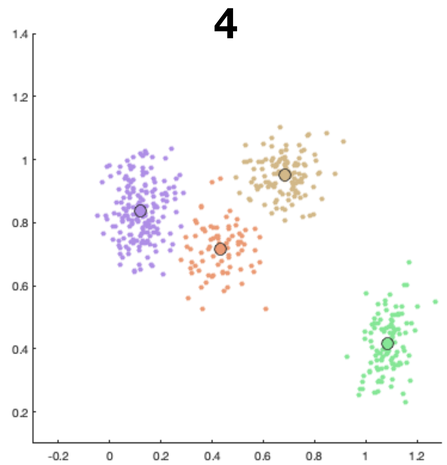
The main approach is the "elbow method":

- Run *K*-means multiple times with different *K*'s (e.g., from 1 to 10)
- Keep track of cost $L(\Theta)$ for each K value.
- Select *K* where the rate of decrease sharply shifts

Average distance between the samples in a cluster and the cluster mean





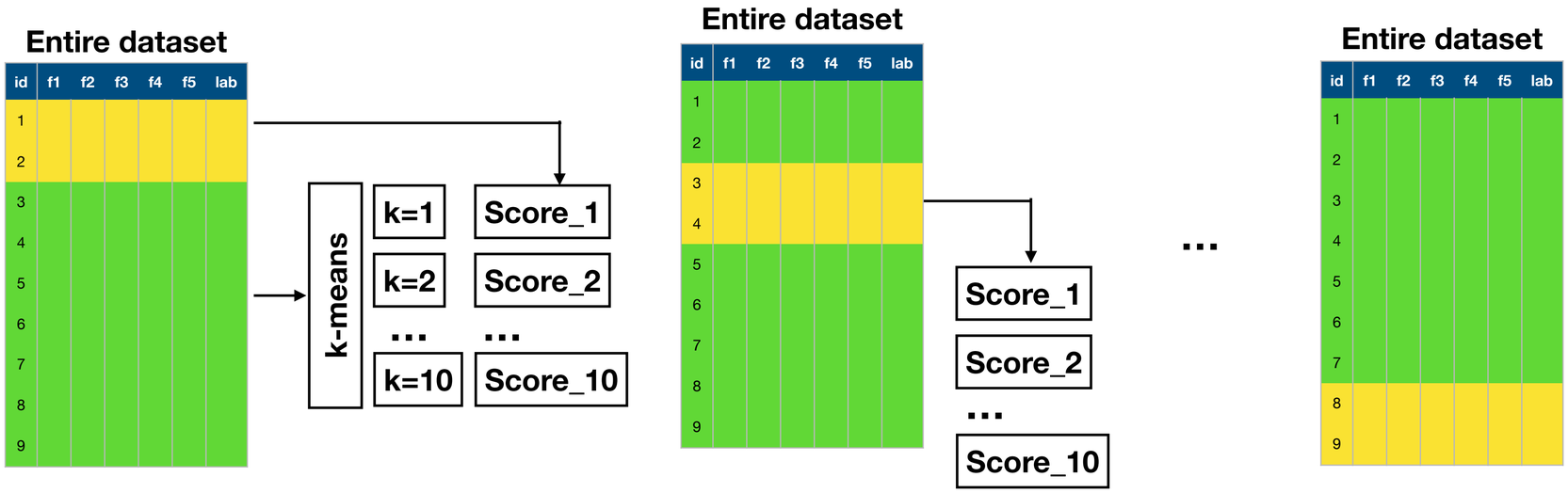


The score alone cannot be used, as it will always converges to 0 when k is equal to the number of datapoints.

*How to select **K**: The cross-validation method*

Another approach is based on cross-validation.

The dataset is split in N folds, and N-1 folds are used to compute the centroids positions with k-means. Then, we compute the average score (same as before) on the validation datasets. We do this for various values of k and we pick the best configuration.



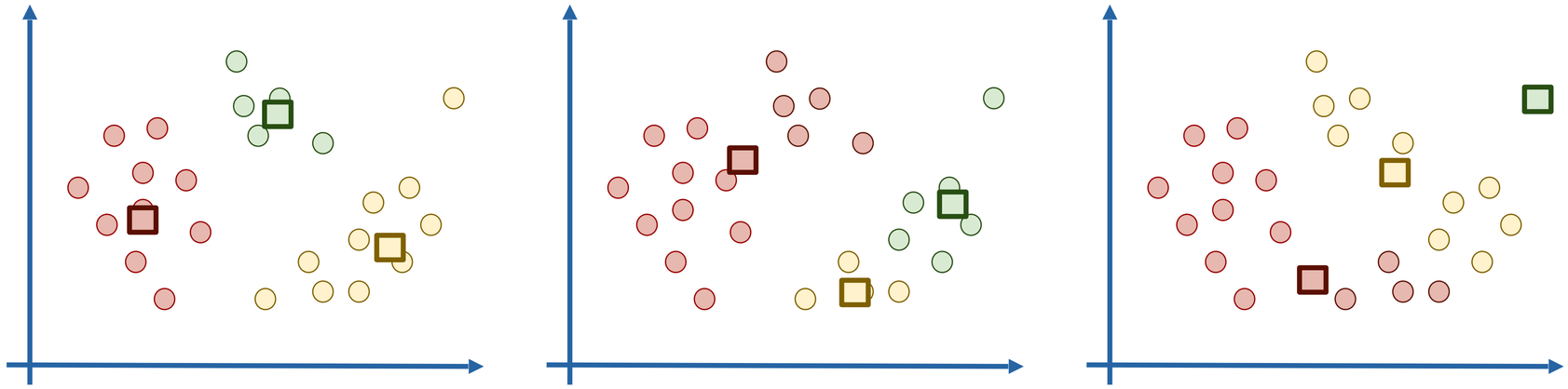
Select K such that further increase in number of clusters leads to only a small improvement in the average score_k.

K-means: Strengths

- **Simple**
  - Easy to understand and implement
- **Popular**
  - Used very often for clustering
- **Efficient**
  - Linear complexity
  - $O(TKN)$ for T iterations, K clusters, N examples.
    Since $K << N$ and $T << N$, k-means is considered a linear algorithm.
  -

K-means: Weaknesses

- **We have to define k**
- *K*-means finds a local optimum
- … and is sensitive to the initial centroid positions

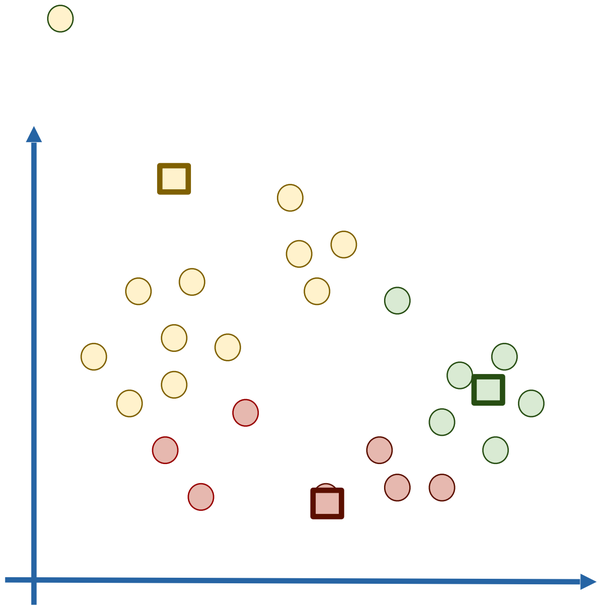


- Solutions:
  - Run multiple times, choose model with lowest cost $L(\Theta)$
  - Use better initialisation: e.g. *K*-means++

K-means++ : Arthur, D.; Vassilvitskii, S. (2007). "k-means++: the advantages of careful seeding" (PDF). Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035.

K-means: Weaknesses

- K-means is only applicable if a distance function exists.
  - A variant exist for categorical data: k-mode, the centroids represent the most frequent values.
- *K*-means is sensitive to outliers



- Solution:
  - Use an algorithm that is less sensitive to outliers, e.g. *K*-medoid

K-means: Weaknesses

k-means is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).

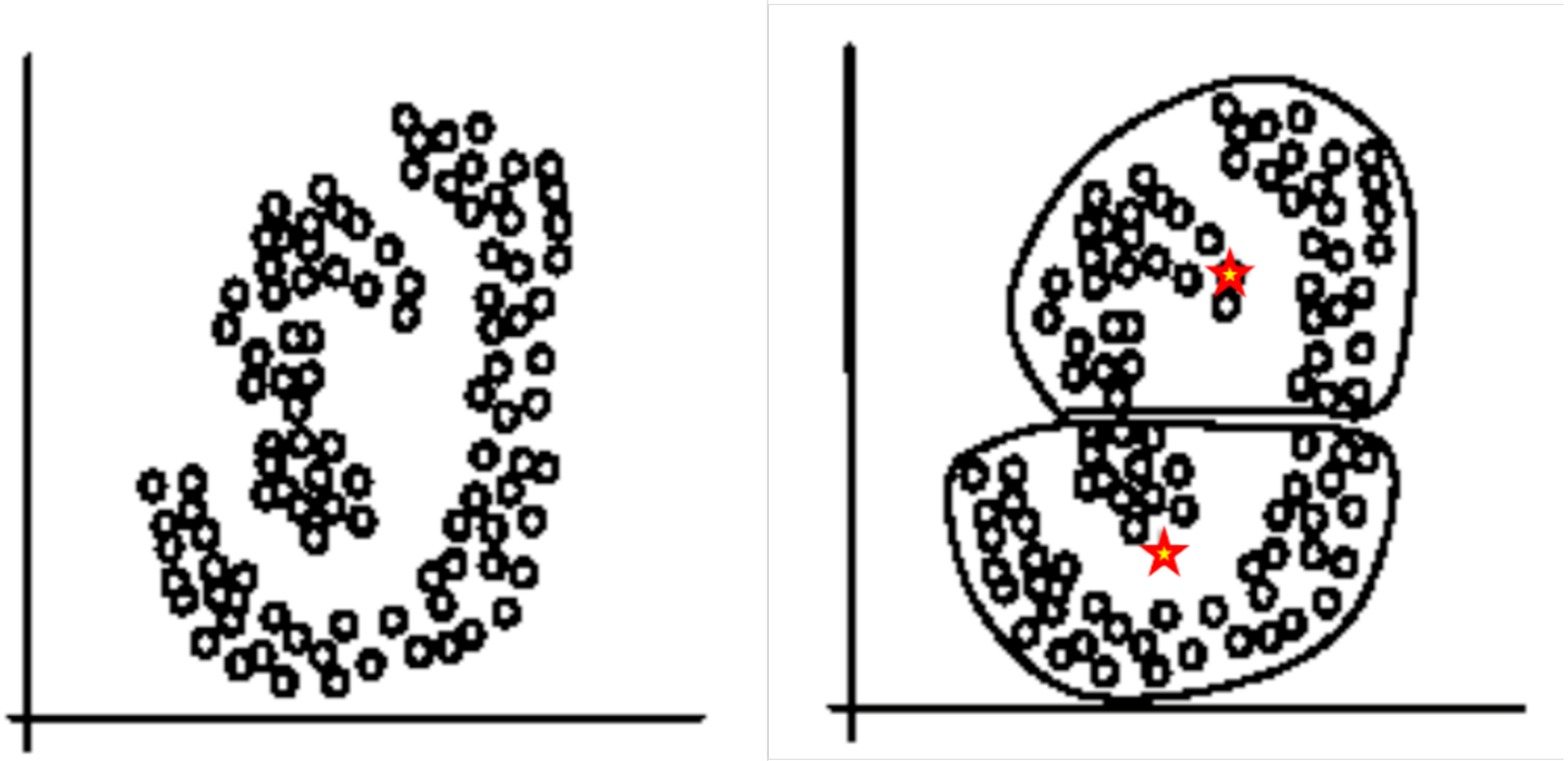
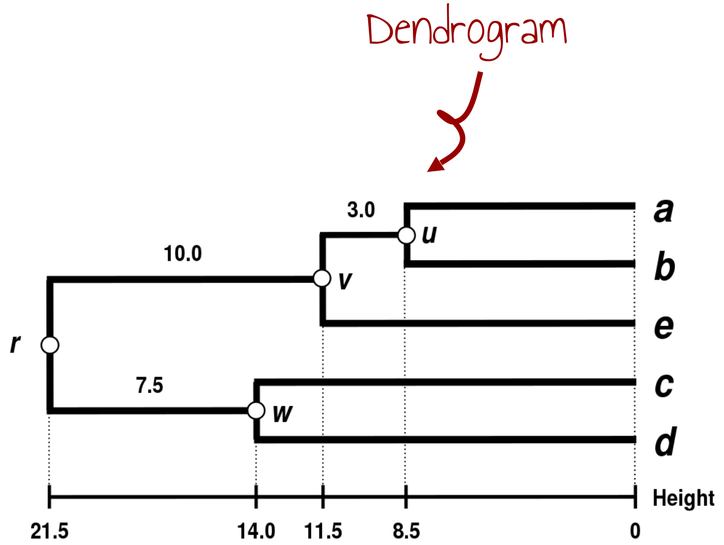


Image from 9.54 MIT

Other clustering techniques

- Hierarchical clustering
  - Agglomerative (bottom-up), divisive (top-down)
- Spectral clustering
- etc.



Scikit-learn has implementations of different algorithms

https://scikit-learn.org/stable/modules/clustering.html

Wikipedia: https://en.wikipedia.org/wiki/Cluster_analysis

To be continued...
(Probability Density Estimation)

Probability Density Estimation
General concepts

Probability Density Function (PDF)

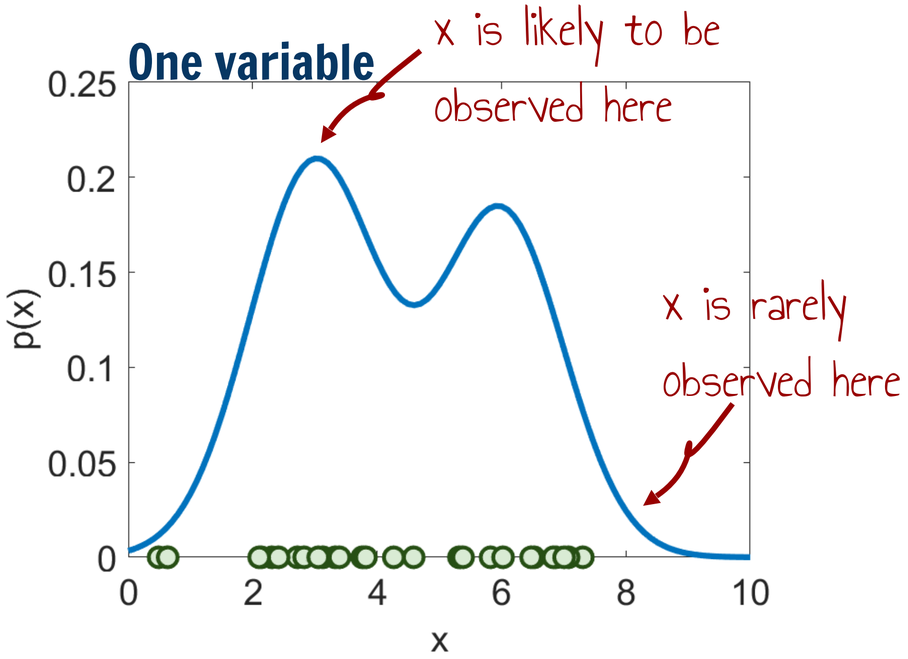
- A PDF $p(x)$ models how likely a continuous variable is to be observed within a particular **interval**

Note: No upper bound!

$$p(x) \geq 0$$

$$\int_a^b p(x)dx = 1$$

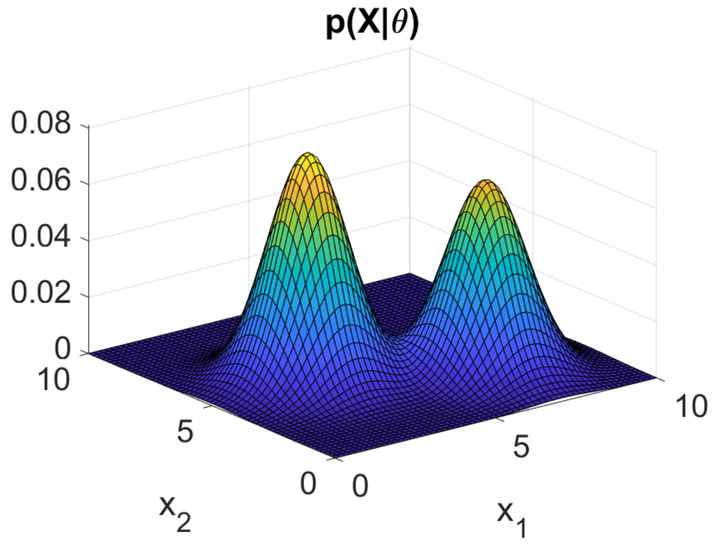
P(x ∈ [a, b])

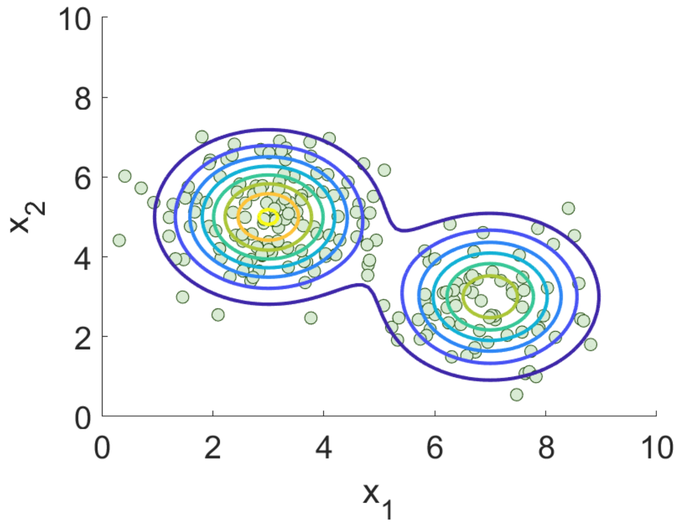


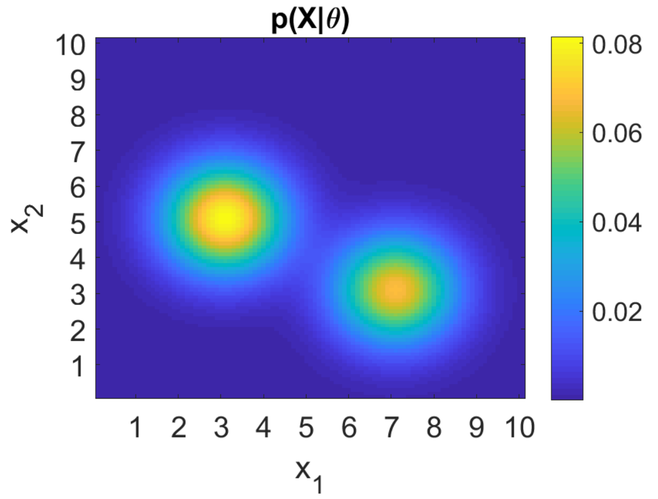
- **Density Estimation**: estimate $p(x)$ from data

Probability Density Function

For 2D variables





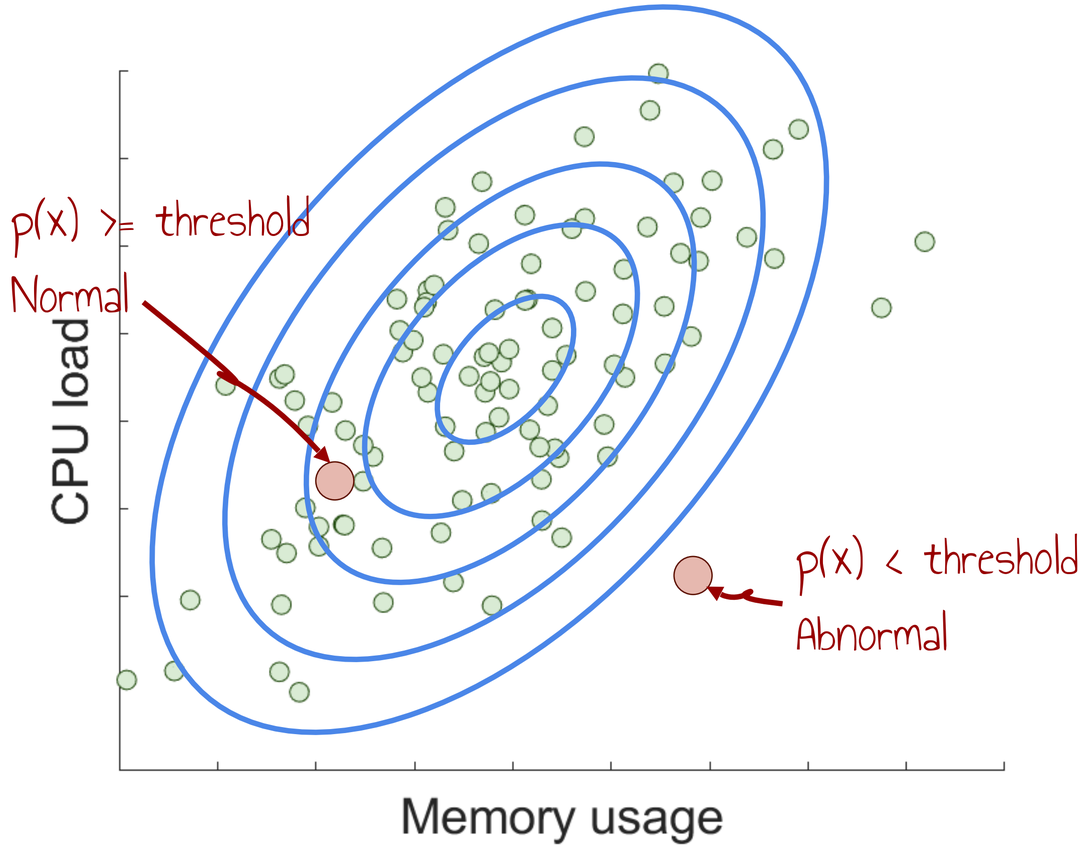


Density Estimation: Applications

- **Anomaly detection** or novelty detection
  - Quality control in manufacturing
  - Credit card fraud detection
  - Security surveillance
  - Detection of viral videos
  - etc.

Density Estimation: Applications

- **Anomaly detection** or novelty detection
  - Example: Monitoring machines in a data center



Density Estimation: Applications

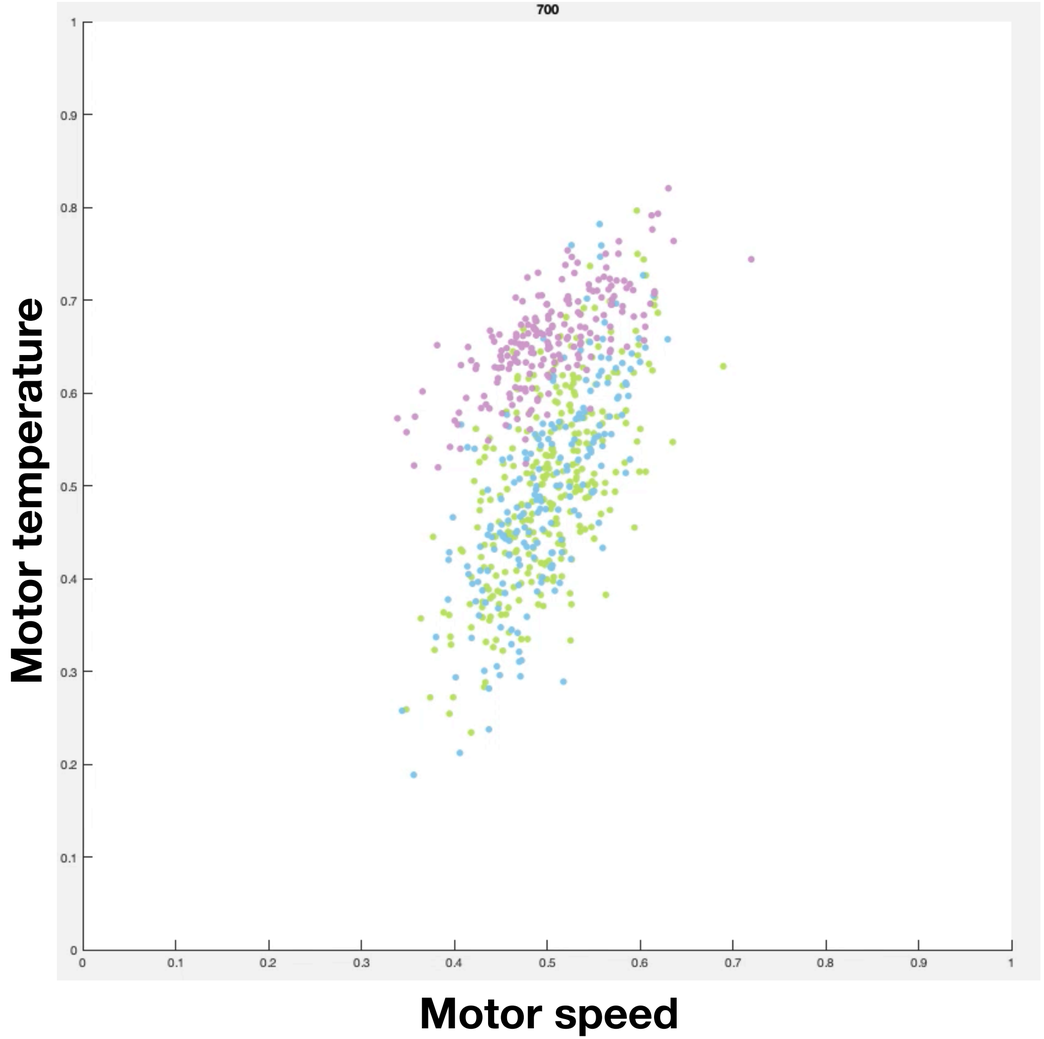
- **Let's take an example:** monitoring jet engines.

- It is important to detect as soon as possible anomalies, to repaire/replace the engines when required.

- In particular, we can measure every minute, the current speed of the engine and its temperature.

Density Estimation: Applications

- **Let's take an example:** monitoring jet engines.
- **Green points** are collected in factories (quality control)
- **Blue points** are collected during flights on recent planes
- **Magenta points** are collected on a suspicious plane



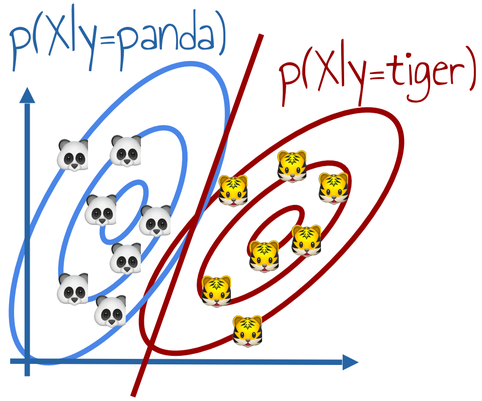
Density Estimation: Applications

- **Generative models**

Model the distribution of a class $p(X|y)$

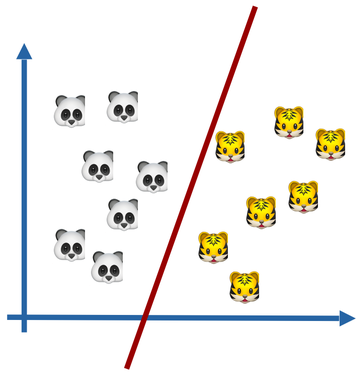
Able to generate new samples ('artist')

Picture from *Shaham et al. ICCV 2019*



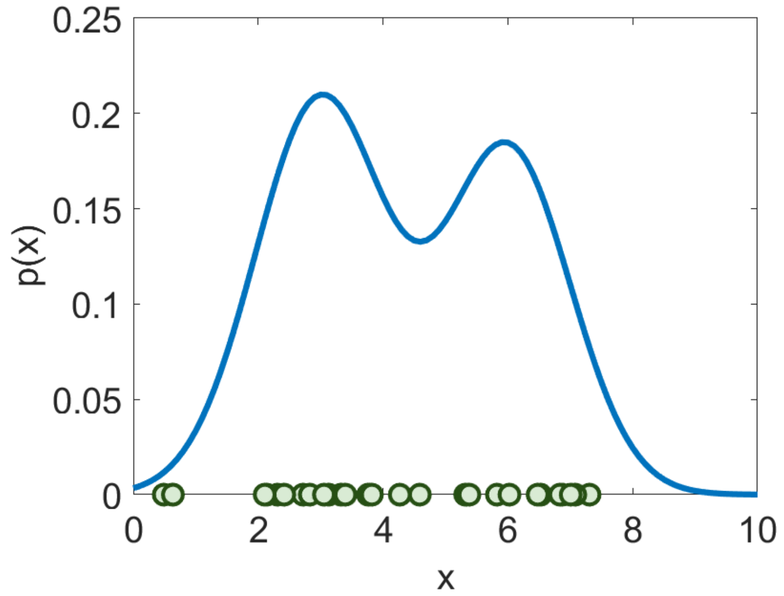
Turn into discriminative classifier: $p(y|X) = \frac{p(X|y)p(y)}{p(X)}$

- Compared to **discriminative models** which directly model $p(y|X)$
  - Neural networks, linear regression, decision trees ...



Back to Density Estimation…

- How do we estimate $p(x)$ from data?



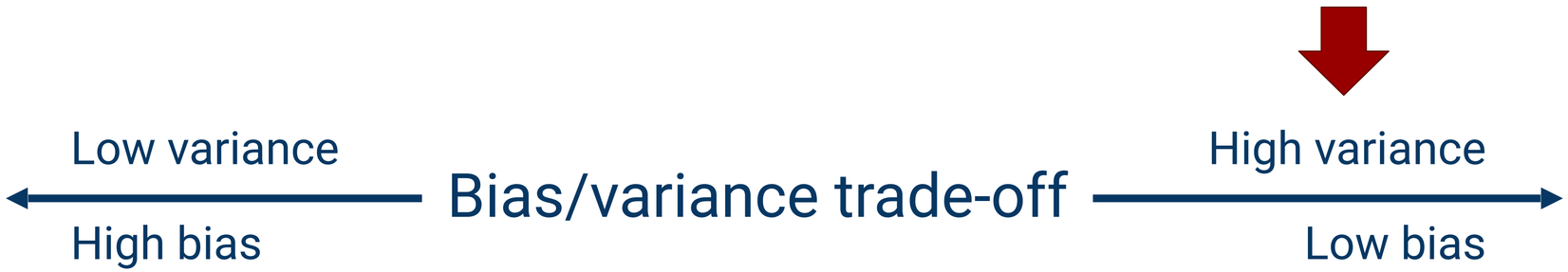
- We'll discuss two classes of approaches:
  - Non-parametric
  - Parametric

To be continued...
(Probability Density Estimation:
Non-parametric approaches)

Probability Density Estimation
Non-parametric approaches

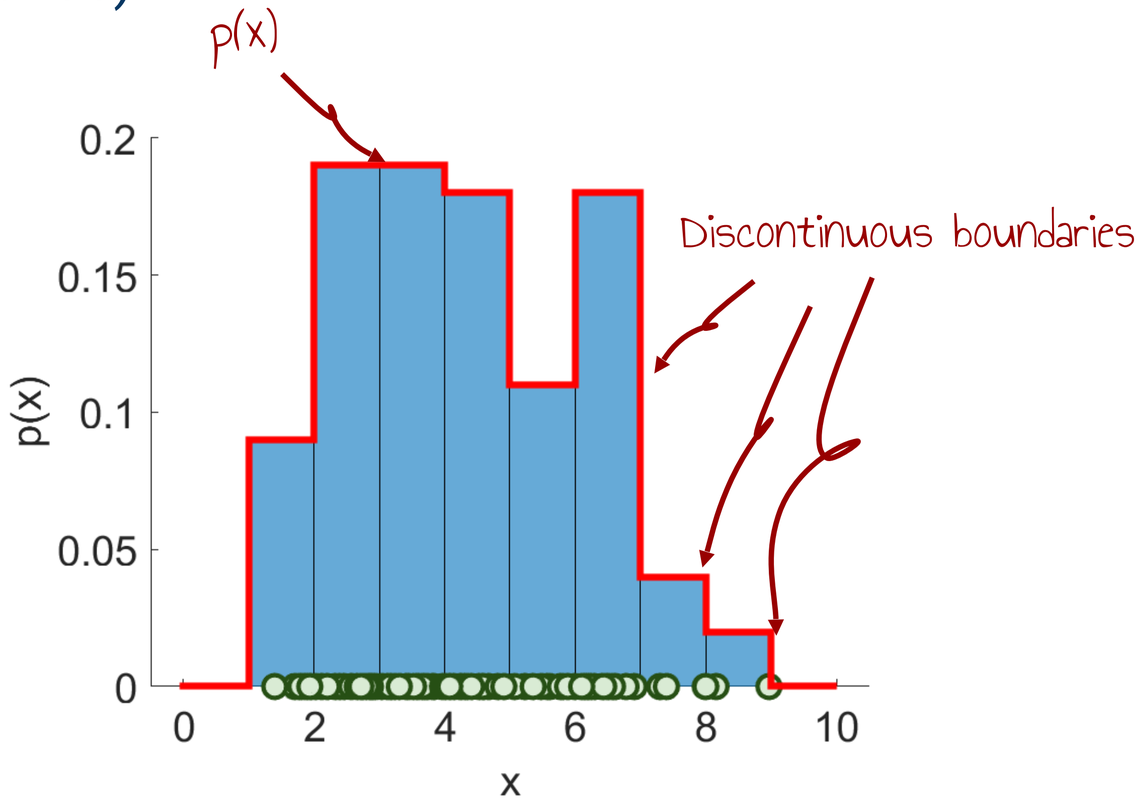
Density Estimation: Non-parametric

- Non-parametric methods
  - Don’t make any assumption about the form
    - Follow the data! (Low bias)
    - (Remember *k*-nearest neighbours?)
  - Number of parameters grow with data



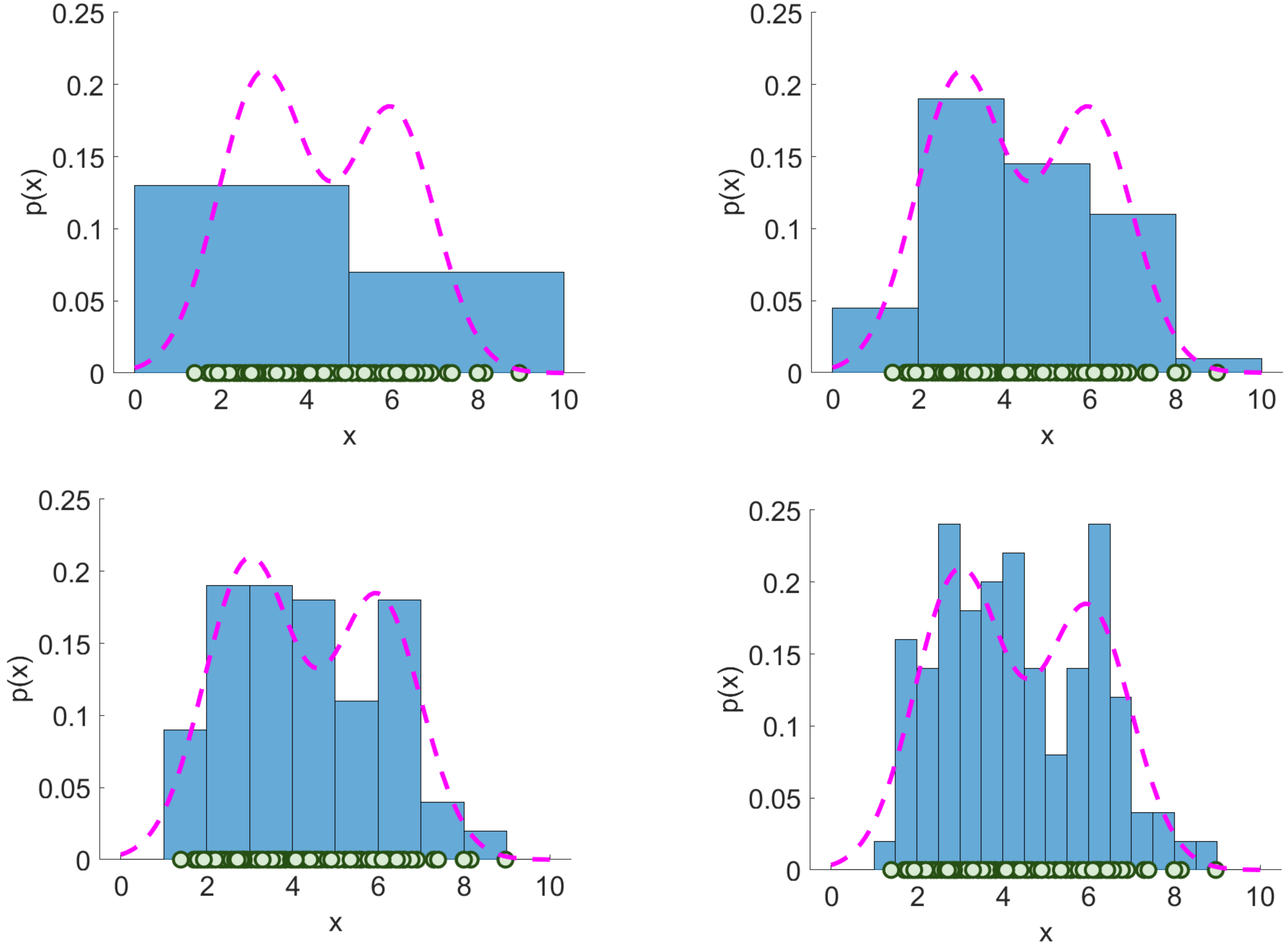
Density Estimation: Histograms

- Histograms
  - Group data into contiguous bins
  - Count the number of occurrences in each bin (and normalise)



Density Estimation: Histograms

- Decreasing the number of bins smooths out $p(x)$



Kernel (or Parzen) Density Estimation

- Kernel density estimation
  - Compute $\hat{p}(x)$ by looking at training examples within a kernel function H (or Parzen window)

$$\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h^D}H\left(\frac{x - x^{(i)}}{h}\right)$$

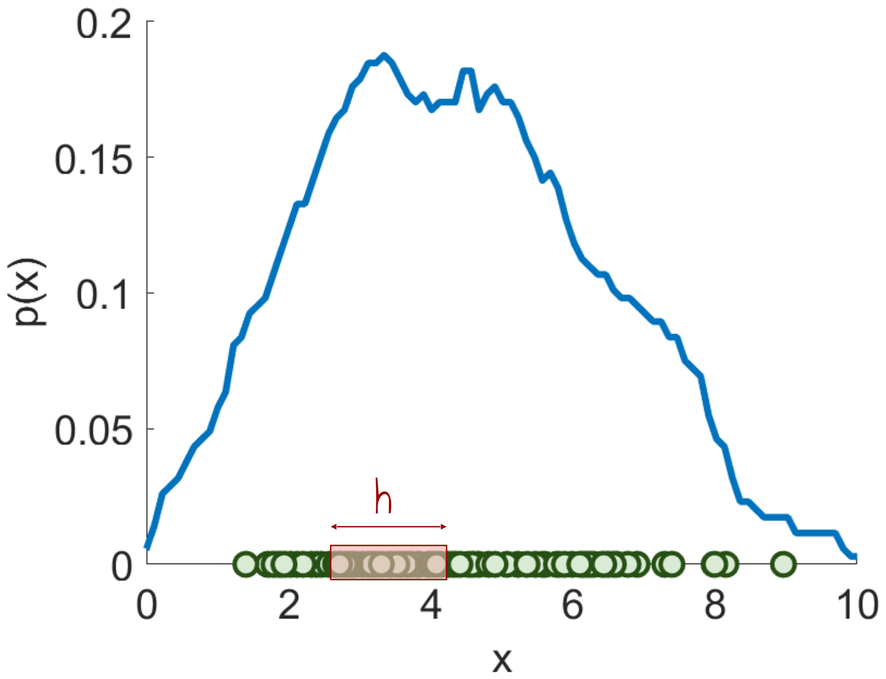
Bandwidth (window size)

Volume of window (D is number of dimensions)

Kernel function

$X^{(i)}$ in inside the window

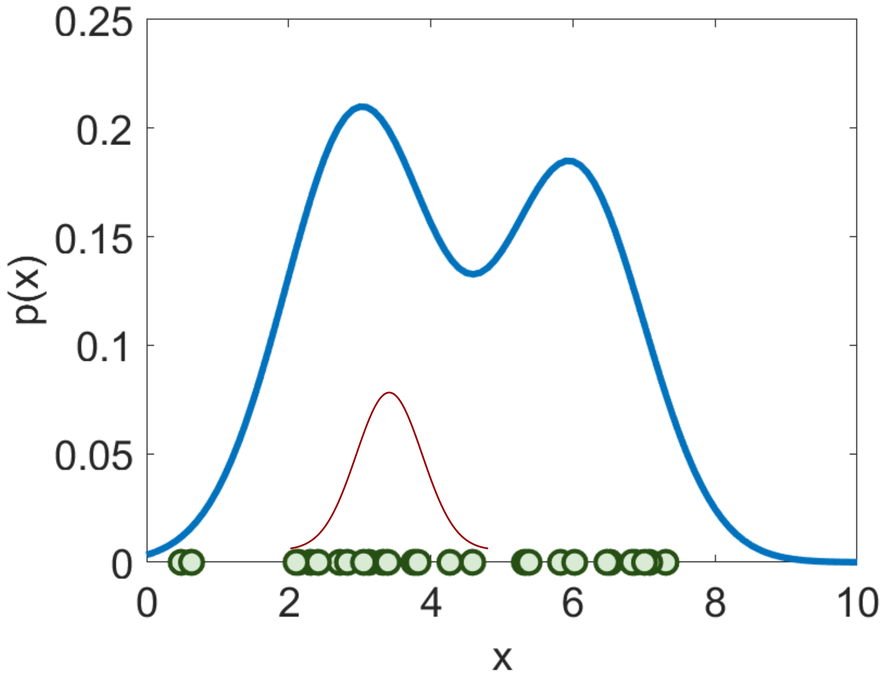
$$H(u) = \begin{cases} 1 & |u_j| < \frac{1}{2}; j = 1, \ldots, D \\ 0 & otherwise \end{cases}$$



Kernel (or Parzen) Density Estimation

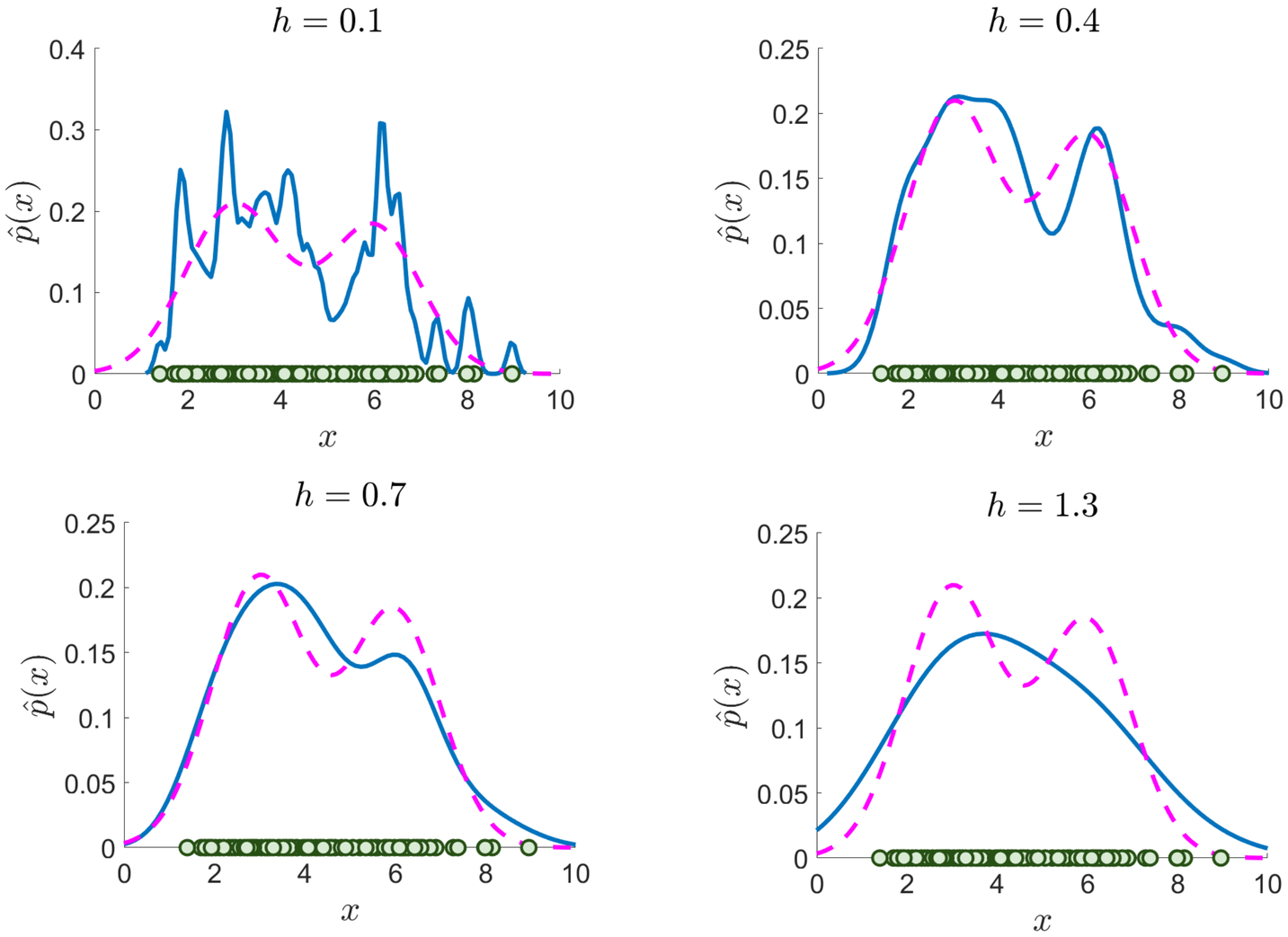
- Kernel density estimation
  - Define kernel function H as a Gaussian: Smoother $\hat{p}(x)$

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{(2\pi h^2)^{\frac{D}{2}}} \exp\left(-\frac{||x - x^{(i)}||^2}{2h^2}\right)$$



Kernel (or Parzen) Density Estimation

- Increasing the bandwidth h smooths out $\hat{p}(x)$

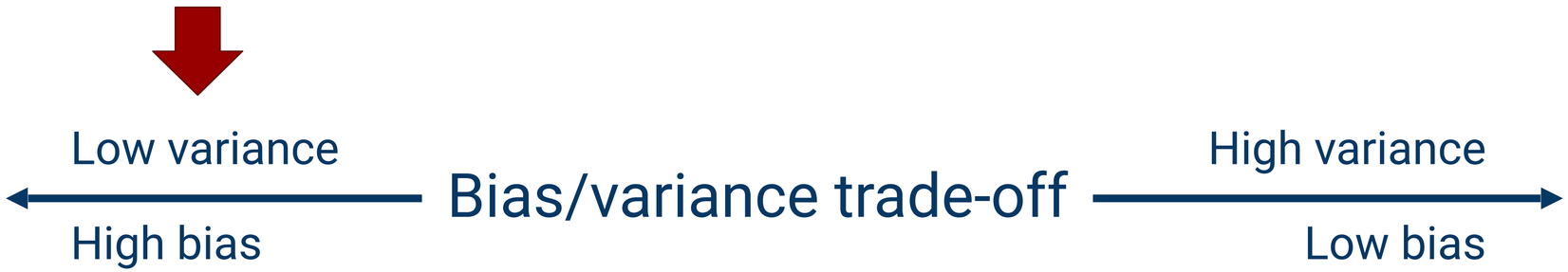


To be continued...
(Probability Density Estimation:
Parametric approaches)

Probability Density Estimation
Parametric approaches

Density Estimation: Parametric

- Parametric methods
  - Makes simplified assumptions about the form
    - High bias
  - Number of parameters is fixed
    - Does not grow with data



Density Estimation: Gaussian distribution

Univariate Gaussian (Normal) distribution

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Larger variance: flatter and more spread out

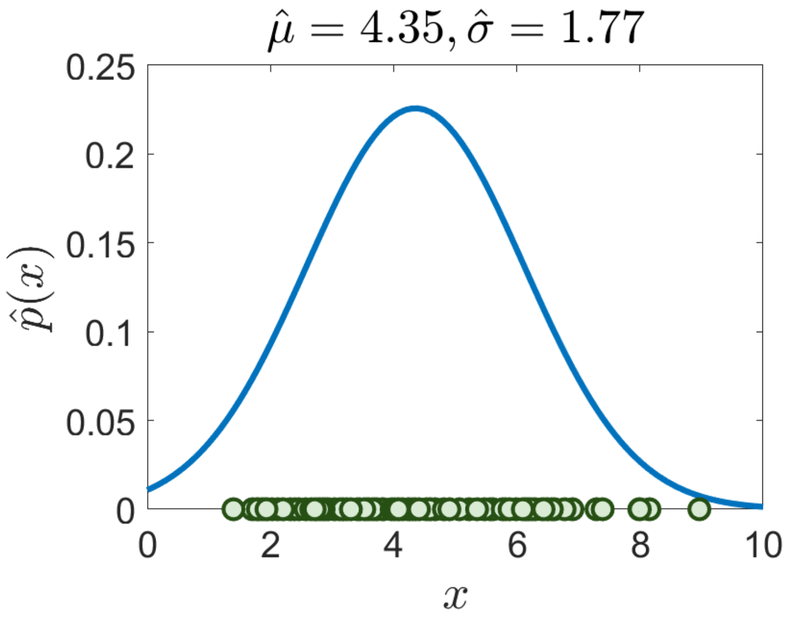
Small variance: more peaked, and tighter

Compute $\hat{p}(x)$ by fitting the parameters $\hat{\mu}$ and $\hat{\sigma}^2$ to the training examples

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} (x^{(i)} - \hat{\mu})^2$$

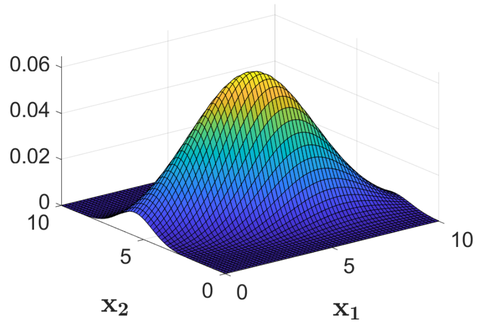
$$\hat{p}(x) = \mathcal{N}(x|\hat{\mu},\hat{\sigma}^2)$$



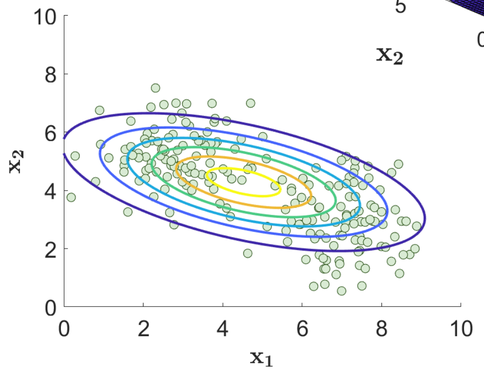
Density Estimation: Gaussian distribution

Multivariate Gaussian (Normal) distribution

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \cdot \exp\left(-\tfrac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$



Compute $\hat{p}(\mathbf{x})$ by fitting the parameters $\hat{\mu}$ and $\hat{\Sigma}$ to the training examples



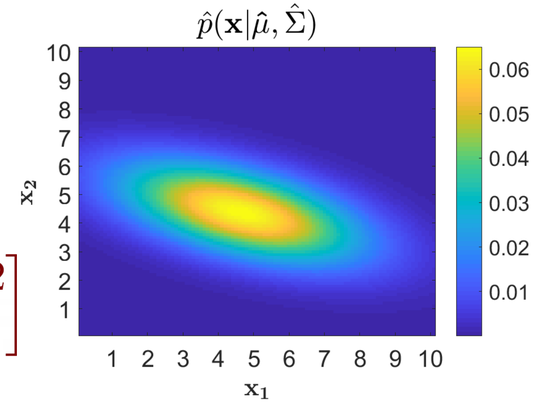
$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)}$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)}-\hat{\mu})(\mathbf{x}^{(i)}-\hat{\mu})^T$$

$$\hat{p}(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\hat{\mu},\hat{\Sigma})$$

$$\hat{\mu} = \begin{bmatrix} 4.53 \\ 4.28 \end{bmatrix}$$

$$\hat{\Sigma} = \begin{bmatrix} 5.56 & -1.52 \\ -1.52 & 1.50 \end{bmatrix}$$



How good is a model?: Likelihood

- How do we quantify that we have fitted a good model?
  - Goal: model captures the probability of generating or observing data *x* within an interval
- **Likelihood**: measures the probability of observing data *x* from a dataset

$$likelihood = p(\mathcal{X}|\theta) = \prod_{i=1}^{N} p(x^{(i)}|\theta)$$

θ: Parameters of your model (mean, covariance)

i.i.d. assumption

- **Negative Log-Likelihood** is often used:

$$\mathcal{L} = -\log p(\mathcal{X}|\theta) = -\sum_{i=1}^{N} \log p(x^{(i)}|\theta)$$

Negative: Turn it to a minimization problem

LOG: for numerical stability (multiplication of small probabilities -> bad!)

Gaussian Distribution: Optimise Likelihood

- Gaussian fitting = minimising the negative log likelihood
- Proof (for univariate Gaussians):

Minimise

$$\mathcal{L} = -\log p(\mathcal{X}|\theta) = -\sum_{i=1}^{N} \log p(x^{(i)}|\theta) = -\sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}\right)$$

$$= \frac{N}{2}\log 2\pi + \frac{N}{2}\log\sigma^2 + \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x^{(i)}-\mu)^2$$

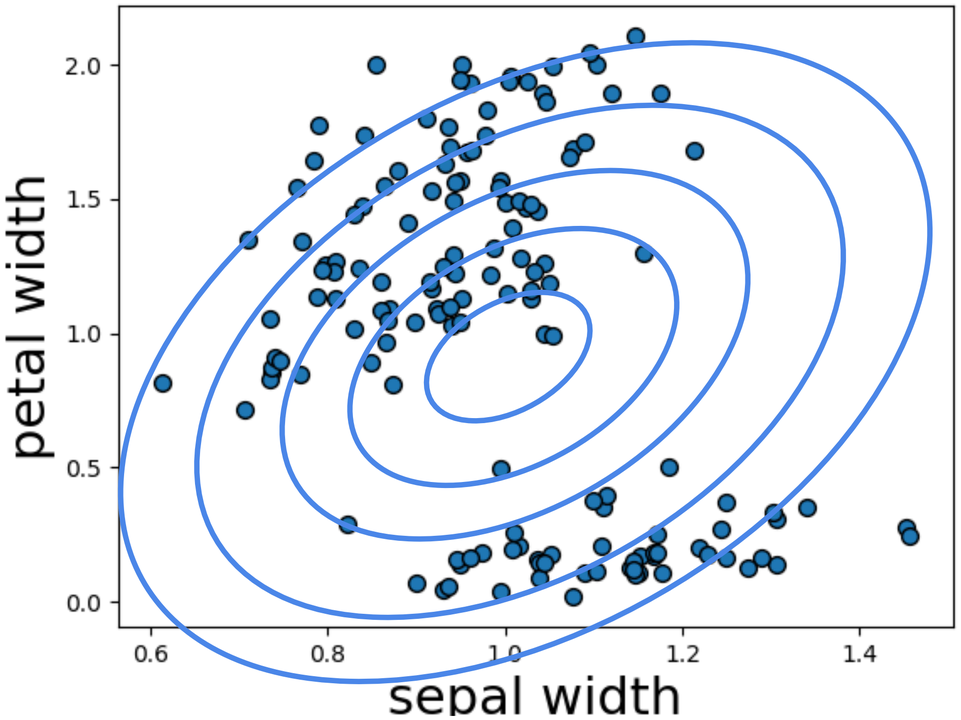
- $\mathcal{L}$ is minimum when derivative=0: $\frac{\partial \mathcal{L}}{\partial \mu} = 0$ $\frac{\partial \mathcal{L}}{\partial \sigma^2} = 0$

$$\frac{\partial \mathcal{L}}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x^{(i)}-\mu) = \frac{1}{\sigma^2}\left(\sum_{i=1}^{N} x^{(i)} - N\mu\right) \qquad \frac{\partial \mathcal{L}}{\partial \mu} = 0 \rightarrow \boxed{\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}}$$

$$\frac{\partial \mathcal{L}}{\partial \sigma^2} = \frac{N}{2}\frac{1}{\sigma^2} - \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{N}(x^{(i)}-\mu)^2 \qquad \frac{\partial \mathcal{L}}{\partial \sigma^2} = 0 \rightarrow \boxed{\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x^{(i)}-\hat{\mu})^2}$$

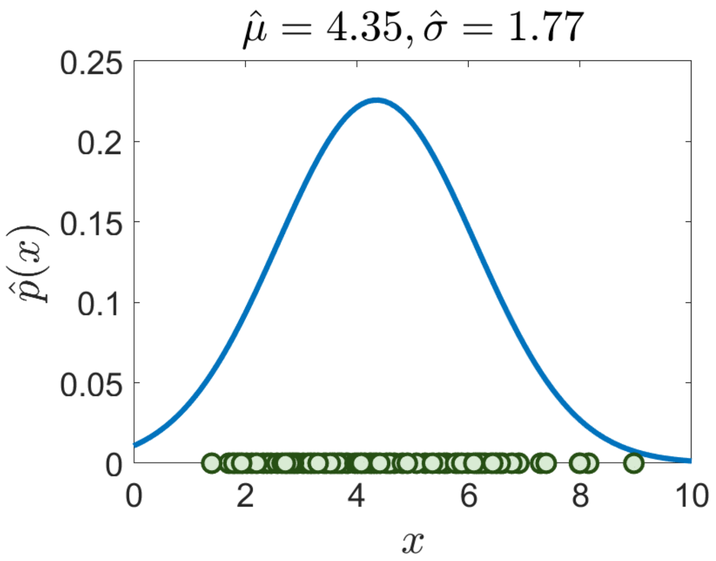
Density Estimation: Better models?

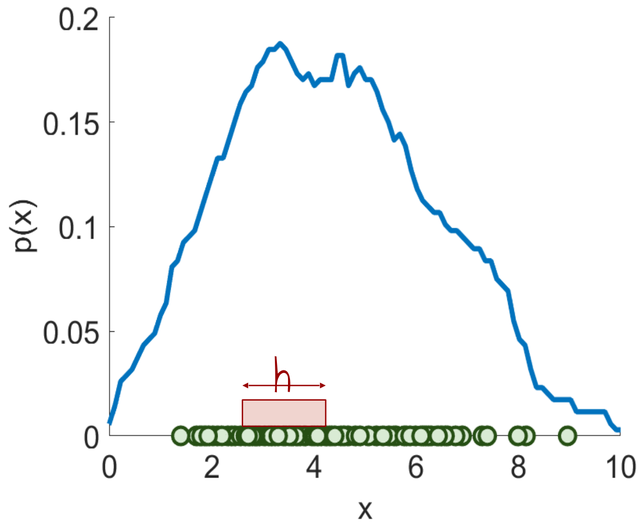
- Is a Gaussian distribution enough for modelling densities?



Density Estimation: Better Model?

- Something that gives the best of both worlds?





Low variance ← Bias/variance trade-off → High variance

High bias ← Bias/variance trade-off → Low bias

Density Estimation: Mixture Models

Mixture Models

$$p(x) = \sum_{k=1}^{K} \pi_k p_k(x)$$

Mixing proportions

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

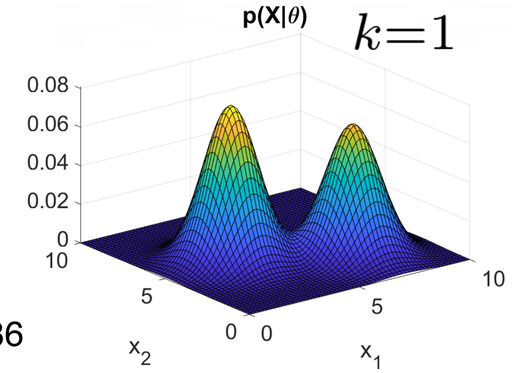
$$\sum_{k=1}^{K} \pi_k = 1$$

Gaussian Mixture Models (most popular)

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \Sigma_k)$$

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

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



$$\theta = \{\pi_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$$

To be continued...
(GMM-EM)

Probability Density Estimation
GMM-EM

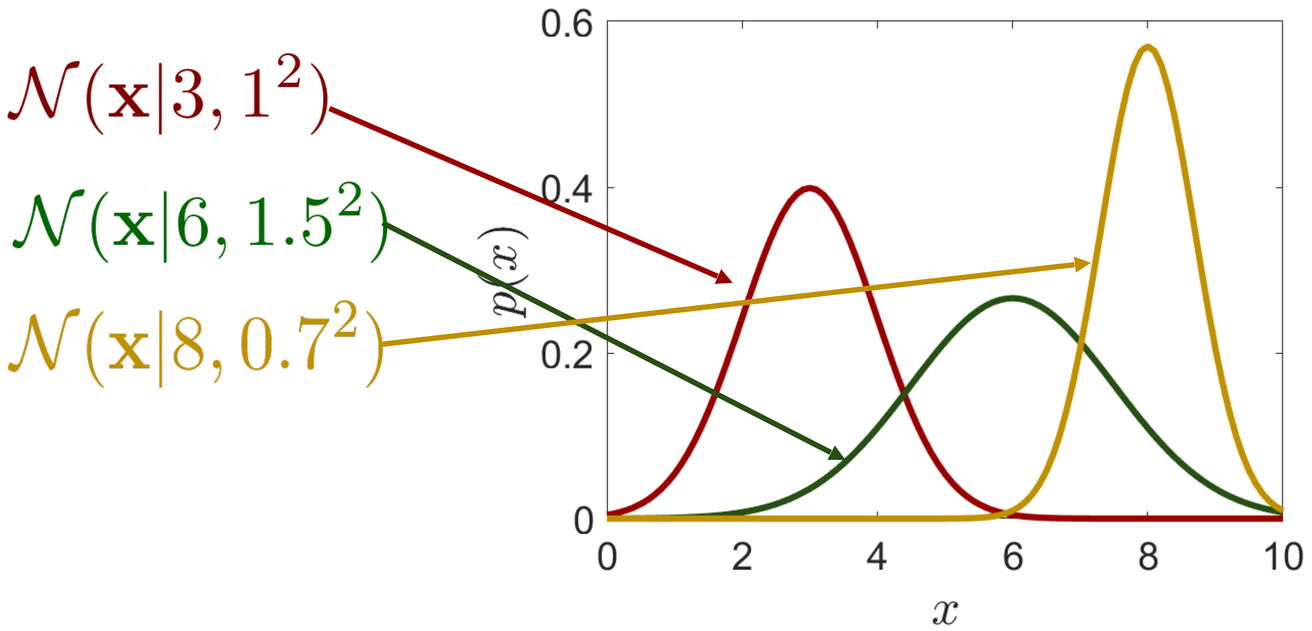
Density Estimation: GMMs

Gaussian **M**ixture **M**odel: Weighted mixture of Gaussians!

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \Sigma_k)$$

$$0 \leq \pi_k \leq 1 \qquad \sum_{k=1}^{K} \pi_k = 1$$

$$\theta = \{\pi_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$$



Density Estimation: GMMs

Gaussian Mixture Model: Weighted mixture of Gaussians!

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \Sigma_k)$$

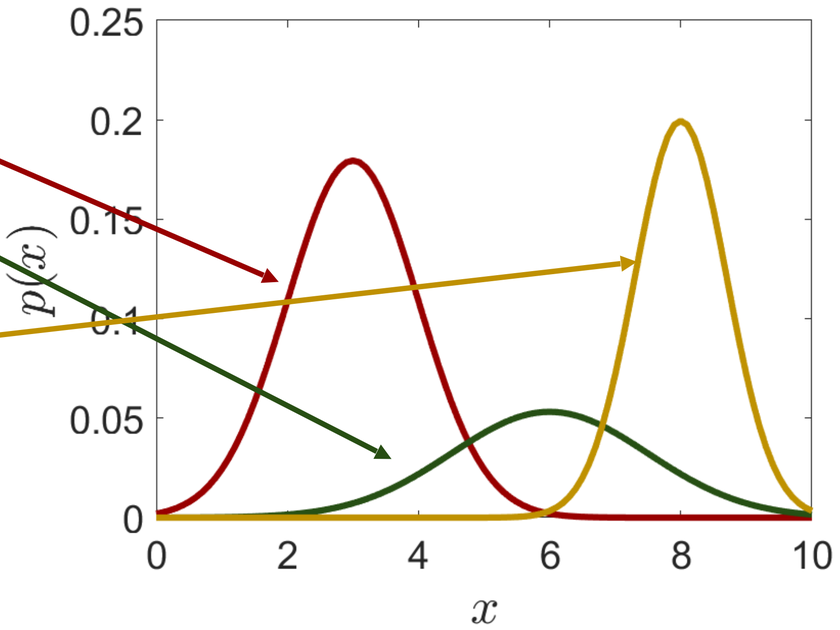
$$0 \leq \pi_k \leq 1 \qquad \sum_{k=1}^{K} \pi_k = 1$$

$$\theta = \{\pi_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$$

$0.45 \times \mathcal{N}(\mathbf{x}|3, 1^2)$

$0.20 \times \mathcal{N}(\mathbf{x}|6, 1.5^2)$

$0.35 \times \mathcal{N}(\mathbf{x}|8, 0.7^2)$



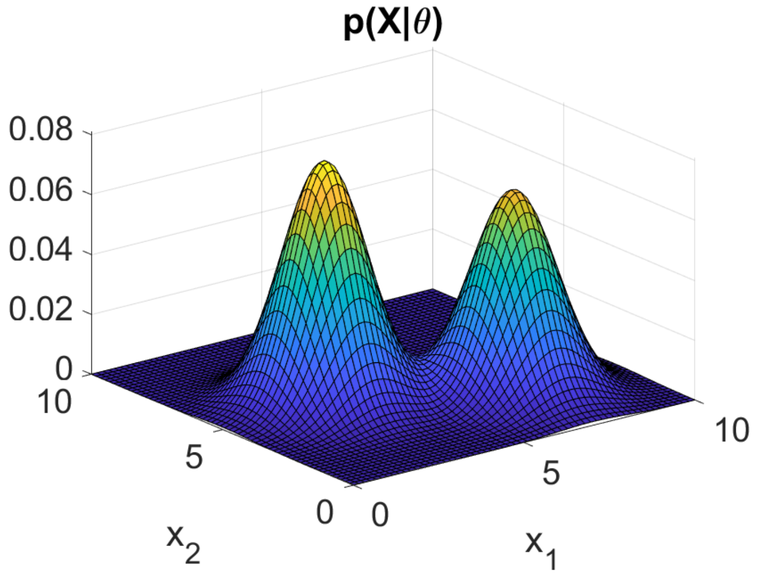
Density Estimation: GMMs

Gaussian Mixture Model: Weighted mixture of Gaussians

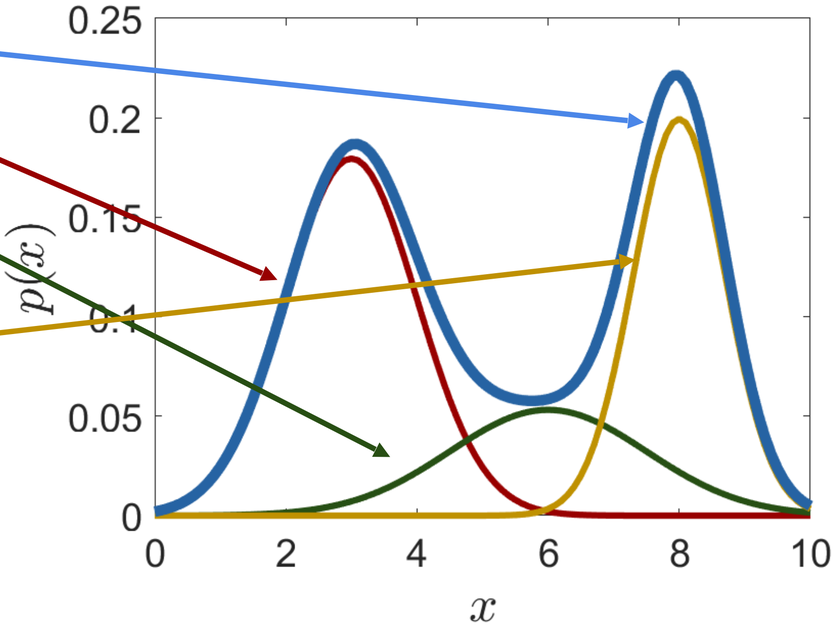
$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \Sigma_k)$$

$$0 \leq \pi_k \leq 1 \qquad \sum_{k=1}^{K} \pi_k = 1$$

$$\theta = \{\pi_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$$



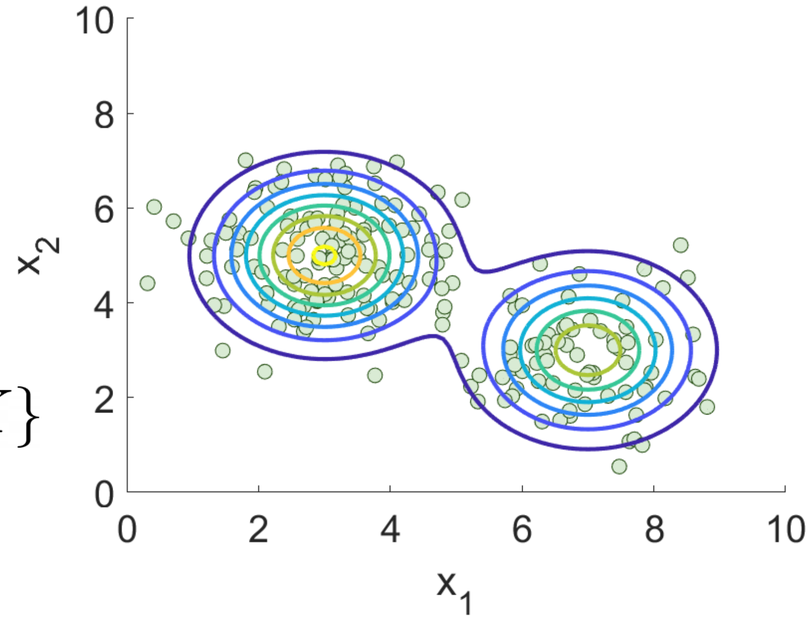
$$p(\mathbf{x}|\theta) =$$
$$0.45 \text{ x } \mathcal{N}(\mathbf{x}|3, 1^2)$$
$$+ 0.20 \text{ x } \mathcal{N}(\mathbf{x}|6, 1.5^2)$$
$$+ 0.35 \text{ x } \mathcal{N}(\mathbf{x}|8, 0.7^2)$$



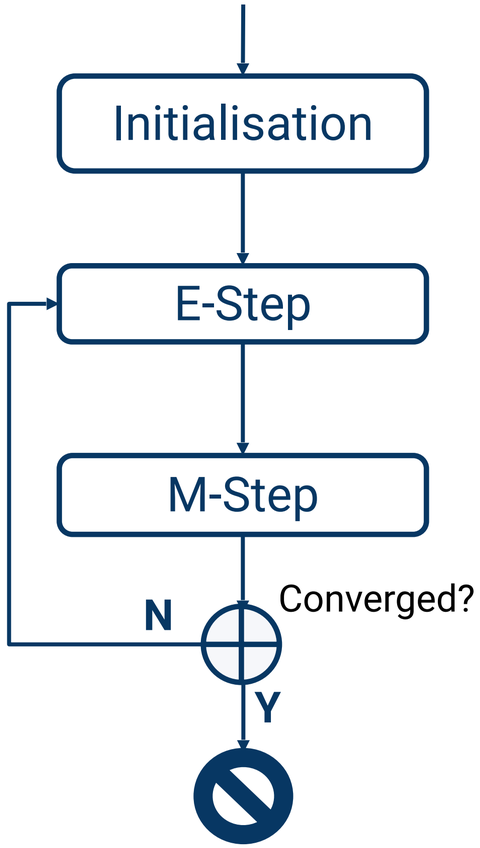
Density Estimation: Fitting a GMM

- GMM's can model complicated data distributions
  - ...but have more parameters to optimise!
- How do we fit the GMM to training examples?
  - Maximise the likelihood?
  - We can't. The update of each parameter depends on other parameters.

$$\theta = \{\pi_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$$



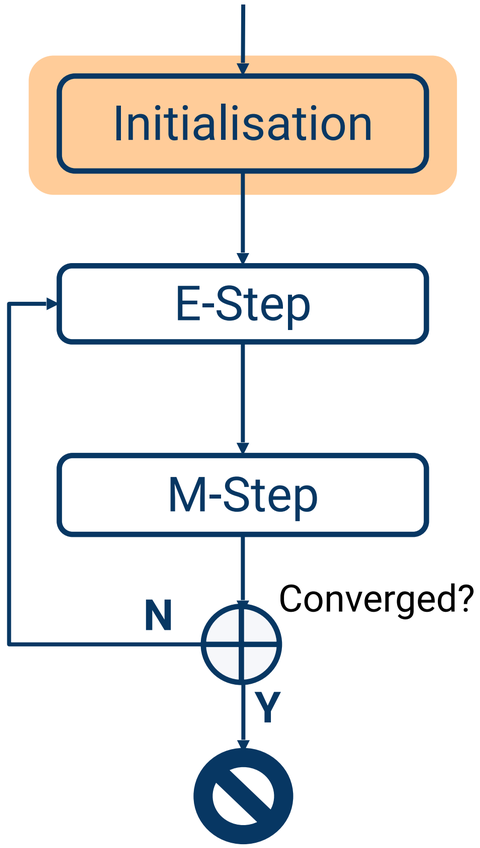
Expectation-**M**aximisation to the rescue!



- To fit the parameters of the GMM, we use an iterative approach
  - Sounds familiar?
- EM can also be used with other mixture models
- The two main steps are the
  - E-step (Expectation)
  - M-step (Maximisation)

For proofs, you can refer to the Mathematics for Machine Learning book, by Deisenroth, Faisal, and Ong (chapter 11).
https://mml-book.github.io/

Expectation-Maximisation (EM) algorithm

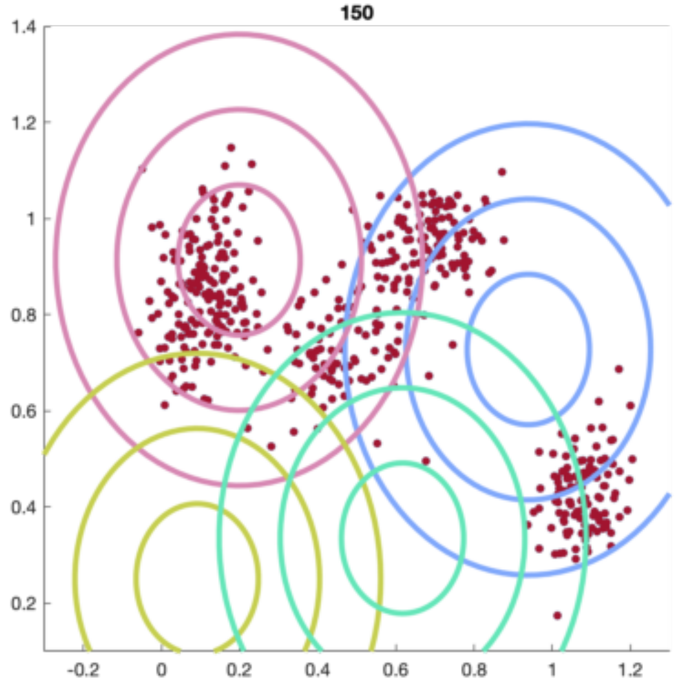


Step 1: Initialisation

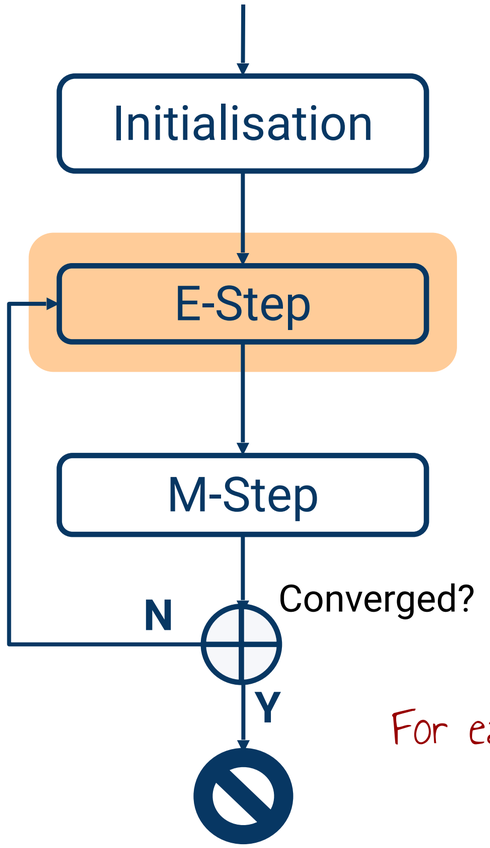
- Choose a *K*
- Randomly initialise parameters

$$\theta = \{\pi_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$$

$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leq \pi_k \leq 1$$



Expectation-Maximisation (EM) algorithm

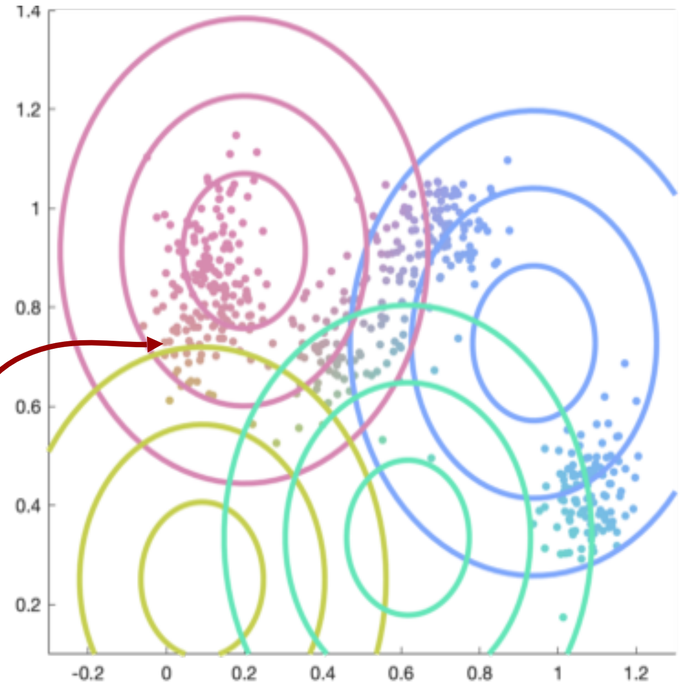


Step 2: E-Step

- Compute the **responsibilities** for each training example $x^{(i)}$ and each mixture component k

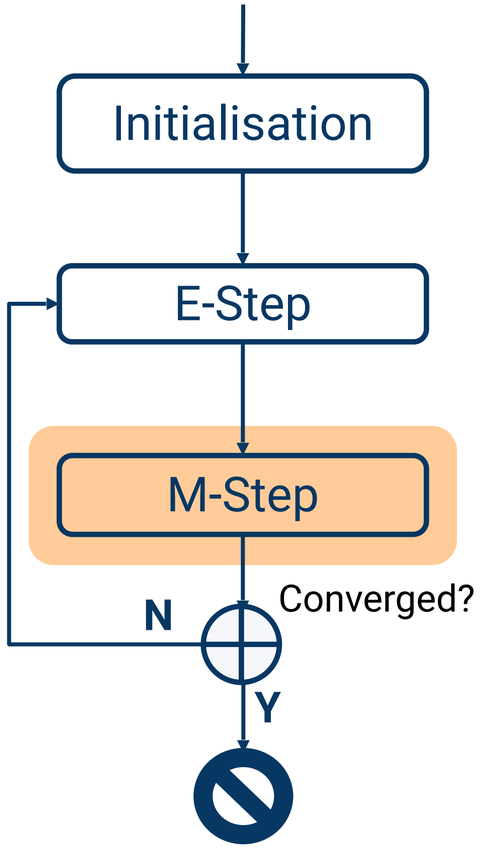
$$r_{ik} = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)} | \mu_k, \Sigma_k)}{\sum_j^K \pi_j \mathcal{N}(\mathbf{x}^{(i)} | \mu_j, \Sigma_j)}$$

For each i AND k



Notice the dot colour changed

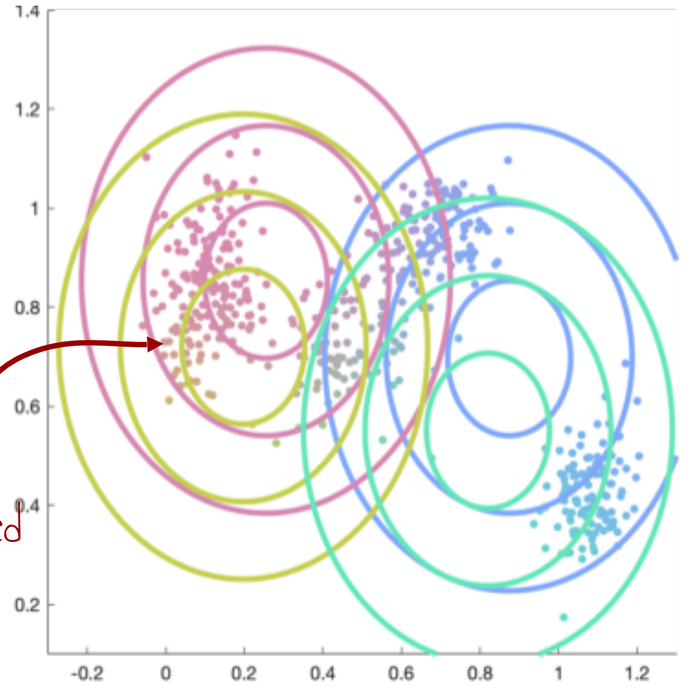
Expectation-Maximisation (EM) algorithm



Step 3: M-Step

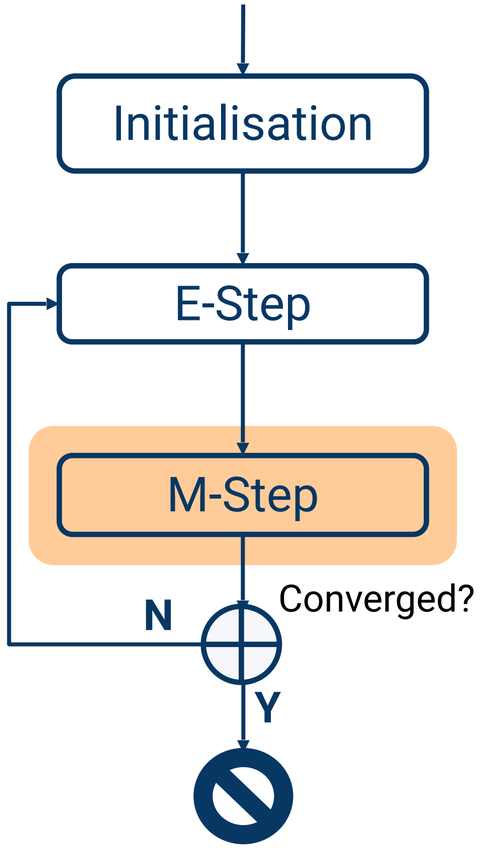
- Update the **mean** of all mixture component:

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} r_{ik} \mathbf{x}^{(i)} \qquad N_k = \sum_{i=1}^{N} r_{ik}$$



Notice the gaussians' centres moved

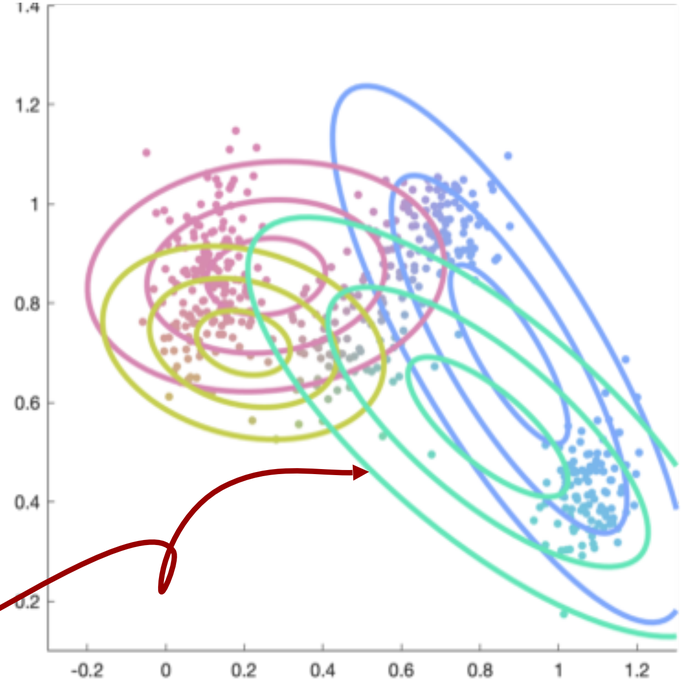
Expectation-Maximisation (EM) algorithm



Step 3: M-Step

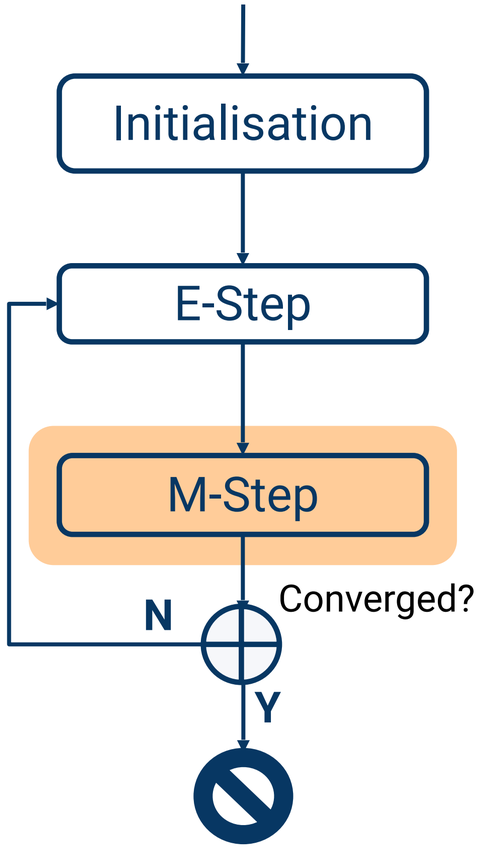
- Update the **mean** of all mixture components.
- Update the **covariance** of all mixture components:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} r_{ik} \left(\mathbf{x}^{(i)} - \hat{\mu}_k\right) \left(\mathbf{x}^{(i)} - \hat{\mu}_k\right)^T \qquad N_k = \sum_{i=1}^{N} r_{ik}$$



Notice shapes changed

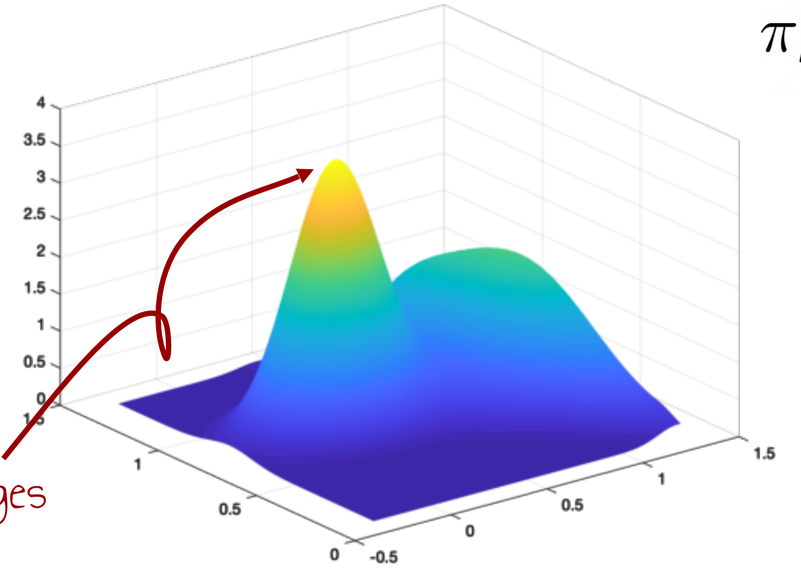
Expectation-Maximisation (EM) algorithm



Step 3: M-Step

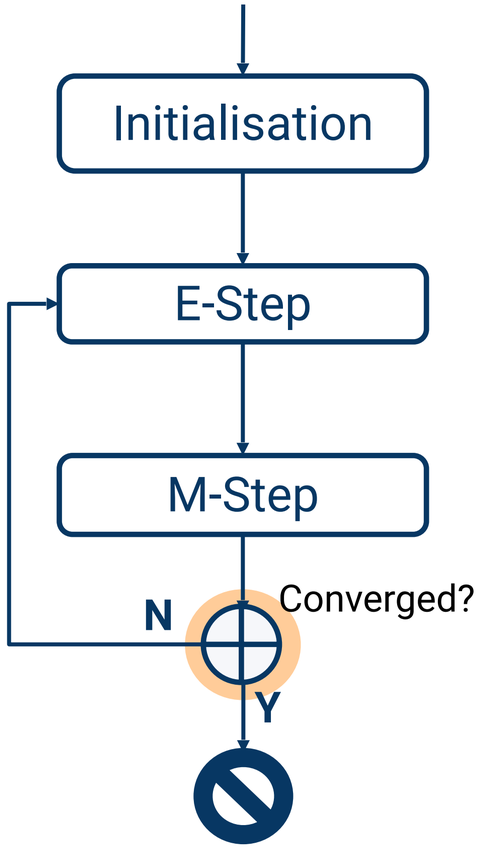
- Update the **mean** of all mixture components.
- Update the **covariance** of all mixture components.
- Update the **mixing proportion** of all mixture components:

$$\pi_k = \frac{N_k}{N}$$



Notice: the "height" changes

Expectation-Maximisation (EM) algorithm

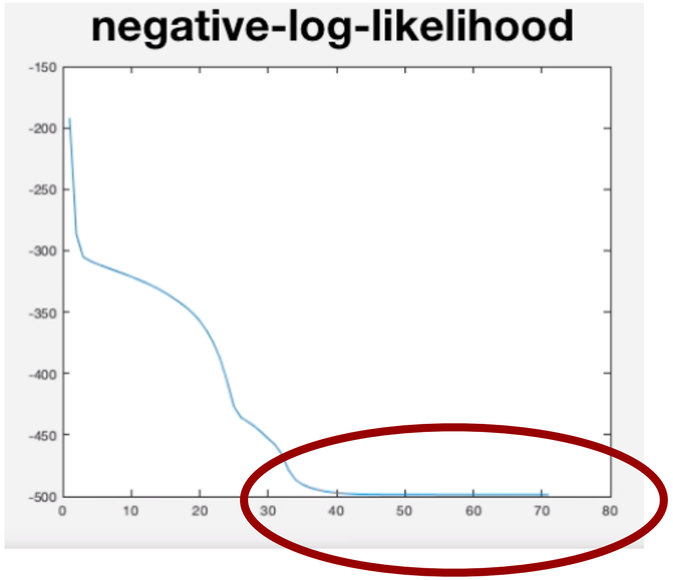


Step 4: Convergence check

- No significant change in parameters

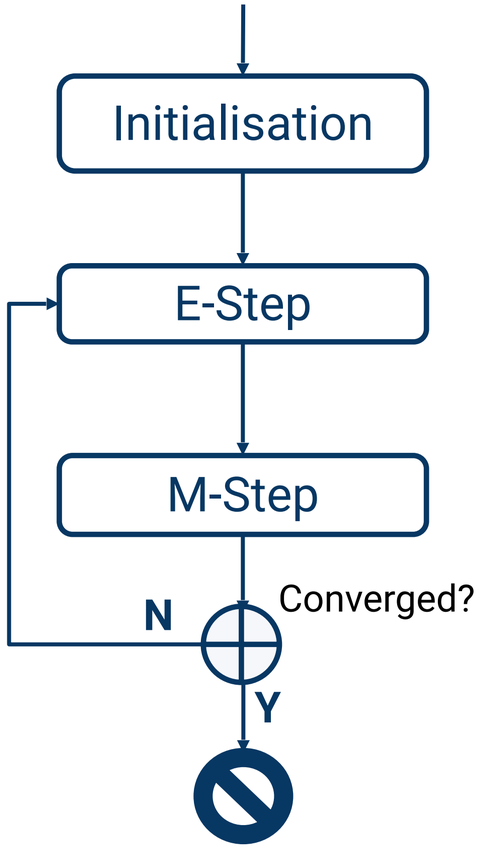
$$\theta = \{\pi_k, \mu_k, \Sigma_k : k = 1, \ldots, K\}$$

- Or stagnation of the likelihood



Note: EM converges to local optimum

GMM-EM: Summary



Step 1: Initialisation

- Select *K*, randomly initialise all parameters

Step 2: E-Step

- Compute the responsibilities

Step 3: M-Step

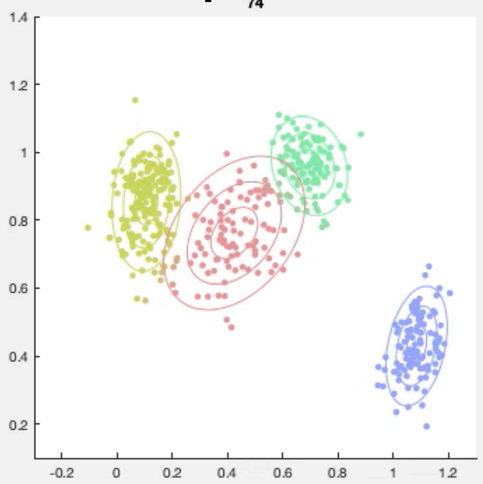
- Update the mean
- Update the covariance
- Update the mixing proportions

Step 4: Convergence check

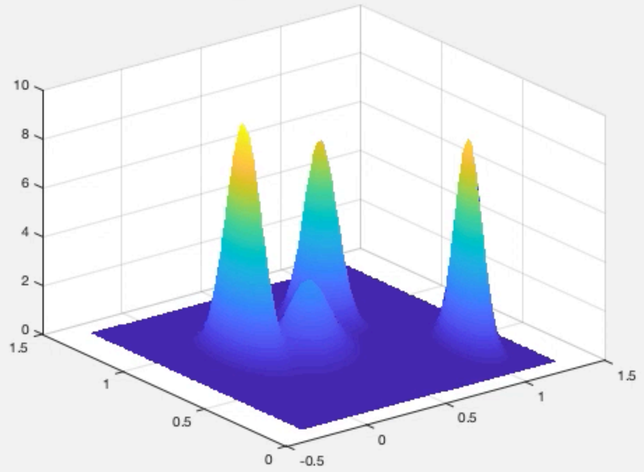
- Stop if converged (parameters not changed, or likelihood stagnated). Otherwise go to Step 2.

GMM-EM: Summary

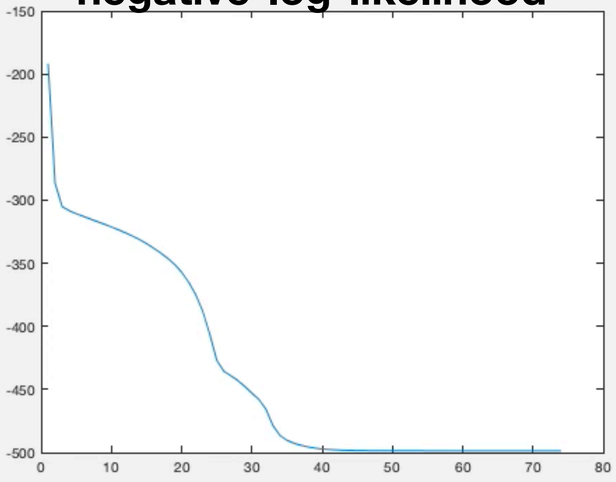
Top view



$p(\boldsymbol{x} \mid \boldsymbol{\theta})$



negative-log-likelihood



GMM-EM pseudo-code:

Step 1: Initialisation

- Select *K*, randomly initialise all parameters

Step 2: E-Step

- Compute the responsibilities

Step 3: M-Step

- Update the mean
- Update the covariance
- Update the mixing proportions

Step 4: Convergence check

- Stop if converged (parameters not changed, or likelihood stagnated). Otherwise go to Step 2.

To be continued...
(GMM-EM: Hyper-parameter tuning,
strengths and weaknesses)

GMM-EM:
Hyper-parameter tuning, strengths and weaknesses

GMM: Selecting number of components *K*

- Minimize the **Bayesian Information Criterion (BIC)**

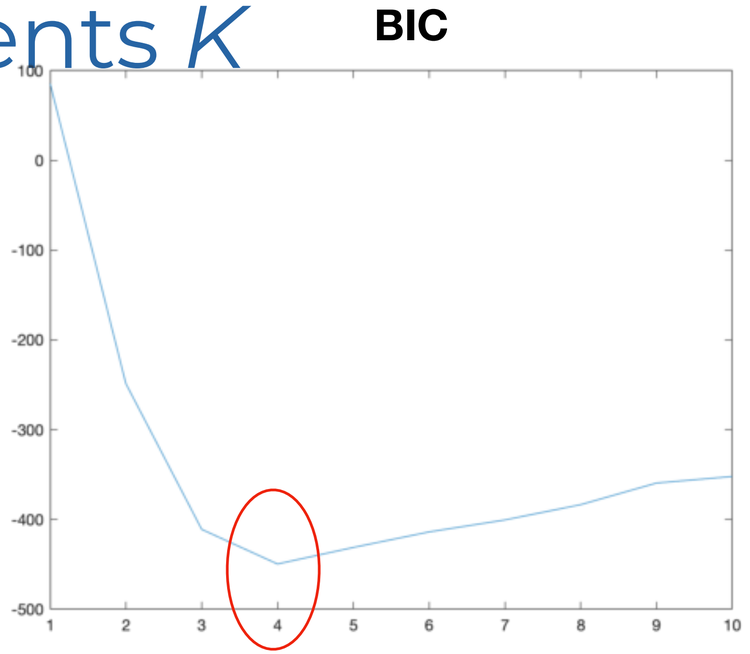
Number of parameters

Number of examples

$$BIC_K = \mathcal{L}(K) + \frac{P_K}{2} log(N)$$

Negative Log Likelihood (encourages fitting of data)

Penalises complex models

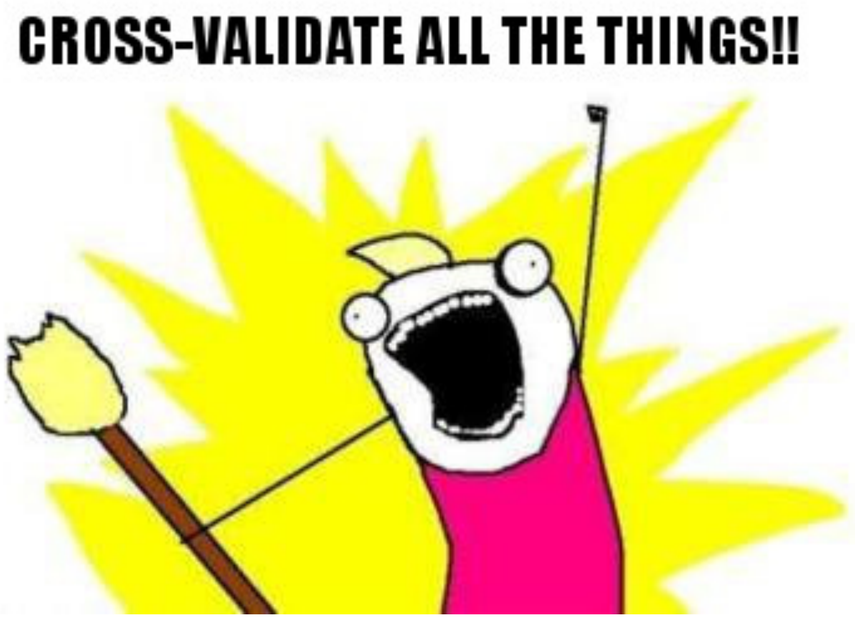


Number of parameters for 2D Gaussians: $P_K = 6*K - 1$
(2 for mean, 3 for covariance, 1 for mixing proportion)

Ockham's (Occam's) razor

"Pick the simplest of all models that fit"

GMM: Selecting number of components *K*



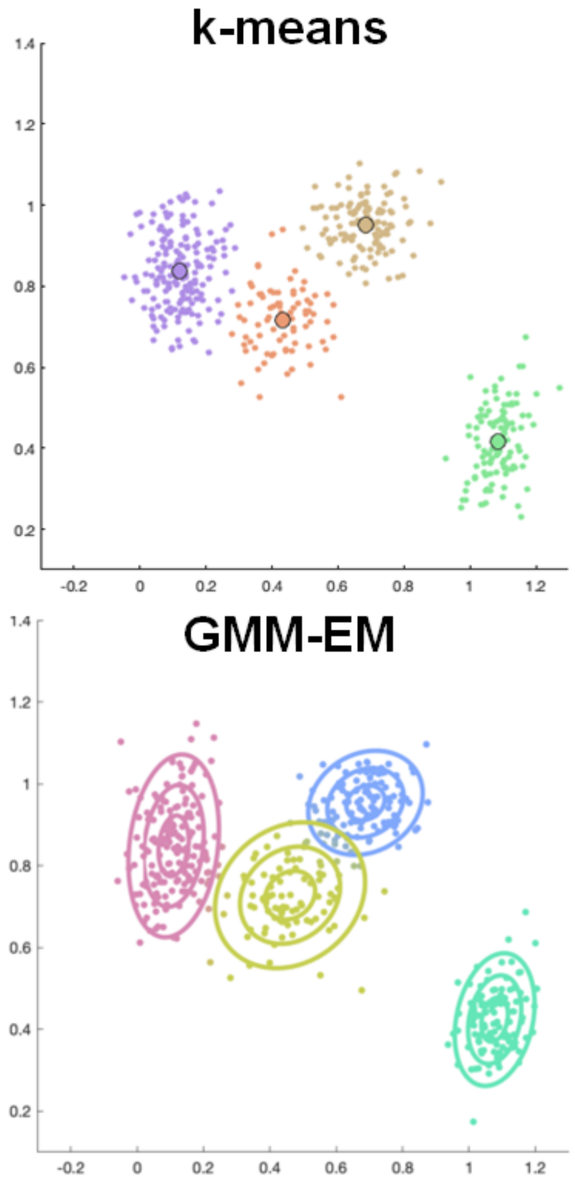
- **(Cross-)validation**
  - Split into training and validation sets.
  - Run GMM-EM on the training set with different K's, and evaluate the likelihood on the validation set.
  - (Same procedure as K-means).

GMM-EM vs *K*-means

There are many connections between *K*-means and GMM-EM.

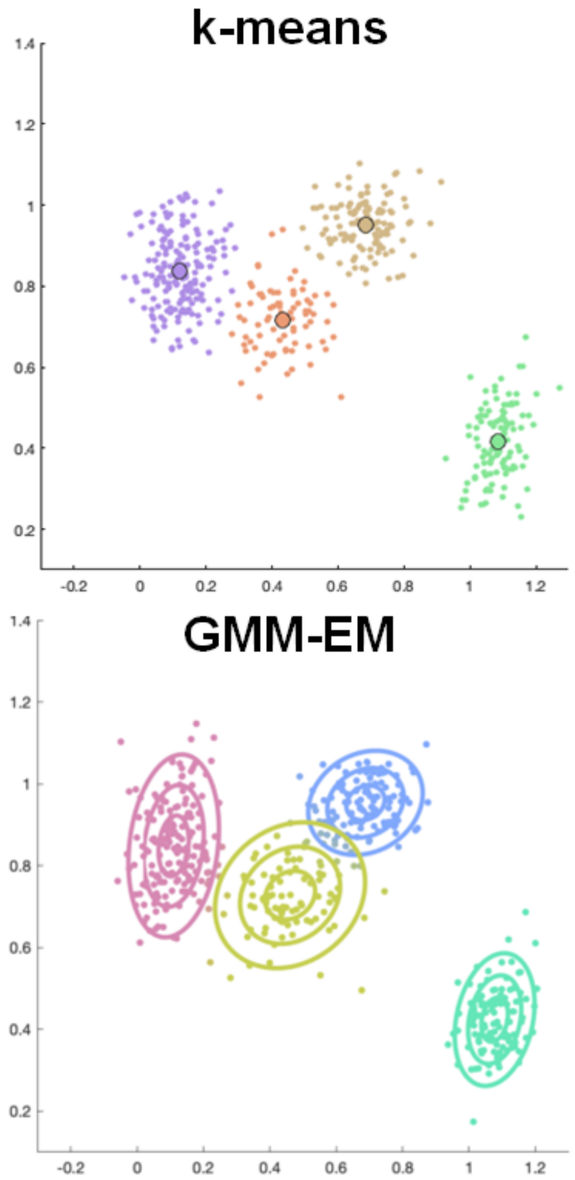
Similar to *K*-means:

- We have to select *K*
- Convergence happens when changes are sufficiently small
- Sensitive to initialisation
  (We often initialise the means of GMM-EM from the result of *K*-means)



GMM-EM vs *K*-means

- We have seen GMM-EM from the perspective of **density estimation**.
- But like *K*-means, it can also be used for **clustering**.
- We can see each mixture component as a “source” that generates data points with a different probability (the mixing proportions).



GMM-EM vs *K*-means

- GMM-EM = “soft *K*-means clustering”
- Hard clustering (k-means): Every point belongs to exactly one cluster
- Soft clustering (GMM): Every point belongs to several clusters in certain degrees
  - Each mixture component represents a different cluster (and the responsibilities define the probability of each data point to belong to the clusters).
  - The distance to the centroids (here, the means) is not isotropic and varies during the learning process
    - The distance is actually related to the Mahalanobis distance.

GMM-EM vs *K*-means

K-means	GMM-EM
Objective function: Minimize average mean squared distance	Objective function: Maximize log-likelihood
EM-like algorithm - Assignment: Assign points to cluster - Update: Optimise cluster	EM algorithm - E-step Compute posterior probability of membership - M-step: Optimise parameters
Performs hard assignment during Assignment step	Performs soft assignment during E-step
Assume spherical clusters with equal probability for each cluster	Can be used for non-spherical clusters Can generate clusters with different probabilities

See you next week!