CLUSTERING

ARTIFICIAL INTELLIGENCE | COMP 131

TODAY ON AI

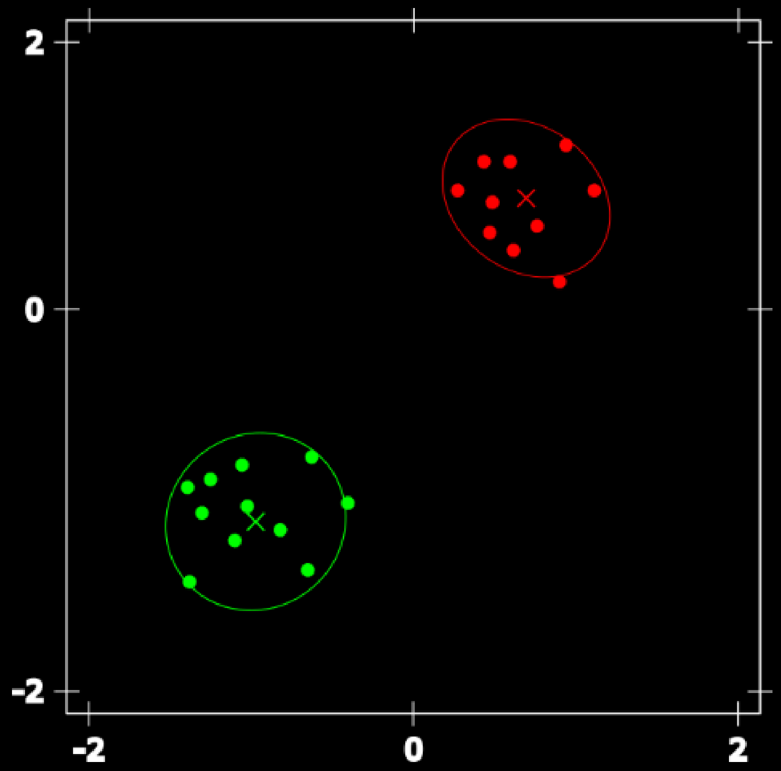
- Clustering
- The EM algorithm
- Hard EM for GMMs
- Soft EM for GMMs
- Questions?

Clustering is the most common of process of grouping a set of examples into classes of similar objects. It is the most common form of **unsupervised learning**.

Clusters are groups of examples with similar features. A good cluster has the following properties:

- The **intra-cluster similarity**, or similarity among members of the **same class**, is high
- The **inter-class similarity,** or similarity among members of **different classes**, is low

- Determine a good representation of clusters
- Establish a notion of similarity / distance between examples
- Determine the number of clusters to classify



Steps required for clustering

When we apply any ML algorithm to data, we aim to represent data density in some **compact** form:

- In many cases we can use a family of **parametric distributions** (i.e., **Gaussian** or **Beta** distributions)
- A single distribution alone tends to be a **poor approximation**
- Linear combinations of distributions, namely **mixture models**, can produce more precise representations:

$$P(x) = \sum_{k=1}^{C} w_k P_k(x) \qquad 0 \le w_k \le 1 \quad \sum_{k=1}^{C} w_k = 1$$

Particularly useful and powerful are **Gaussian Mixture Models**:

$$P(x \mid \theta) = \sum_{k=1}^{C} w_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \qquad 0 \le w_k \le 1 \quad \sum_{k=1}^{C} w_k = 1 \qquad \theta = \{\mu_k, \Sigma_k, w_k \ : \ k = 1 \dots C\}$$

A **distance measure** or **function** is a function $d(x, y) \in \mathbb{R}, x \in \mathbb{R}^n\ y \in \mathbb{R}^n$ that has the following properties:

- **Positive**: $d(x, y) \geq 0$
- **Symmetrical**: $d(x, y) = d(y, x)$
- **Identity**: $d(x, y) = 0 \text{ iff } x = y$
- **Triangle inequality**: $d(x, z) \leq d(x, y) + d(y, z)$

Examples of distances are:

Manhattan distance: $d(X, Y) = |X - Y|$

Euclidean distance: $$\mathrm{d}(X, Y) = \sqrt{\sum_{i=1}^{|X|} (x_i - y_i)^2}$$

Chebyshev distance: $d(X, Y) = \max(|X - Y|)$

Cosine distance: $$d(X, Y) = \frac{X \cdot Y}{\|X\|\ \|Y\|}$$

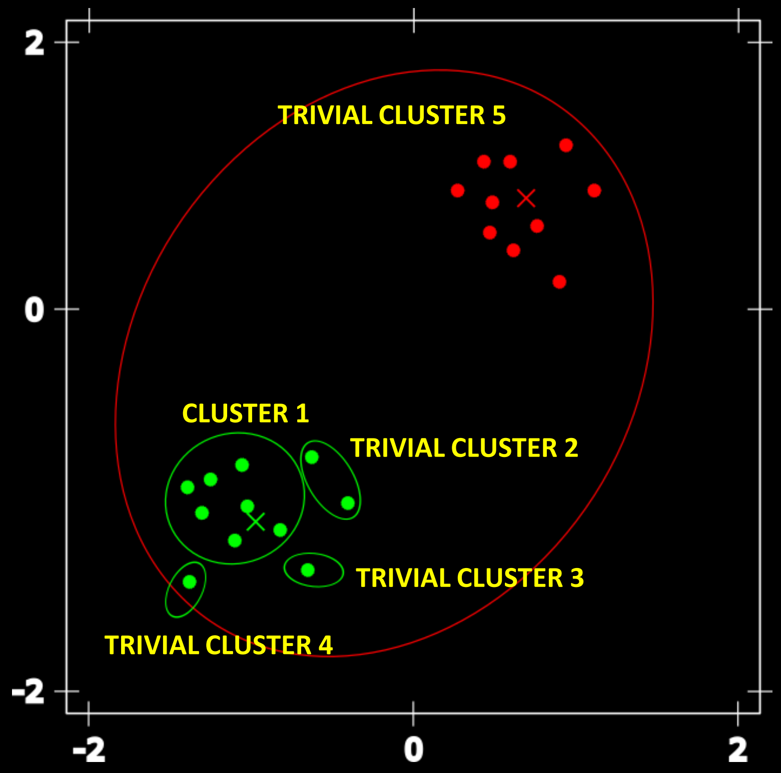
A **similarity measure** or **function** is a function $s(x, y) \in \mathbb{R}$ that measures the similarity between two entities.

There is no single definition of a similarity measure. It can be interpreted as the inverse of **distance function**.

An examples of similarity is:

Normal similarity: $$s(x, \mathrm{M}, \Sigma) = \frac{\mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{N} \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

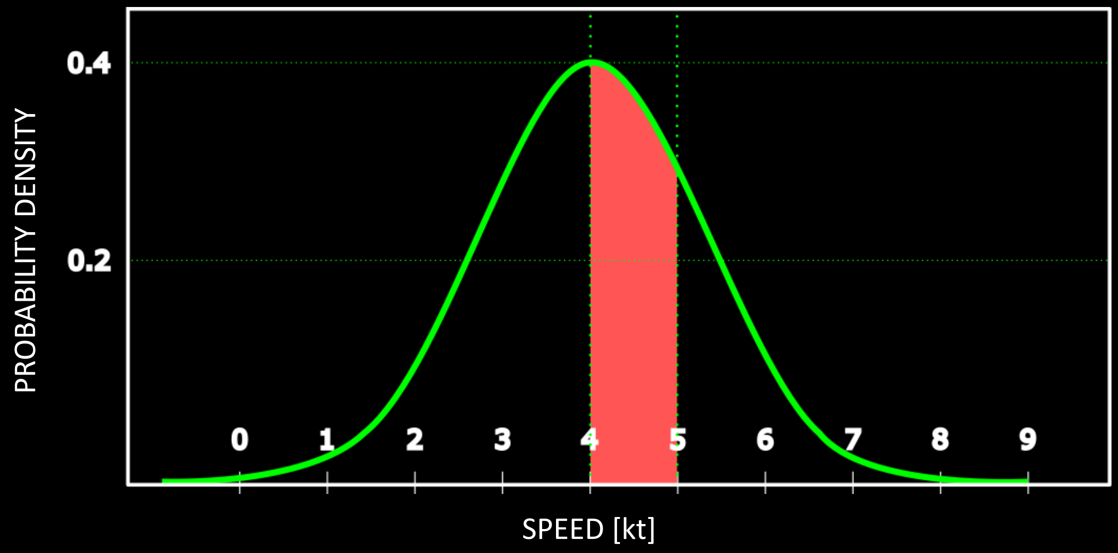
- Determine the number of clusters to classify:
  - Fixed number of clusters:
    - The knowledge about the problem can help
    - Determined by experimentation
  - Completely driven by data
    - The notion of similarity becomes very important
    - It can generate **trivial clusters**, that is clusters that are too large or too small



The **likelihood function** measures the fitness of a statistical model to a sample.

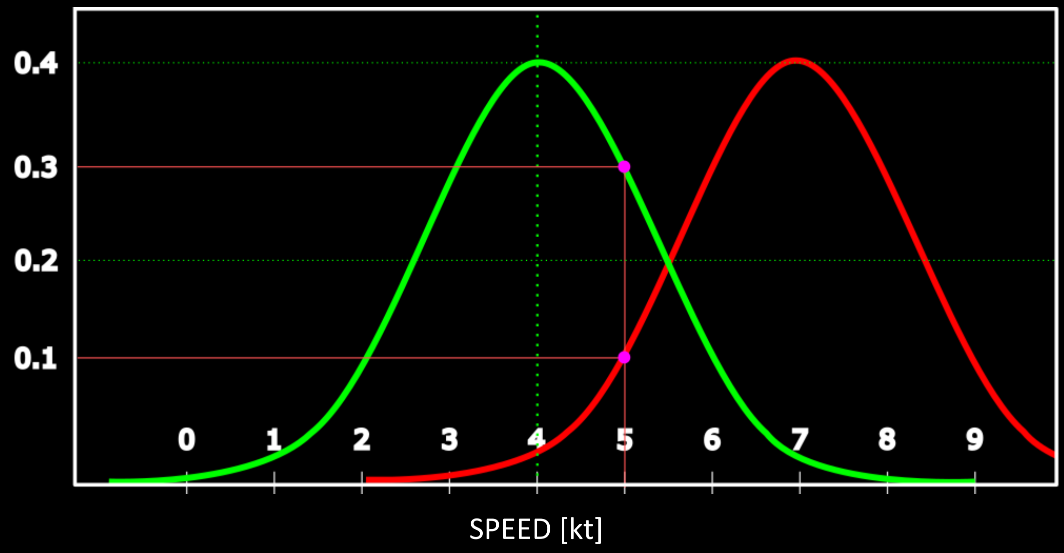
It is very easy to confuse likelihood and probability, and in many cases the two terms are or can be used interchangeably.

PROBABILITY



$P(s \geq 40\text{ kt } and\ s \leq 50\text{ kt} \mid \mathcal{N}(\mu = 40\text{ kt}, \sigma = 1\text{ kt}) = 34\%$

LIKELIHOOD



$P(s = 50\ kt \mid \mathcal{N}(\mu = 40\text{ kt}, \sigma = 1\text{kt}) \cong 30\%$

$P(s = 50\ kt \mid \mathcal{N}(\mu = 70\text{ kt}, \sigma = 1\text{kt}) \cong 10\%$

SECTION 02

The EM algorithm

The **Expectation-Maximization algorithm**, or **EM algorithm**, was proposed in its general form by Dempster. Laird, and Rubin in 1977.

It is a parameter estimation method that belongs to the family of algorithms that perform maximum-likelihood estimation (MLE) over the data.

The algorithm has a wide range of applications in problems with incomplete or hidden data.

Let $O = \{o_1, o_2, \dots, o_n\}$ be a set of n observations called **observed data** (assumed **conditionally independent** given a distribution).

Let $Z = \{z_1, z_2, \dots, z_n\}$ be a set of n values of a hidden or missing data Z, namely **latent variables**.

We define the **incomplete log-likelihood** as:

$$l(\theta; O) = \log P\left(O \mid \theta\right) = \log \prod_i P(o_i \mid \theta)$$

For GMMs:

$$P(o_i|\,\theta) = \sum_{k=1}^{K} w_k \mathcal{N}(o_i\,|\mu_k, \Sigma_k)$$

Incomplete log-likelihood

$$l(\theta; O) = \log \prod_{i=1}^{N} P(o_i\,|\,\theta) = \sum_{i=1}^{N} \log P(o_i\,|\,\theta) = \boxed{\sum_{i=1}^{N} \log \sum_{k=1}^{K} w_k \mathcal{N}(o_i\,|\mu_k, \Sigma_k)}$$

The **EM algorithm** is designed to learn the model parameters θ that would have **maximized** the **incomplete log-likelihood** of the data O.

That is, the MLS of the parameters θ based on the observed data alone.

The algorithm usually starts with **random distributions** or **random parameters** for those distributions.

For each interaction, the EM algorithm consists of two distinct steps:

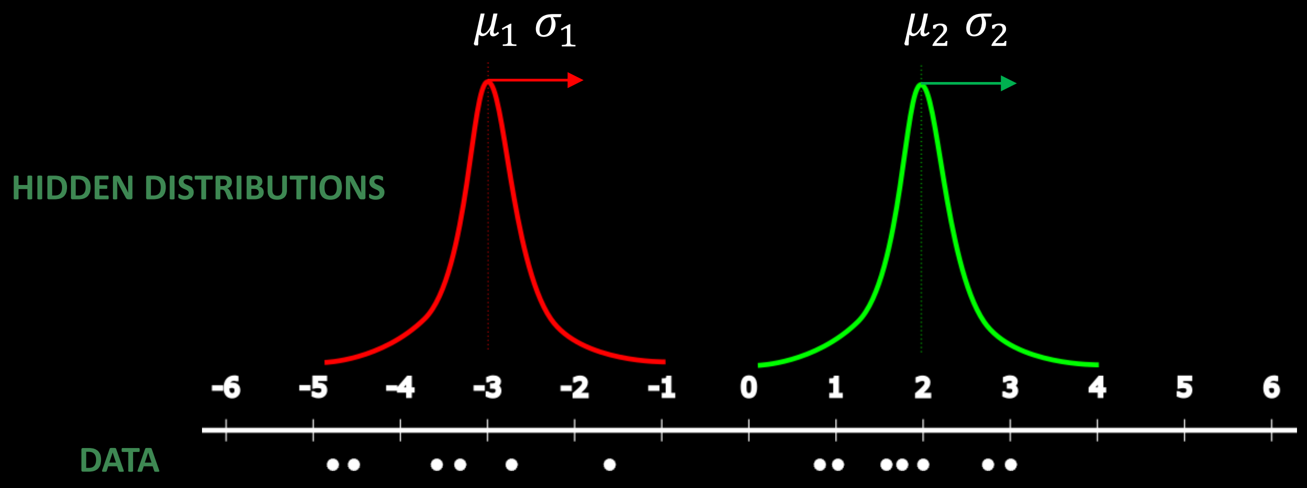
1. **E-step** (**Expectation step**): Compute a **distribution** on the labels of the examples, using the current parameters. That is, find the conditional expected complete-data log likelihood given observed data and θ_t

$$l(\theta; O) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} w_k \mathcal{N}(o_i \mid \mu_k, \Sigma_k)$$

2. **M-step** (**Maximization step**): Update the parameters using current label distributions of the data. That is, determine θ_{t+1} that maximizes $\mathcal{N}(\theta \mid \theta_t)$

$$P(x_i \mid \theta) = \sum_{k=1}^{C} w_k \, \mathcal{N}(x_i \mid \mu_k, \sigma_k)$$

Where $\sum_C w_k = 1$



The parameters in this simplified problem are $\theta = \{w_1, w_2, \mu_1, \sigma_1, \mu_2, \sigma_2\}$

In this example, **if we knew the labels**, and we had w and σ fixed, we could simply estimate μ.

Simplify the problem

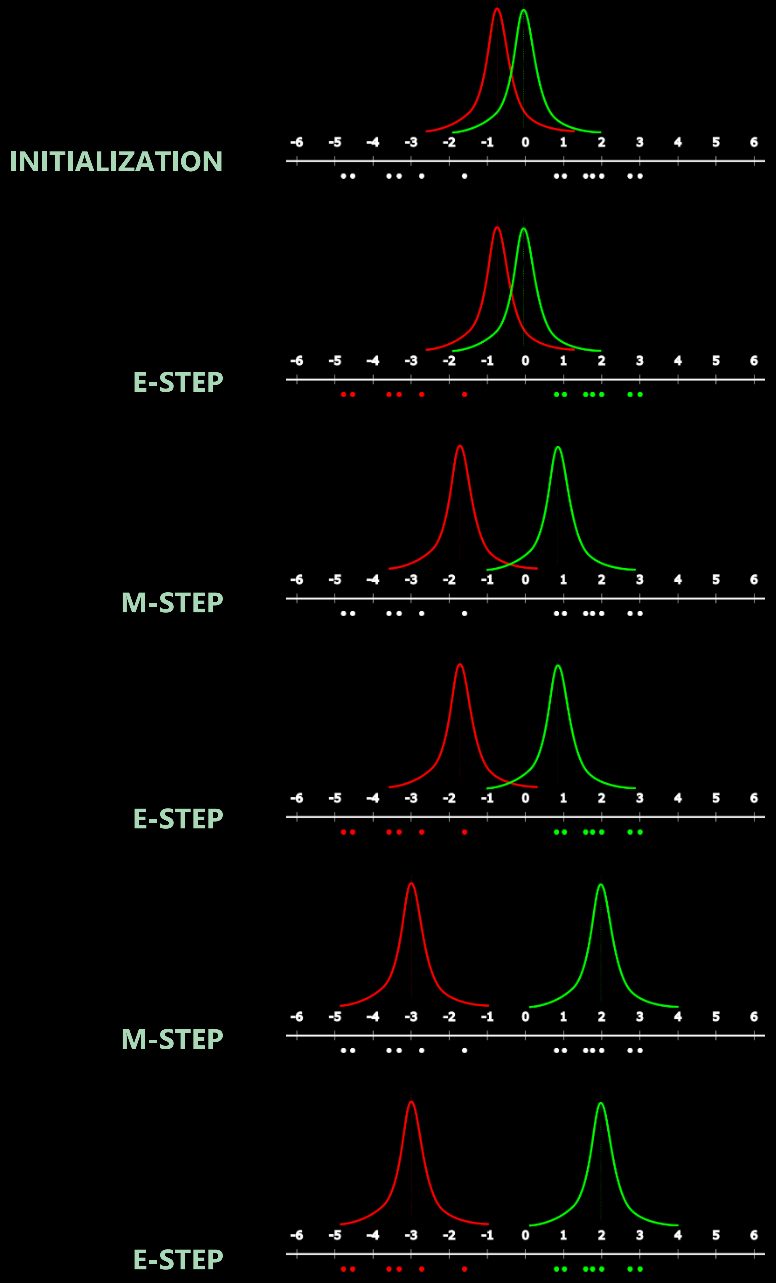
In this example, the EM algorithm estimates the parameters of the model in two recurrent steps:

The algorithm starts with **random distributions**.

1. **E-step**: The data is labeled based on the similarity and the current parameters θ_t (the distributions to guess)

2. **M-step**: New θ_{t+1} parameters are estimated using the current labels

The algorithm can be stopped when nothing changes (either the label assignments or the parameters θ_t).



SECTION 03

Hard EM for GMMs

K-Means is one of the most popular clustering algorithms that searches for a predetermined number of clusters within tan labeled multidimensional dataset:

- It tracks K **centroids** or **center of mass** of the examples in the cluster
- A point is in a particular cluster if it is closer to that cluster's centroid than any other centroid

K-Means finds the best centroids by alternating between the **E-step** and the **M-step** and assigning data points to clusters based on the current closest centroids.

- Random initialization of the model parameters μ and Σ

- While **not converged**:

1. **E-step**: Set the latent variables to the values that maximizes likelihood (closest cluster)

$$z_n(t) = \arg\max_C \left([o_n - \mu_k(t)]^T \Sigma_k^{-1}(t)[o_n - \mu_k(t)]\right) \quad \forall n \in 1 \dots N$$

2. **M-step**: Set the parameters to the values that maximizes likelihood (centroids of the clusters)

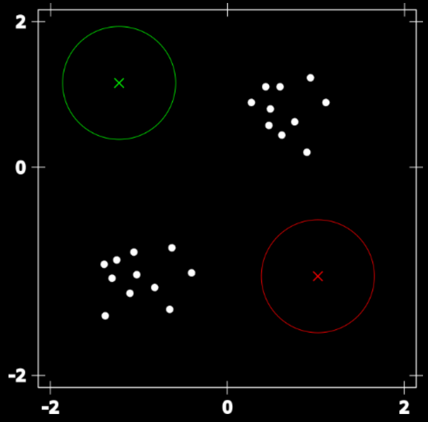
$$w_k(t+1) = \frac{1}{N}\sum_{n=1}^{N} \delta(z_n, k) \quad \forall k \in 1 \dots C$$

$$\delta(z,k) = \begin{cases} 1 & \text{if } z \in C_k \\ 0 & \text{otherwise} \end{cases}$$

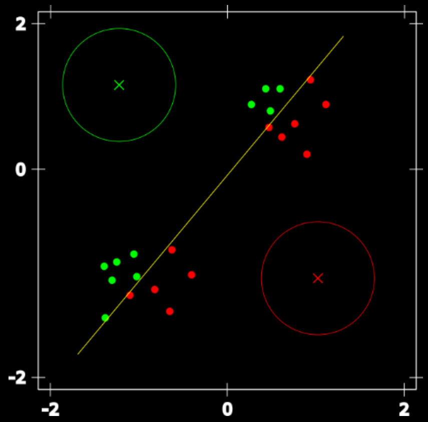
$$\mu_k(t+1) = \frac{\sum_{n=1}^{N} \delta(z_n, k)\, o_n}{\sum_{n=1}^{N} \delta(z_n, k)} \quad \forall k \in 1 \dots C$$

$$\Sigma_k(t+1) = \frac{\sum_{n=1}^{N} \delta(z_n, k)[o_i - \mu_k][o_i - \mu_k]^T}{\sum_{n=1}^{N} \delta(z_n, k)} \quad \forall k \in 1 \dots C$$

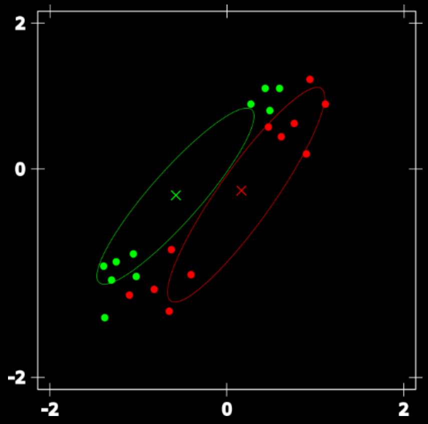
INITIALIZATION



STEP 1

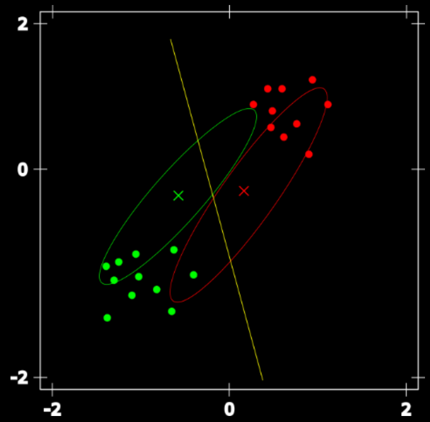


E-STEP

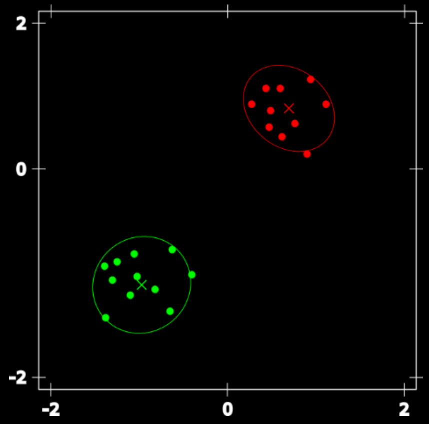


M-STEP

STEP 2



E-STEP



M-STEP

Example of K-means for GMMs

- There are several possible **termination conditions**:
  - A fixed number of iterations
  - Assigned labels do not change
  - Centroid positions do not change

- K-means is a special case of a general EM algorithm which has been proven to converge

GOOD $P_{\theta_{t+1}}(O) \geq P_{\theta_t}(O)$ and it works well in most cases

BAD The number of clusters must be known. It might get stuck in local maxima. Convergence might be slow depending on the initialization. It might overfit the observed data.

Termination condition and convergence

SECTION 04

Soft EM for GMMs

- Clustering typically assumes that each instance is given a **hard** assignment to exactly one cluster
- It does not allow uncertainty in class membership or for an instance to belong to more than one cluster

Soft clustering gives a **probability**, also called **responsibility**, that an example belongs to each cluster:

Each example is assigned a probability distribution across a set of discovered categories (sum of all probabilities must equal to 1).

- Random initialization of the model parameters

- While **not converged**:

1. **E-step**: Weight each observation according to the model's confidence

$$z_{nk}(t+1) = \frac{w_k\,\mathcal{N}(o_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{C} w_j\,\mathcal{N}(o_i \mid \mu_j, \Sigma_j)} \quad \forall n \in 1 \dots N$$

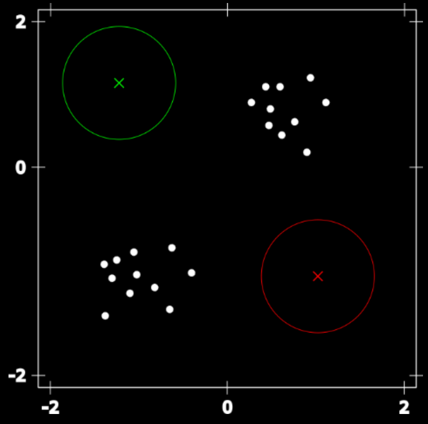
2. **M-step**: Set the parameters to the values that maximizes likelihood

$$w_k(t+1) = \frac{1}{N}\sum_{n=1}^{N} z_{nk}(t) \quad \forall k \in 1 \dots C$$

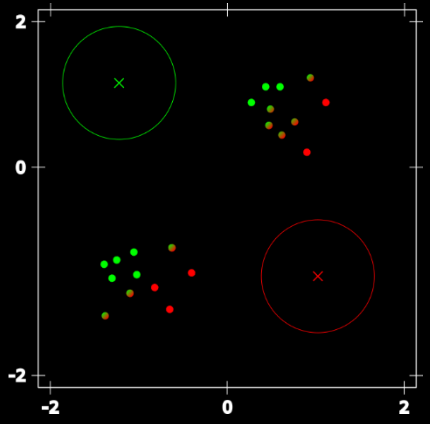
$$\mu_k(t+1) = \frac{\sum_{n=1}^{N} z_{nk}(t)\, o_n}{\sum_{n=1}^{N} z_{nk}(t)} \quad \forall k \in 1 \dots C$$

$$\Sigma_k(t+1) = \frac{\sum_{n=1}^{N} z_{nk}(t)[o_i - \mu_k][o_i - \mu_k]^T}{\sum_{n=1}^{N} z_{nk}(t)} \quad \forall k \in 1 \dots C$$

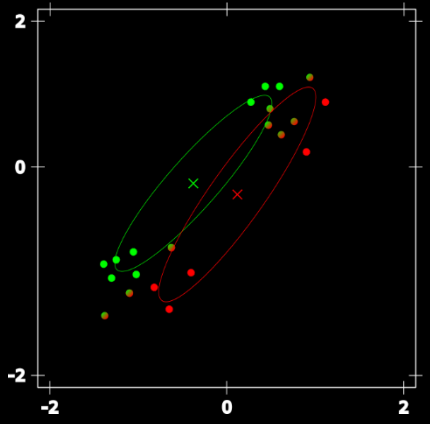
INITIALIZATION



STEP 1

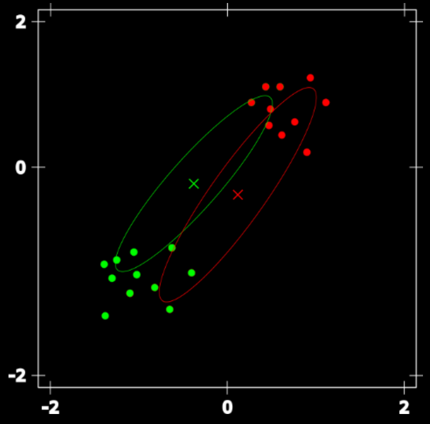


E-STEP

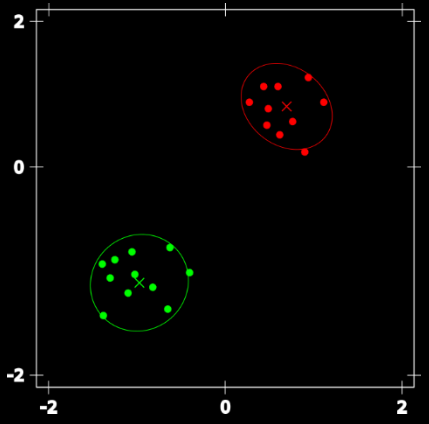


M-STEP

STEP 2



E-STEP



M-STEP

Example of soft EM for GMMs

READINGS

Chapter 20

QUESTIONS ?

ARTIFICIAL INTELLIGENCE

COMP 131

FABRIZIO SANTINI

VERSION 4.0