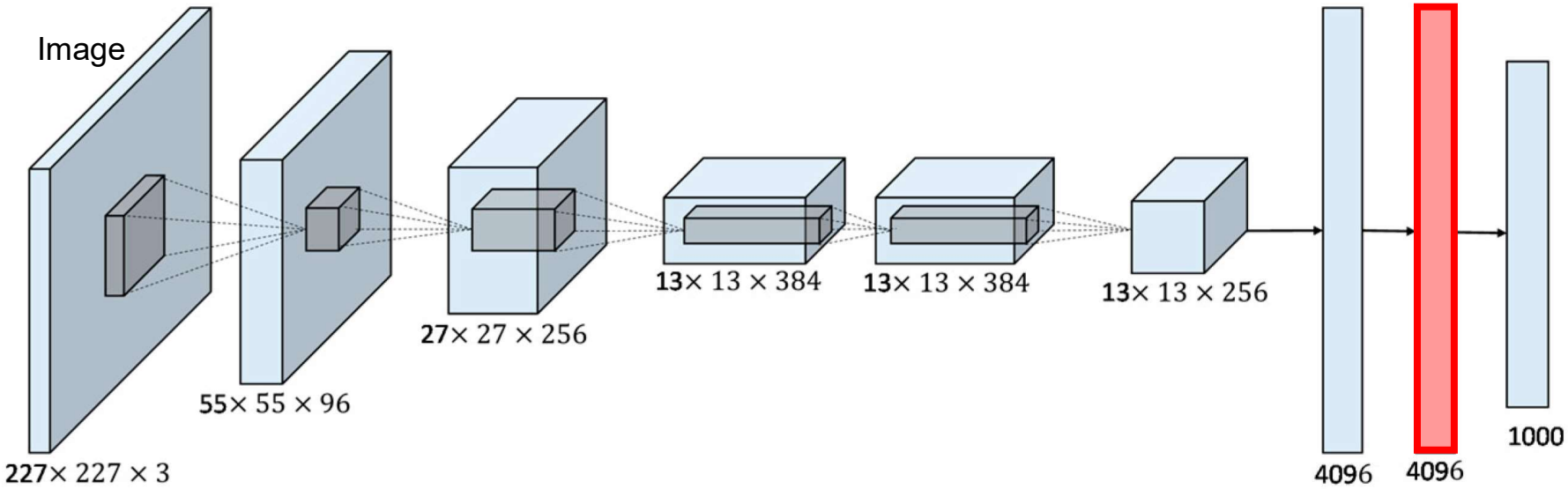
THE UNIVERSITY OF MELBOURNE

Unsupervised learning, mixture models

Semester 1, 2021

Kris Ehinger

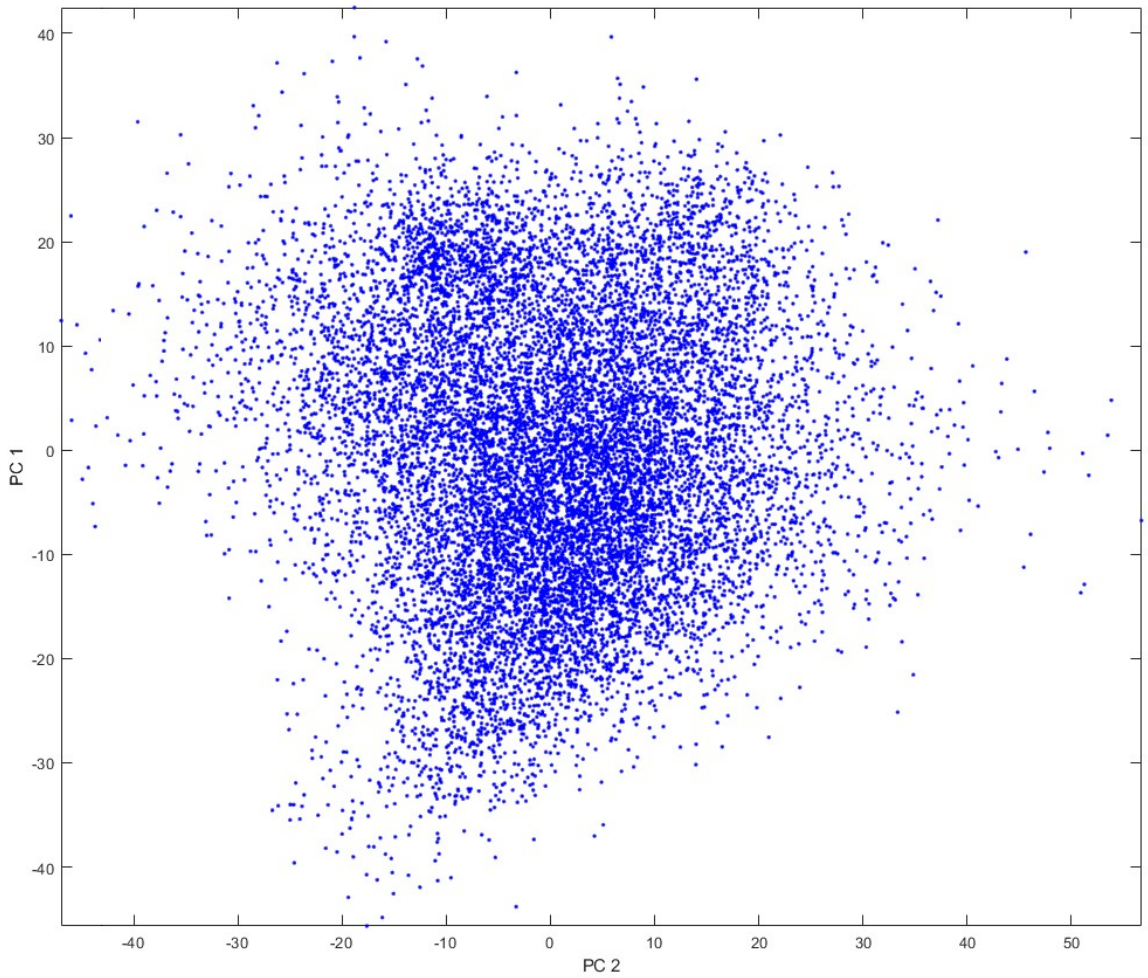
Embeddings, cont.



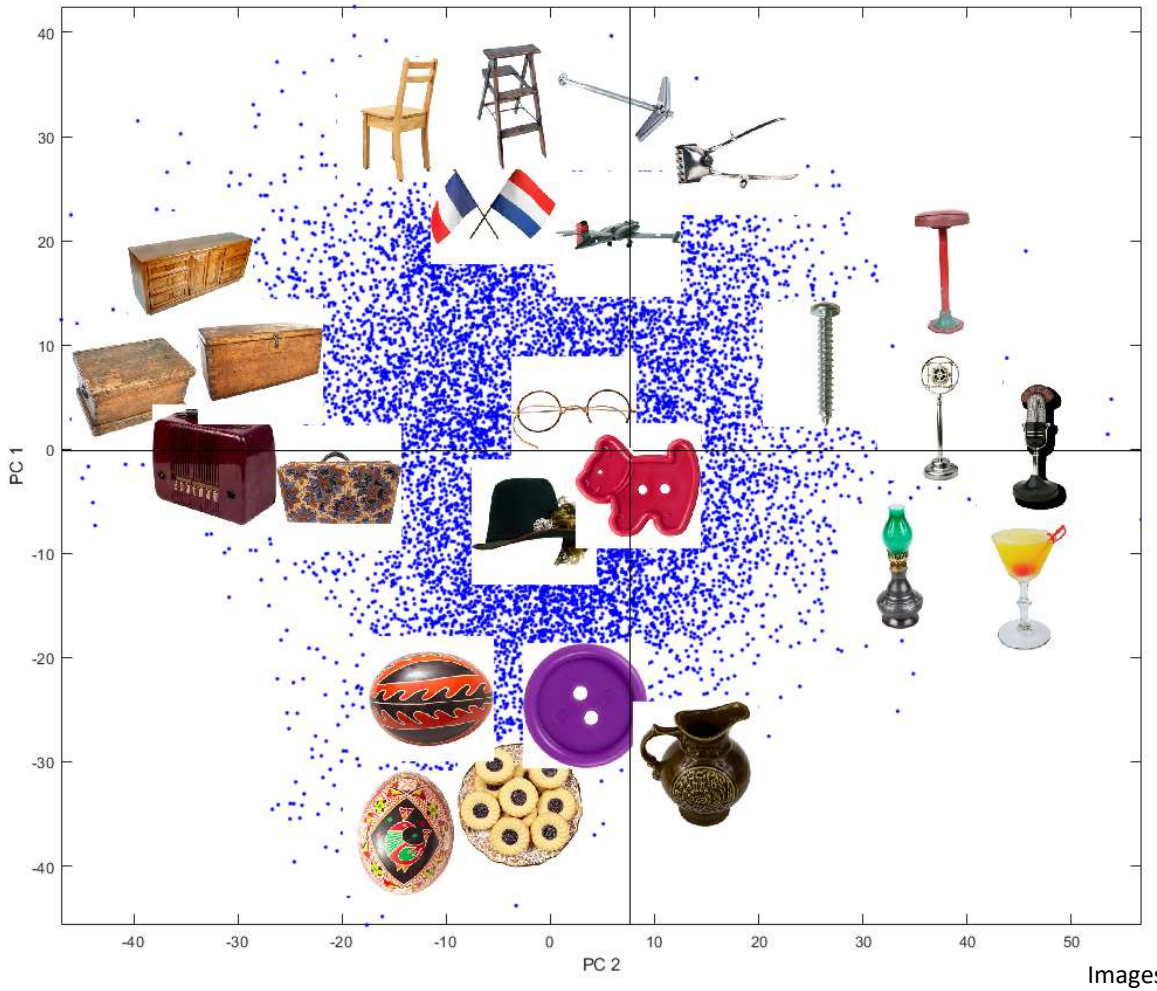
Embedding of an input = the network's response to the input at some layer

Image: Han, Zhong, Cao, & Zhang (2017)

Embeddings, cont.



Embeddings, cont.



Images: Hemera Technologies

Embeddings, cont.

Similar images: layer 7

Images: Hemera Technologies

Outline

- Clustering
- Gaussian mixture model and EM algorithm
- Unsupervised evaluation

Clustering

Clustering basics

- Clustering = unsupervised learning; no explicit or implicit definition of class
- Learn structure from data alone
- But you usually bring your own assumptions about what kind of structure you expect in the data:
  - Exclusive or overlapping clusters?
  - Hierarchical clusters?
  - What defines a good “group”?

Deterministic vs. probabilistic

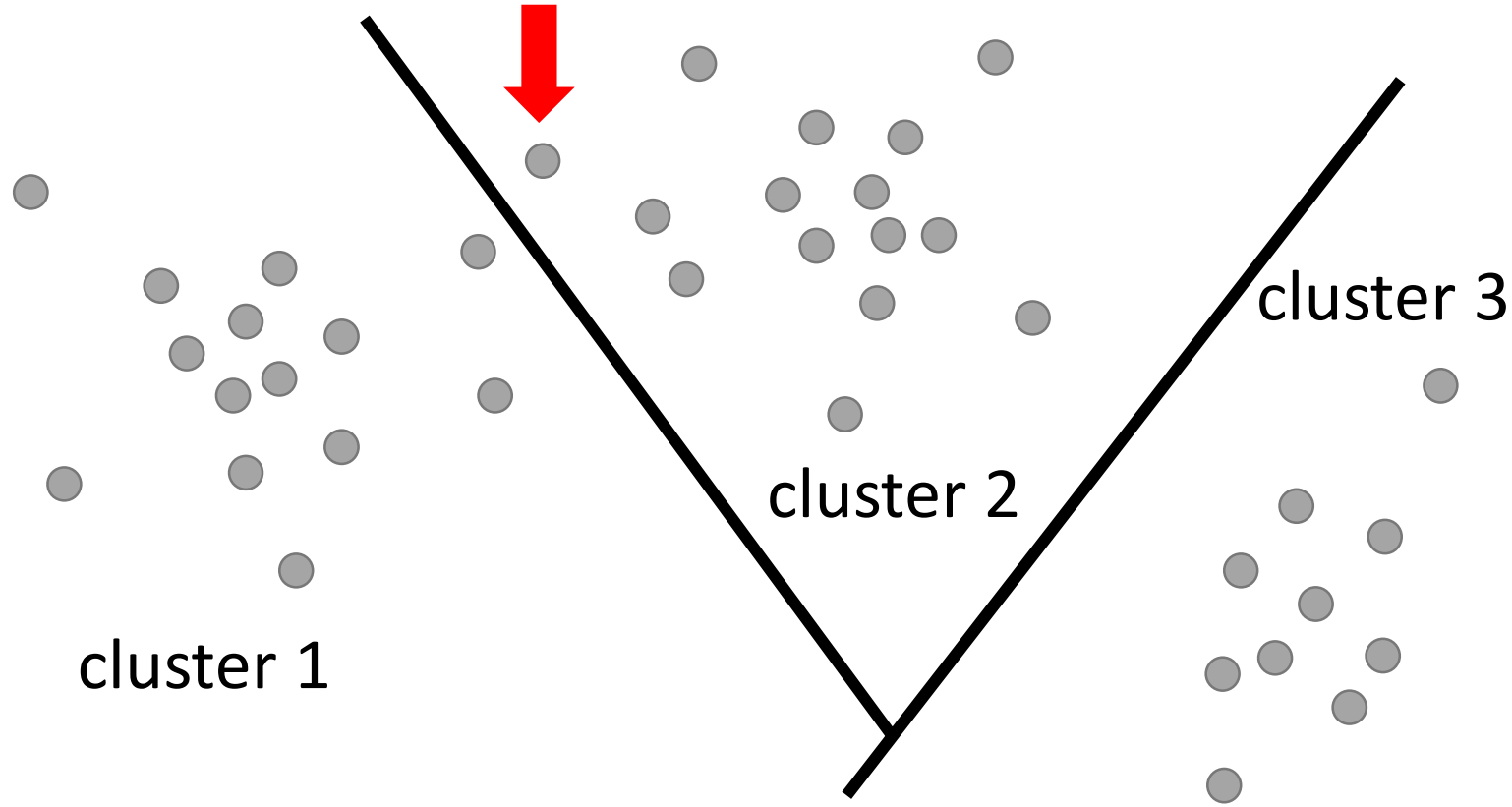
- Deterministic clustering = each instance is a member of one cluster = clusters can't overlap

Instance	Cluster
1	3
2	1
⋮	⋮

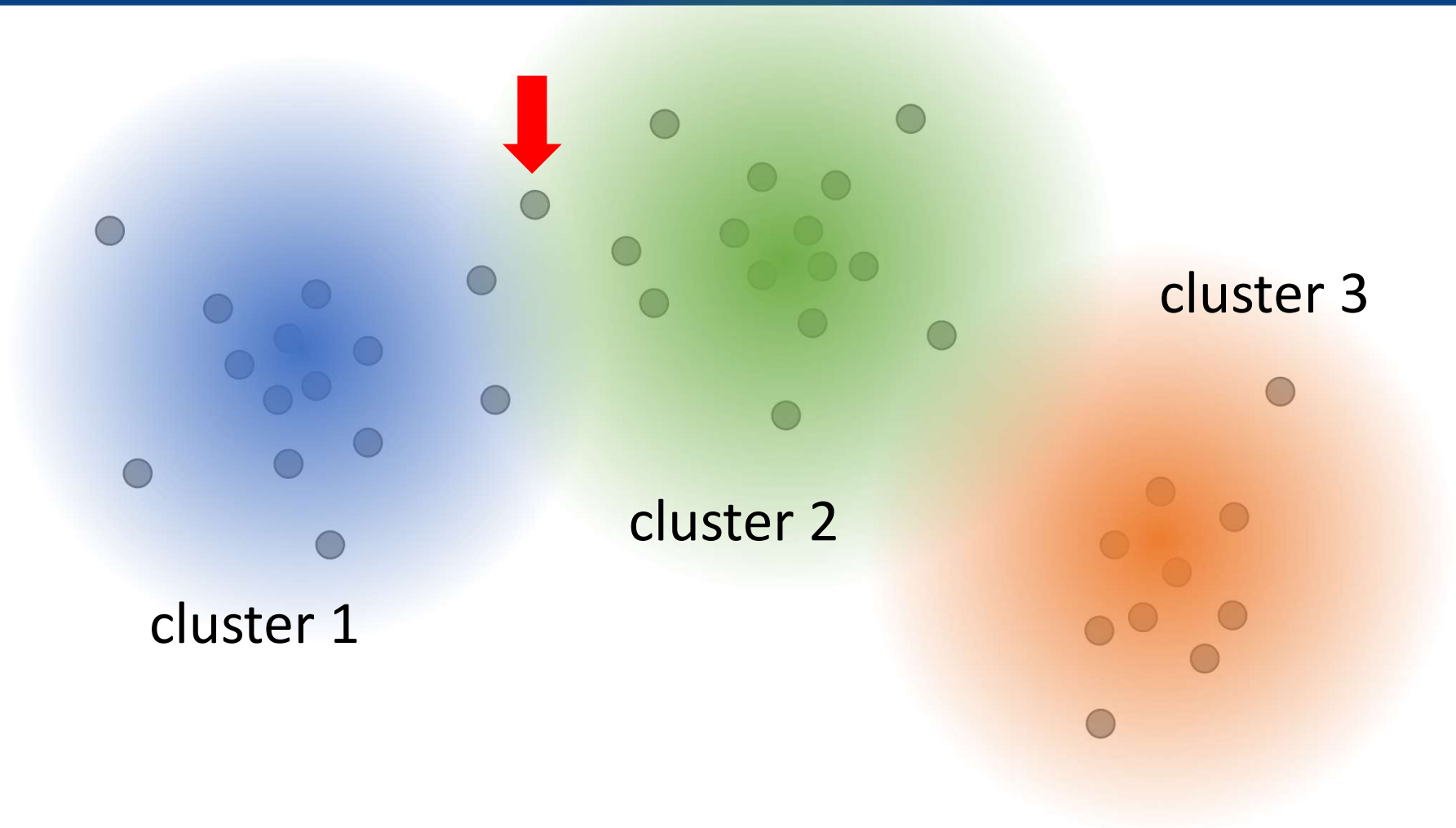
- Probabilistic cluster = each instance has a weight in each class = clusters overlap

Instance	Cluster 1	Cluster 2	Cluster 3
1	0.01	0.87	0.12
2	0.67	0.15	0.18
⋮		⋮	

Deterministic vs. probabilistic



Deterministic vs. probabilistic



Deterministic vs. probabilistic

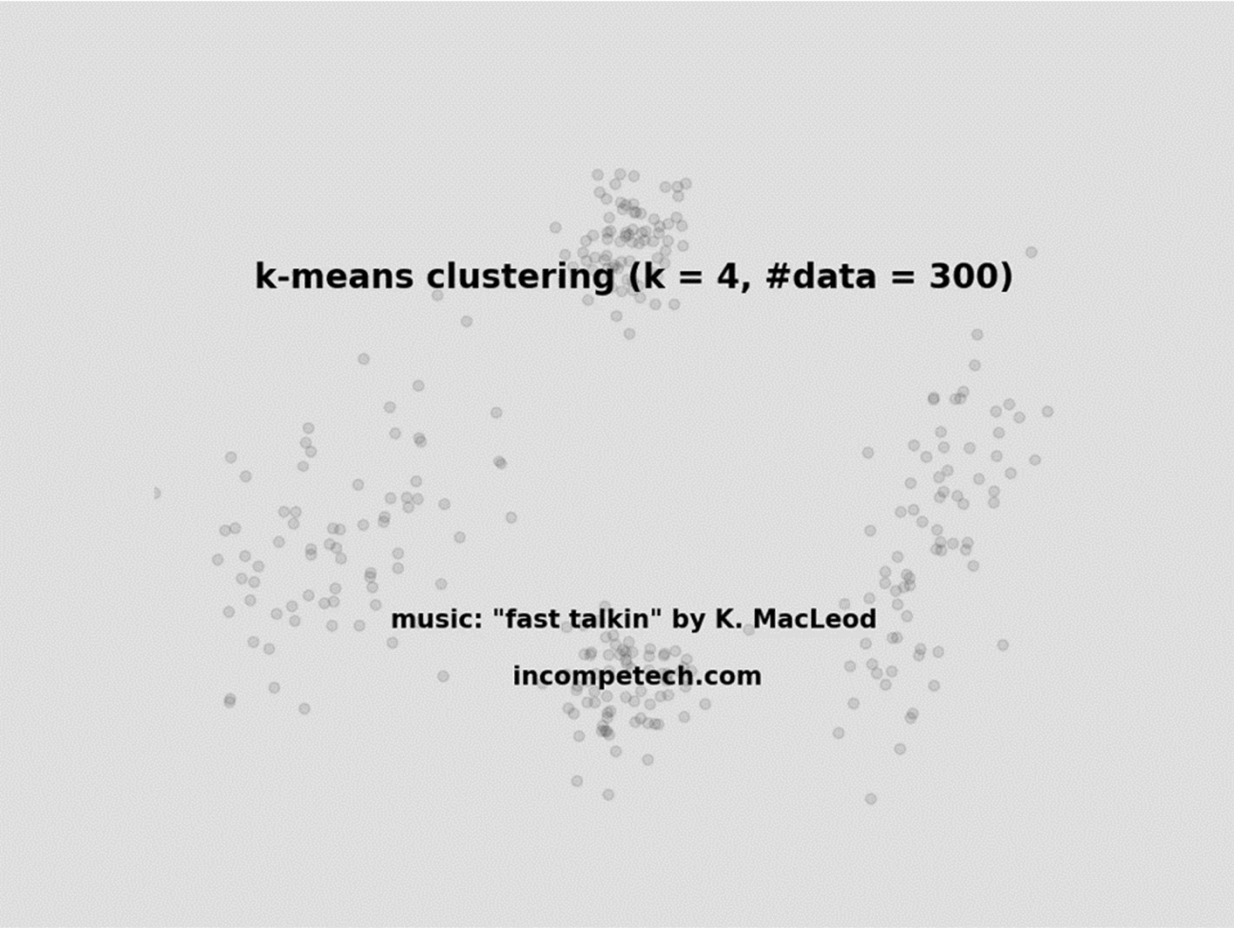
- Why choose one over the other?

K-means clustering

- Given k, the k-means algorithm is implemented as follows (Lloyd's algorithm):

1. Select k points at random as the initial cluster centroids
2. For each instance, compute the distance to each centroid
3. Assign each instance to the cluster with nearest centroid
4. Compute a new centroid for each cluster (centroid = mean of all instances in the cluster)
5. Go to 2, repeat until no instances are reassigned

K-means clustering



https://www.youtube.com/watch?v=5I3Ei69I40s

“Soft” k-means clustering

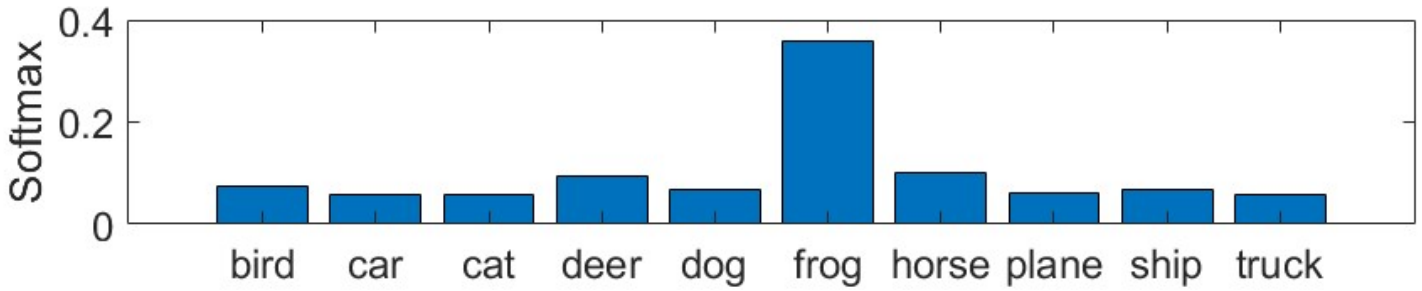
- Is it possible to have a probabilistic (“soft”) version of k-means, where each instance is a member of all clusters, with some probability?
- Solution: use softmax function

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{k} e^{x_i}}$$

- Example: x = [-0.5, 0, 1.3, 2]
- After softmax: σ = [0.048, 0.079, 0.29, 0.58]

Note on softmax function

- Produces a vector that has the properties of a probability distribution:
  - All values in range 0-1
  - Values sum to 1
- Common application: normalise the output of a multiclass classifier:



- Are these values actual probabilities?

“Soft” k-means clustering

- Choose random centroids $\mu_1, \mu_2, \ldots, \mu_k$
- Probability instance x_i is in cluster j is:

$$z_{ij} = \frac{\exp\left[-\beta \left\| x_i - \mu_j \right\|\right]}{\sum_\ell \exp[-\beta \| x_i - \mu_\ell \|]}$$

$\beta > 0$, and is the “stiffness” parameter

- Update each of the centroids with a weighted average of all instances:

$$\mu_j = \frac{\sum_i z_{ij} x_i}{\sum_i z_{ij}}$$

- Result: overlapping clusters

Gaussian mixture model and EM algorithm

Finite mixtures

- A **finite mixture** is a distribution composed of *k* component distributions
- Used to represent subgroups or latent factors in a dataset
- **Gaussian mixture model** (GMM) represents a distribution as composed of *k* Gaussian distributions

Expectation maximization

- **Expectation maximisation (EM)** = parameter estimation method with guaranteed "positive" hill-climbing characteristics relative to the gradient of log-likelihood
- Used to estimate (hidden) parameter values, such as cluster membership
- Family of algorithms rather than a specific algorithm

GMM with EM algorithm

- Basic idea: generalization of (soft) k-means
- **E(xpectation) step:**
  - Based on current estimate of the parameters, calculate the log-likelihood of the instances
- **M(aximization) step:**
  - Compute the parameter distribution that maximizes the log-likelihood
- Repeat until convergence

EM log likelihood

- The log likelihood of a given finite mixture is:

$$L = \sum_i log \sum_j P(c_j) P(x_i | c_j)$$

  where each x_i is an instance and each c_j is a class

- This gives an estimate of the "goodness" of the cluster model; guaranteed to increase on each iteration of the EM algorithm
- Convergence: if the change in log likelihood falls below a predefined value, we consider the estimate to have converged

GMM with EM algorithm

- Assume this data is a mixture of two normal distributions
- Each instance is drawn from one of the two distributions; probability of drawing from distribution 1 is γ and from 2 is $(1-\gamma)$
- The probability density can be written as:

$$g(x) = \gamma\phi_{\mu_1,\sigma_1}(x) + (1-\gamma)\phi_{\mu_2,\sigma_2}(x)$$

  where $\phi_{\mu,\sigma}$ is a normal distribution with mean μ and standard deviation σ
- Problem: estimate the parameters $\gamma, \mu_1, \sigma_1, \mu_2, \sigma_2$

GMM with EM algorithm

- We want to find the parameters that maximize the log likelihood of the instances x_i:

$$\sum_{i=1}^{N} \log\left[\gamma \phi_{\mu_1,\sigma_1}(x_i) + (1-\gamma)\phi_{\mu_2,\sigma_2}(x_i)\right]$$

- This is difficult to solve numerically, but if we know which instances come from which generating distribution, the computation is simpler:

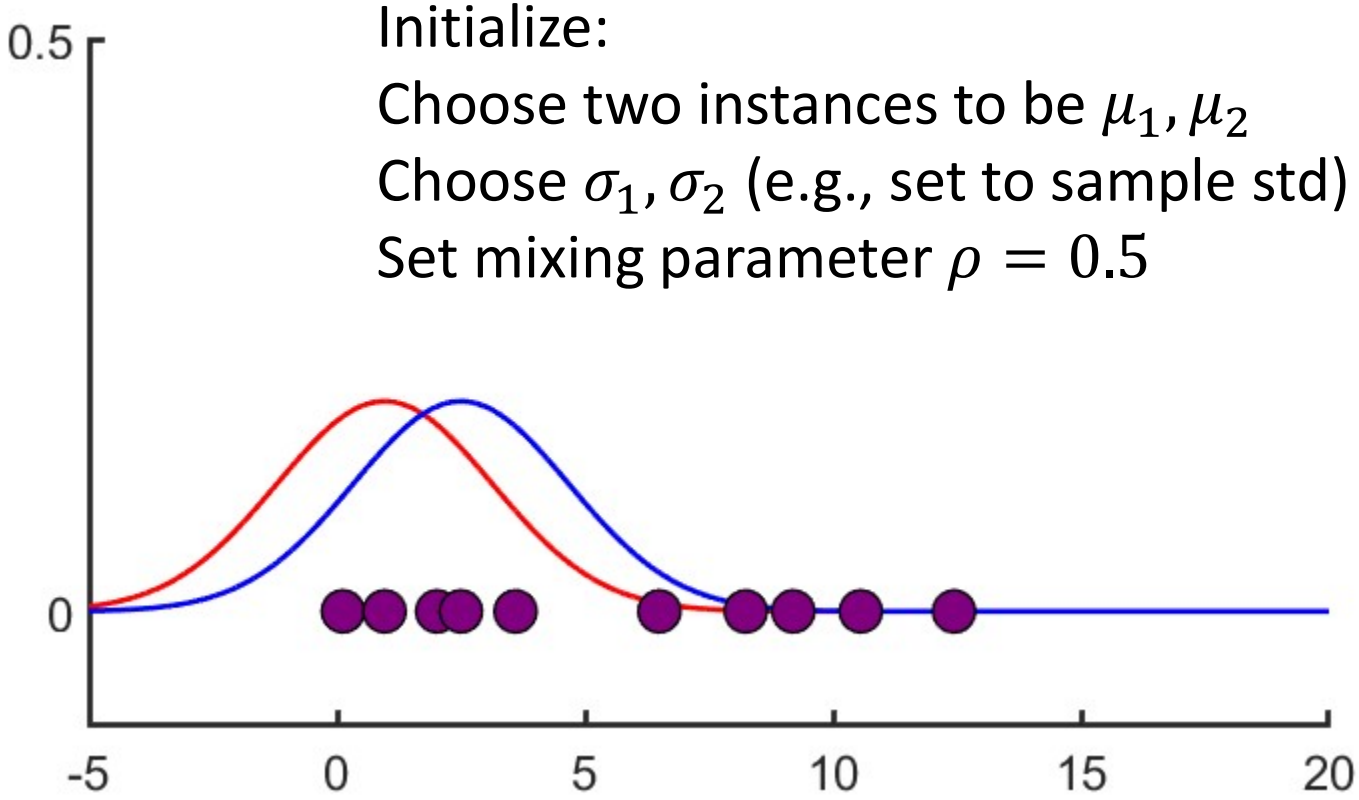
Distribution 1: $\sum_{j \in D_1} \left[\log\left(\phi_{\mu_1,\sigma_1}(x_j)\right) + \log(\gamma)\right]$

Distribution 2: $\sum_{j \in D_2} \left[\log\left(\phi_{\mu_2,\sigma_2}(x_j)\right) + \log(1-\gamma)\right]$

GMM with EM algorithm

- We don't know which distribution produced each instance
- But given some parameters $\gamma, \mu_1, \sigma_1, \mu_2, \sigma_2$ we could estimate how likely each distribution was to have generated each instance
- So, in the iterative EM algorithm:
  - Expectation step: do a soft assignment of instances to each distribution based on current parameters
  - Maximization step: update parameters based on current assignment

GMM with EM algorithm



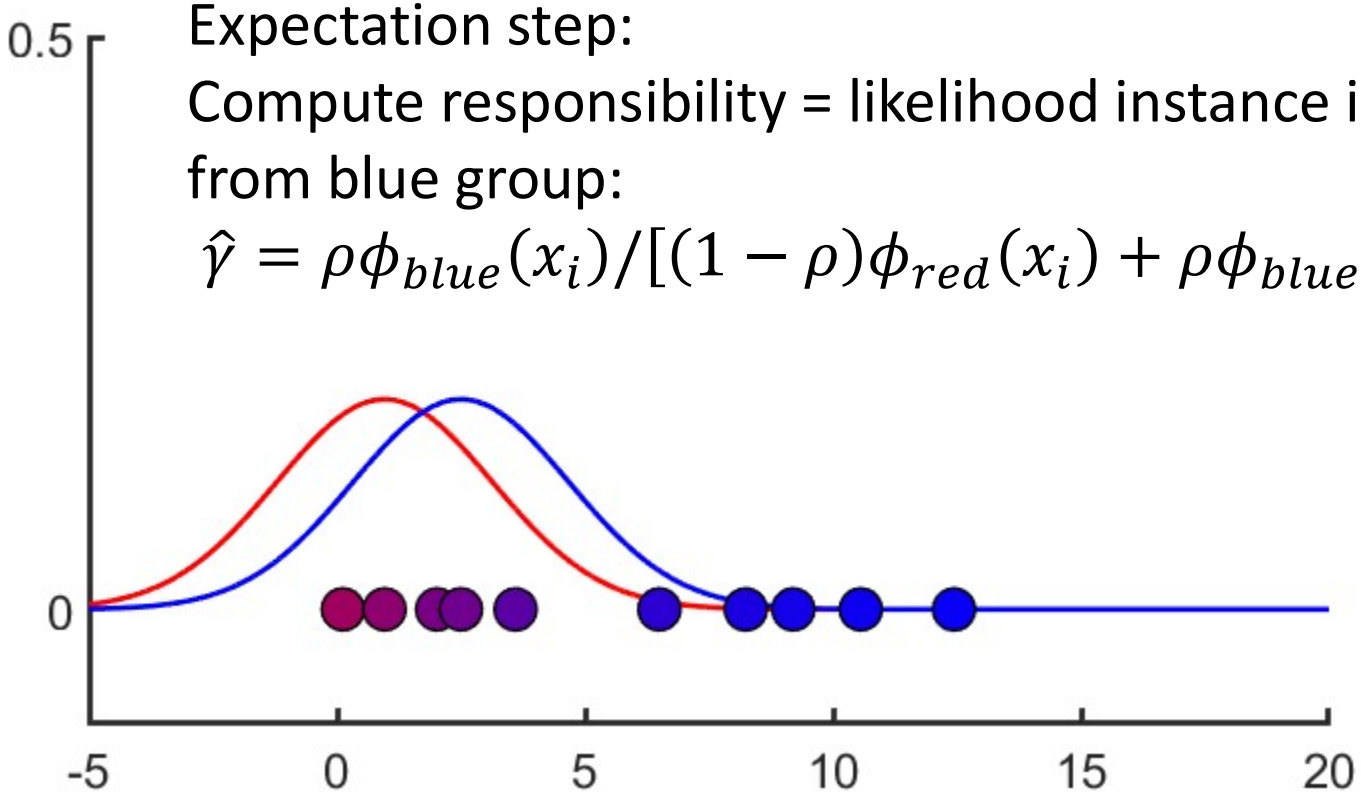
Initialize:

Choose two instances to be μ_1, μ_2

Choose σ_1, σ_2 (e.g., set to sample std)

Set mixing parameter $\rho = 0.5$

GMM with EM algorithm

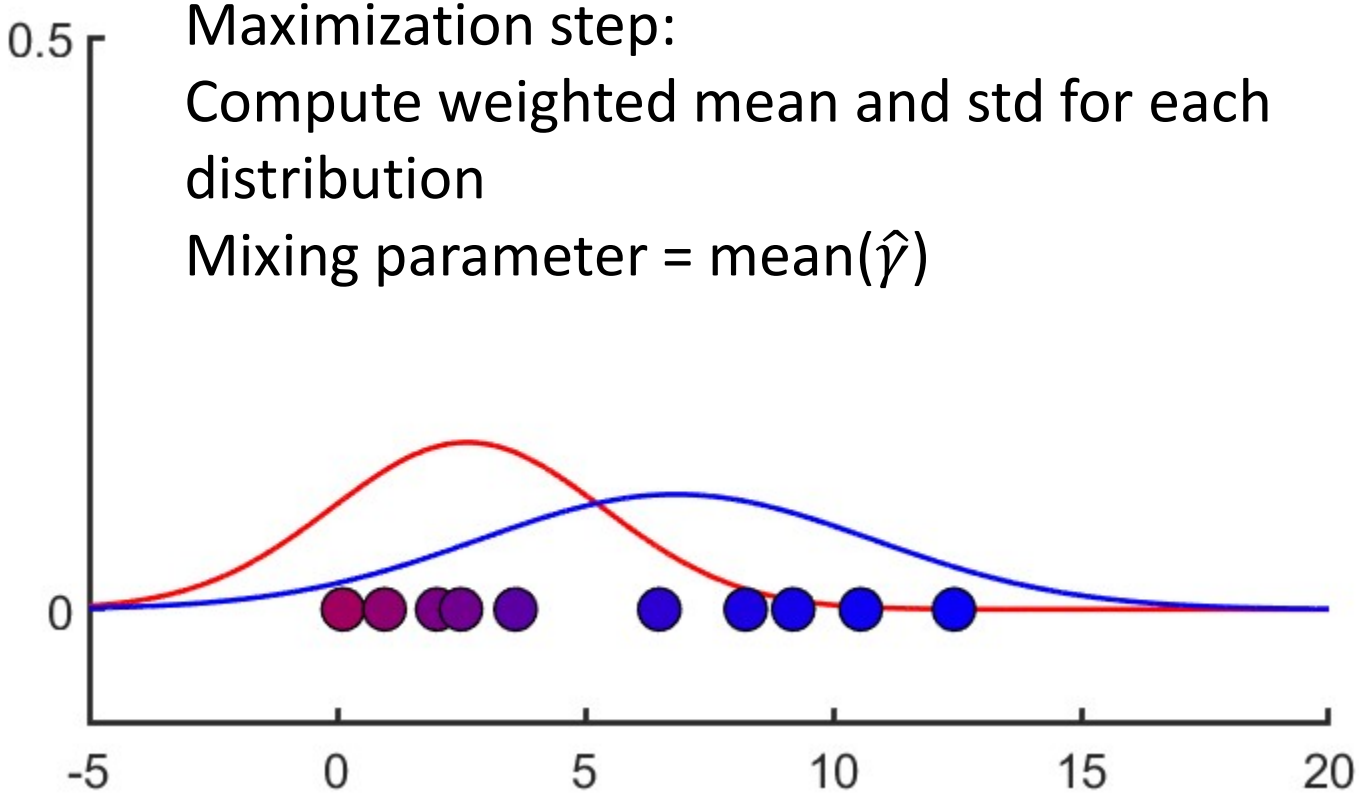


Expectation step:

Compute responsibility = likelihood instance is from blue group:

$$\hat{\gamma} = \rho\phi_{blue}(x_i)/[(1-\rho)\phi_{red}(x_i) + \rho\phi_{blue}(x_i)]$$

GMM with EM algorithm

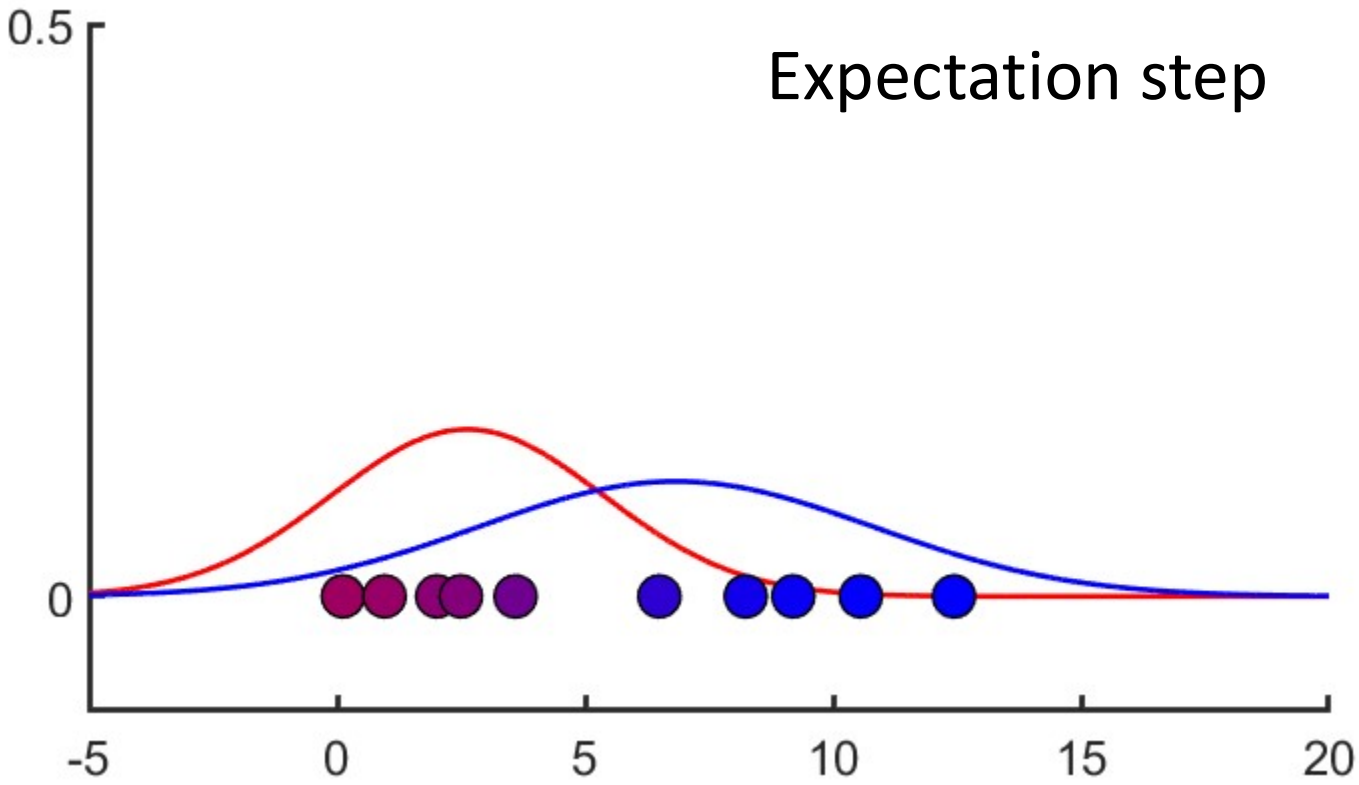


Maximization step:

Compute weighted mean and std for each distribution

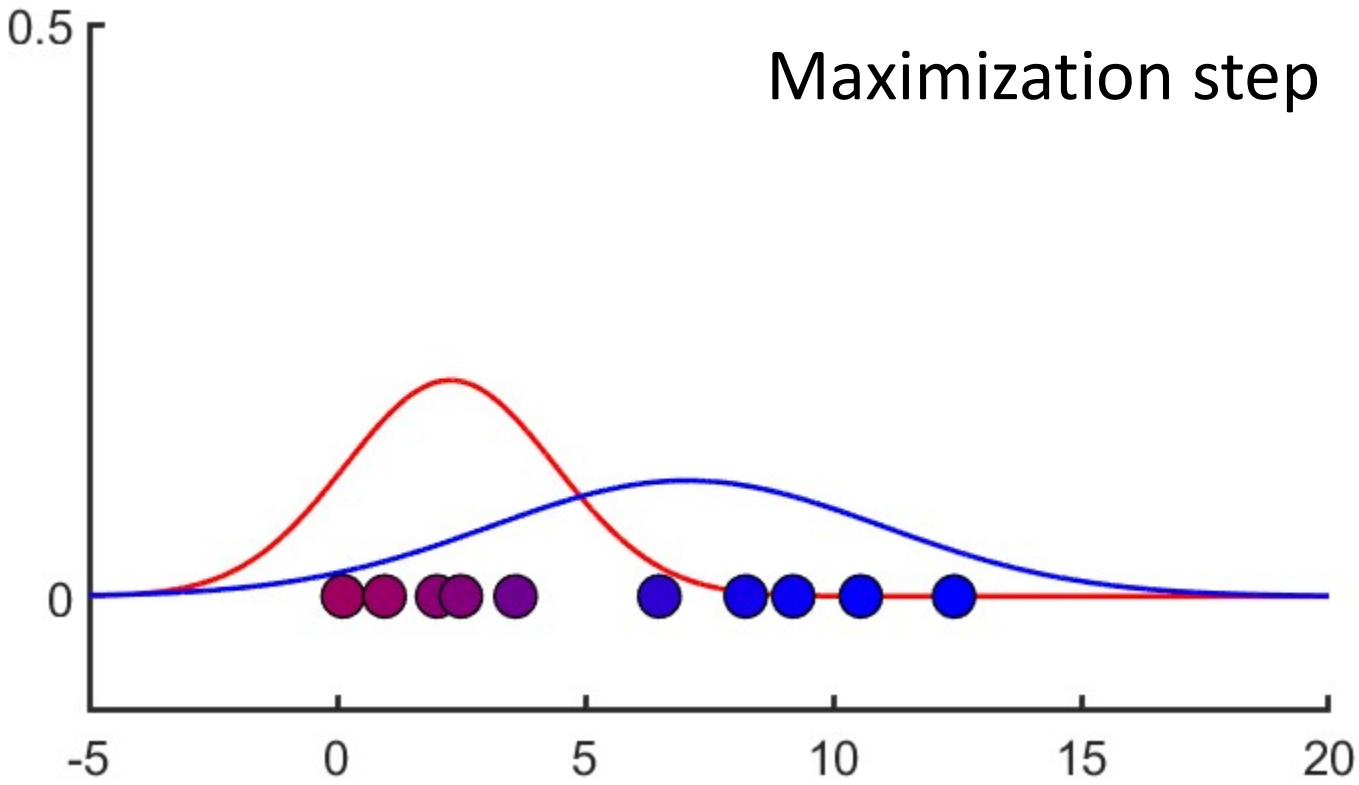
Mixing parameter = mean($\hat{\gamma}$)

GMM with EM algorithm



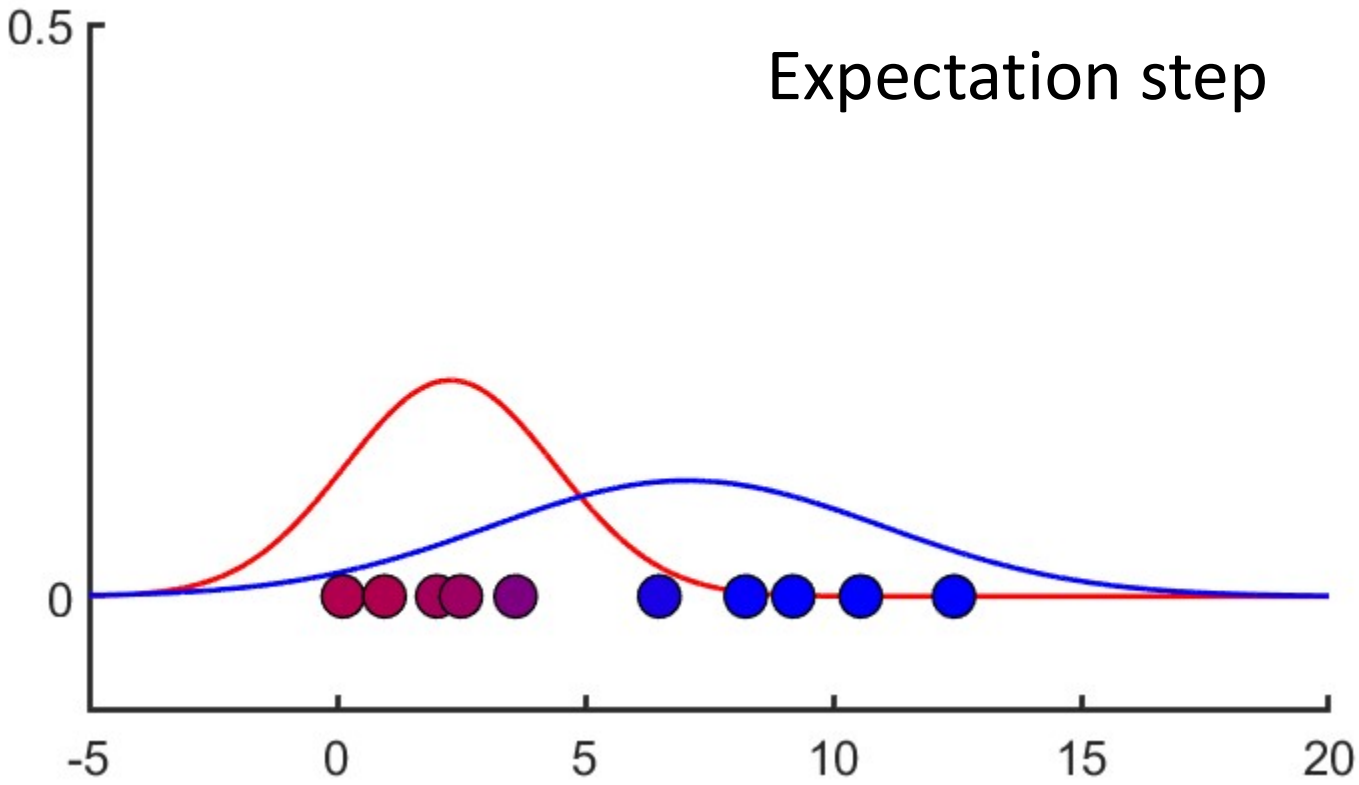
Expectation step

GMM with EM algorithm



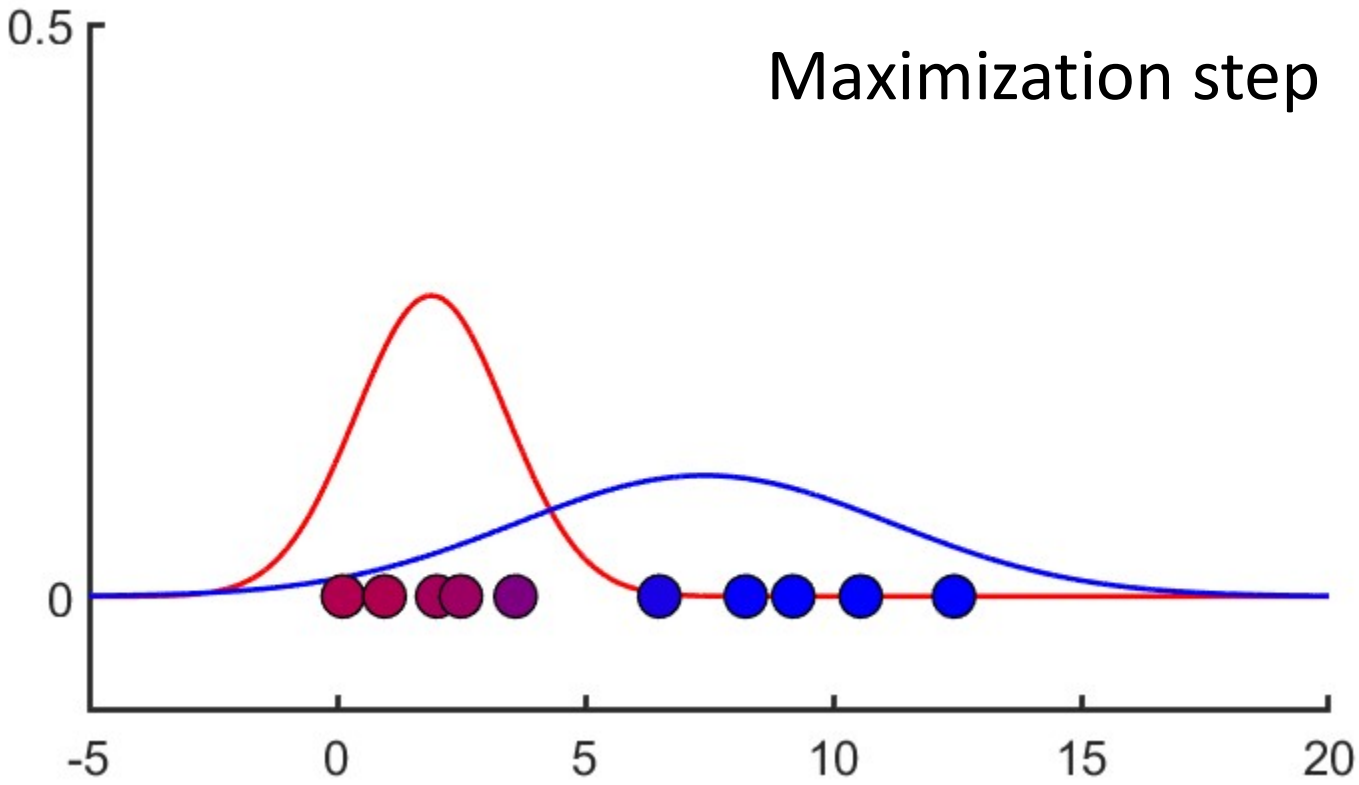
Maximization step

GMM with EM algorithm

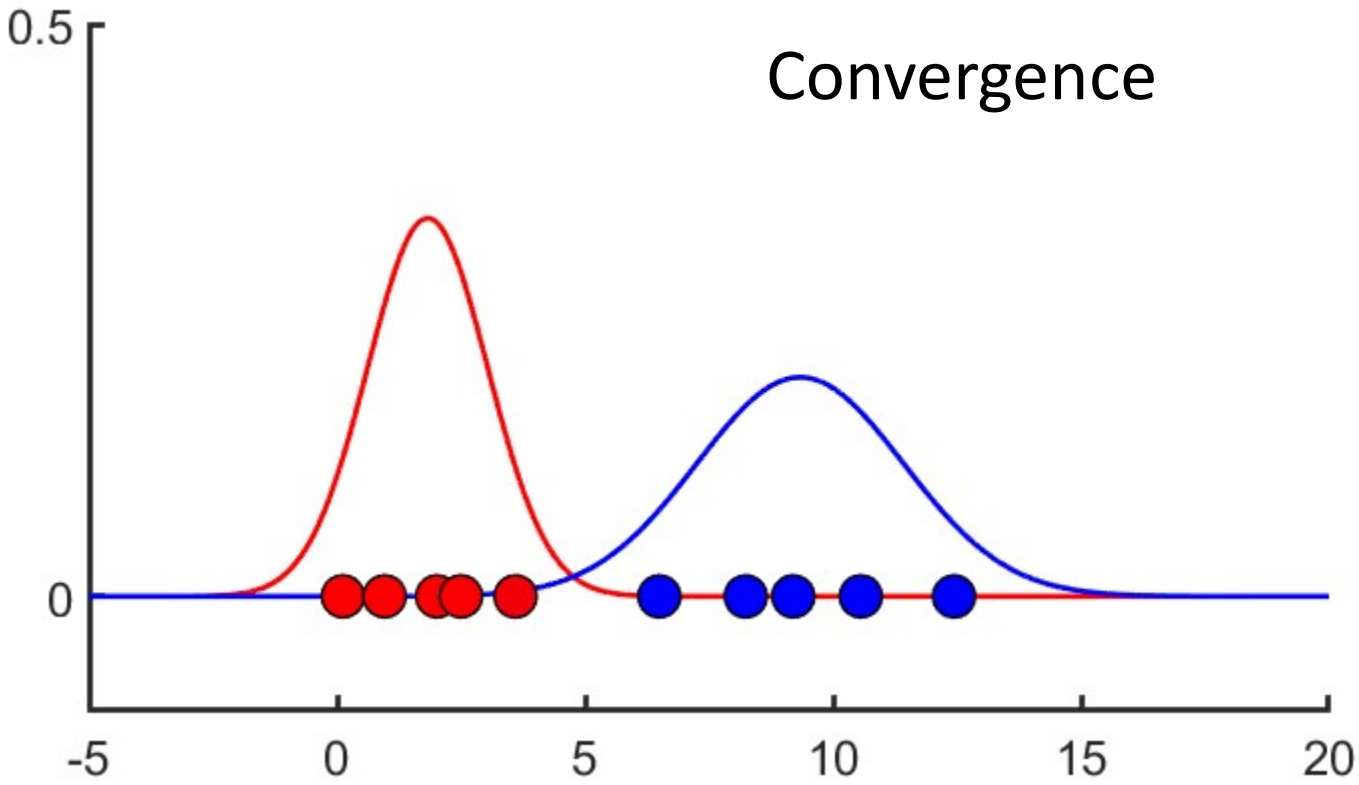


Expectation step

GMM with EM algorithm



GMM with EM algorithm



EM algorithm

- Advantages
  - Guaranteed “positive” hill-climbing behaviour
  - Fast to converge
  - Results in probabilistic cluster assignment
- Disadvantages
  - Has an element of randomness – final model may differ depending on initial parameters
  - Can get stuck in a local maximum; not guaranteed to find the maximum-likelihood solution
  - The number of clusters (k) must be assumed

Unsupervised evaluation

Why do unsupervised learning?

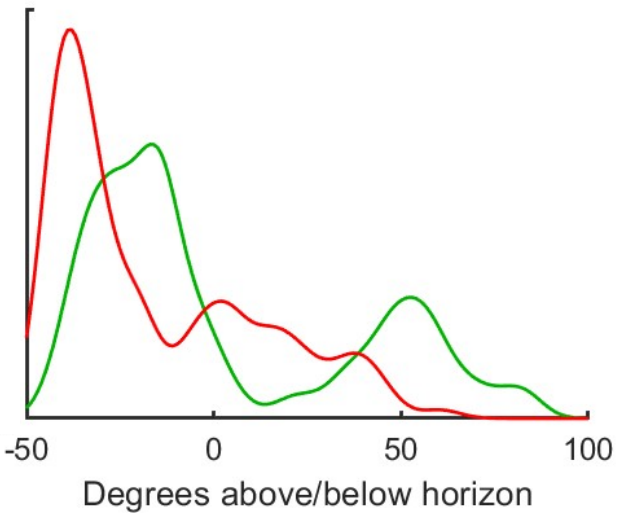
- Map high-dimensional data to a smaller set of clusters or latent factors
- Discover relationships / trends in data
- Model probability density function for probabilistic models (e.g., naïve Bayes)

Example: KDE naïve Bayes

- Naïve Bayes classifier chooses the class that maximizes the posterior probability:

$$\hat{c} = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i | c_j)$$

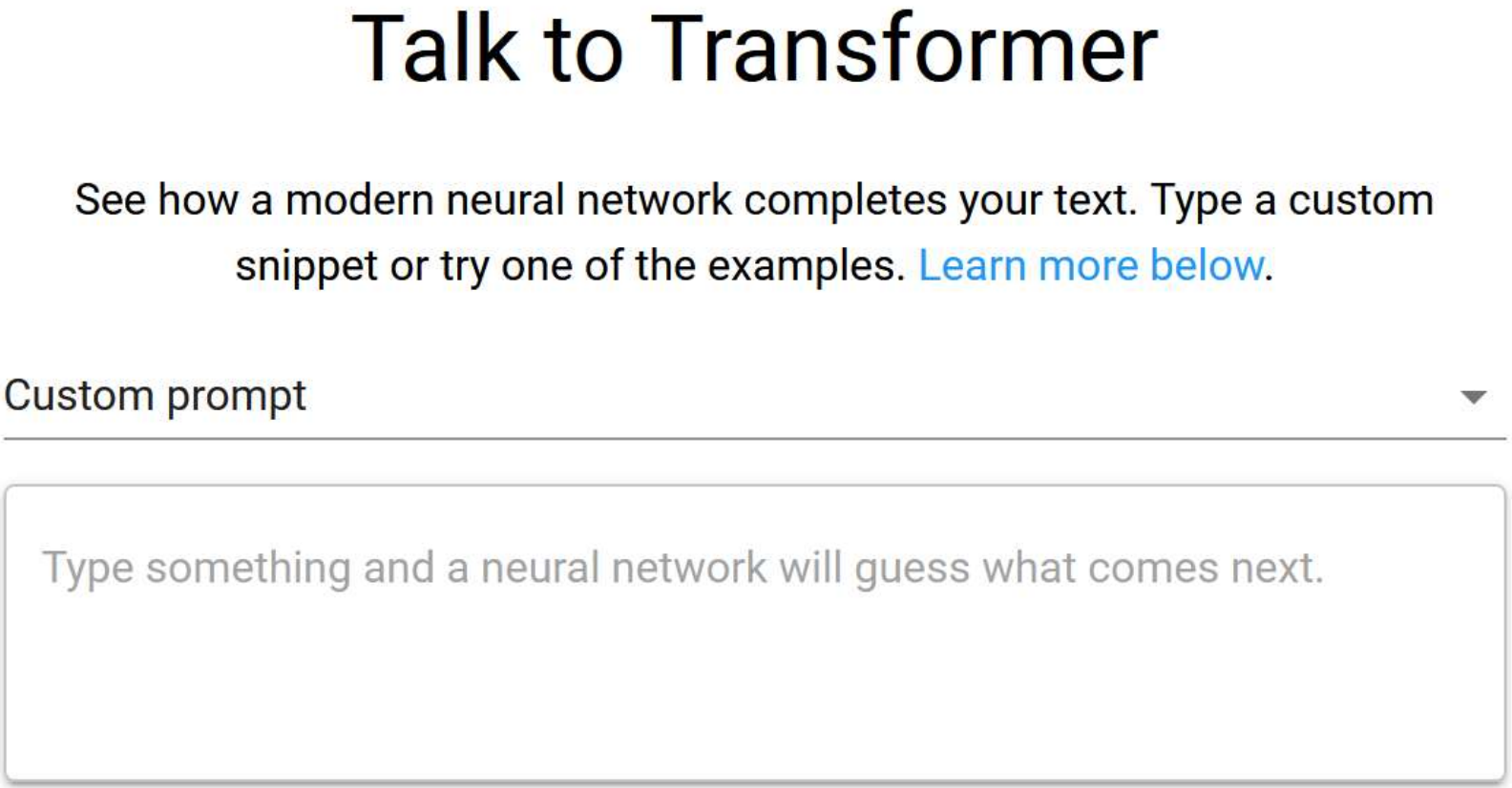
- Likelihood $P(x_i | c_j)$ can be computed by KDE:



Why do unsupervised learning?

- Map high-dimensional data to a smaller set of clusters or latent factors
- Discover relationships / trends in data
- Model probability density function for probabilistic models (e.g., naïve Bayes)
- Generate new samples from the modelled probability distribution

Example: Text generation



https://talktotransformer.com/

Unsupervised learning: Evaluation

- No “ground truth” labels – how to tell if a model is correct, or if one model is better than another?
  - Subjective evaluation
  - Check similarity of clusters over multiple iterations
  - Cross-validation: train on a subset of data and see how well the model predicts held-out data
  - Supervised evaluation: if labels are available, compare how well the discovered clusters match the labels

Unsupervised cluster evaluation

- High cluster cohesion: instances in a given cluster should be closely related to each other

$$cohesion(C_i) = \frac{1}{\sum_{\mathbf{x},\mathbf{y} \in C_i} proximity(\mathbf{x}, \mathbf{y})}$$

- High cluster separation: instances in different clusters should be distinct from each other

$$separation(C_i, C_j) = \sum_{x \in C_i, y \in C_{j \neq i}} proximity(\mathbf{x}, \mathbf{y})$$

Cluster compactness

- One way to evaluate the quality of clusters (especially for k-means) is sum of squared errors (SSE):

$$SSE = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} proximity(\mathbf{x}, \mathbf{c_i})^2$$

  where $\mathbf{c_i}$ is the centroid of cluster C_i

- Proximity measure is often Euclidean for numeric data, or Hamming for nominal

Sum of squared errors: Example

Cluster 1 centroid: sunny	mild	high	no	
sunny	hot	high	no	$1^2 = 1$
sunny	hot	high	yes	$2^2 = 4$
overcast	hot	high	no	$2^2 = 4$
rainy	mild	high	no	$1^2 = 1$
sunny	mild	high	no	0
overcast	mild	high	yes	$2^2 = 4$
rainy	mild	high	yes	$2^2 = 4$

Cluster 2 centroid: overcast	cool	normal	yes
rainy	cool	normal	yes
overcast	cool	normal	yes
sunny	cool	normal	no
overcast	mild	normal	no
sunny	mild	normal	yes
overcast	hot	normal	no
rainy	cool	normal	no

$SSE_1 = 18$

$SSE_2 = 20$

$SSE = SSE_1 + SSE_2 = 38$

Supervised cluster evaluation

- If labels are available, evaluate the degree to which class labels are consistent within a cluster and different across clusters

$$purity = \sum_{i=1}^{k} \frac{|C_i|}{N} \max_j P_i(j)$$

$$entropy = \sum_{i=1}^{k} \frac{|C_i|}{N} H(x_i)$$

where x_i is the distribution of class labels in cluster i

Supervised cluster evaluation

- Example: calculate the entropy and purity of following cluster output

Cluster	Play = yes	Play = no	Entropy	Purity
1	4	0	0	1
2	4	4	1	0.5

$$entropy = -1\log(1) - 0\log(0) = 0$$

$$entropy = -0.5\log(0.5) - 0.5\log(0.5) = 1$$

$$purity = \max(1,0) = 1$$

$$purity = \max(0.5,0.5) = 0.5$$

Supervised cluster evaluation

- Example: calculate the entropy and purity of following cluster output

Cluster	Play = yes	Play = no	Entropy	Purity
1	4	0	0	1
2	4	4	1	0.5
Total:			**0.67**	**0.67**

$$entropy = \left(\frac{4}{12}\right)0 + \left(\frac{8}{12}\right)1 = 0.67$$

$$purity = \left(\frac{4}{12}\right)1 + \left(\frac{8}{12}\right)0.5 = 0.67$$

Supervised cluster evaluation

- Example: calculate the entropy and purity of following cluster output

Cluster	Play = yes	Play = no	Entropy	Purity
1	2	0	0	1
2	6	4	0.97	0.6

$$entropy = -1\log(1) - 0\log(0) = 0$$

$$entropy = -0.6\log(0.6) - 0.4\log(0.4) = 0.97$$

$$purity = \max(1,0) = 1$$

$$purity = \max(0.6,0.4) = 0.6$$

Supervised cluster evaluation

- Example: calculate the entropy and purity of following cluster output

Cluster	Play = yes	Play = no	Entropy	Purity
1	2	0	0	1
2	6	4	0.97	0.6
Total:			**0.81**	**0.67**

$$entropy = \left(\frac{2}{12}\right)0 + \left(\frac{10}{12}\right)0.97 = 0.81$$

$$purity = \left(\frac{2}{12}\right)1 + \left(\frac{10}{12}\right)0.6 = 0.67$$

Summary

- Unsupervised learning = finding structure in unlabelled data
- Methods to fit probability distributions to data:
  - Gaussian mixture
  - Kernel density estimation
- Expectation-maximization (EM) algorithm: family of algorithms to find maximum likelihood parameters for a model