Unsupervised clustering

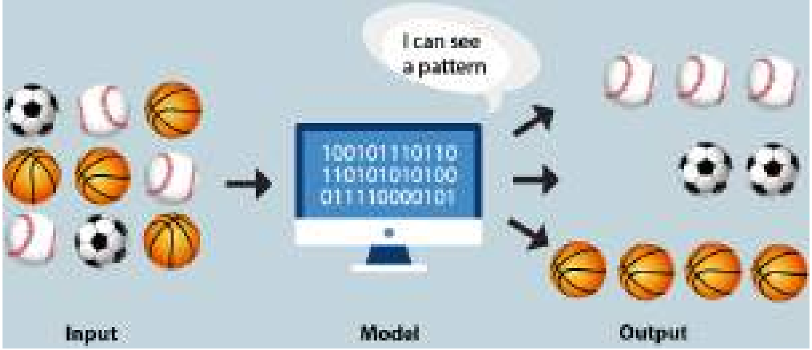
Guillaume Tochon & Joseph Chazalon

LRDE

Why do we care?

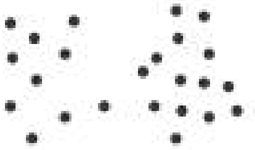
Clustering: Group the input data into clusters that share some characteristics.
(yup, that's a vague definition)

→ Find general patterns in the data (data mining problem)
→ Visualize the data (in a simpler way)
→ Infer some properties of a given data point based on how it relates to other data points (statistical learning)



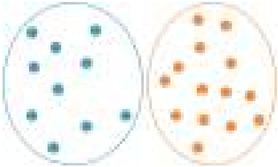
Why is it tricky?

→ Clustering belongs to unsupervised learning ⇒ no ground truth available to learn/evaluate the quality of a clustering algorithm.

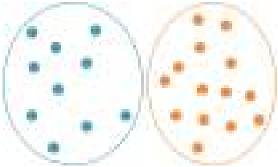


Why is it tricky?

→ Clustering belongs to unsupervised learning ⇒ no ground truth available to learn/evaluate the quality of a clustering algorithm.



Why is it tricky?

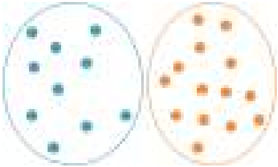


→ Clustering belongs to unsupervised learning ⇒ no ground truth available to learn/evaluate the quality of a clustering algorithm.

→ How to assess how much data points are related to each other?

- ⇒ Which criteria (features) are the more relevant for our problem?
- ⇒ Which metric makes the most sense?

Why is it tricky?



→ Clustering belongs to unsupervised learning ⇒ no ground truth available to learn/evaluate the quality of a clustering algorithm.

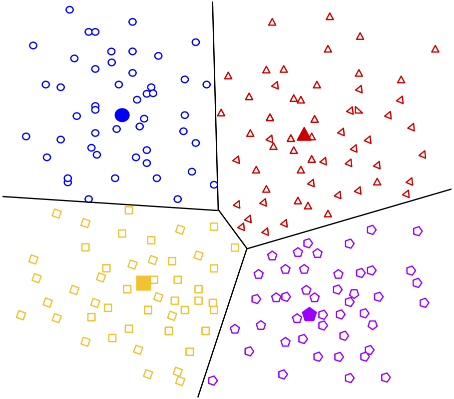
→ How to assess how much data points are related to each other?

- ⇒ Which criteria (features) are the more relevant for our problem?
- ⇒ Which metric makes the most sense?

→ How to assess the soundness of the created clusters? Is that even a relevant question?

One can try to divide existing clustering approaches into several categories:

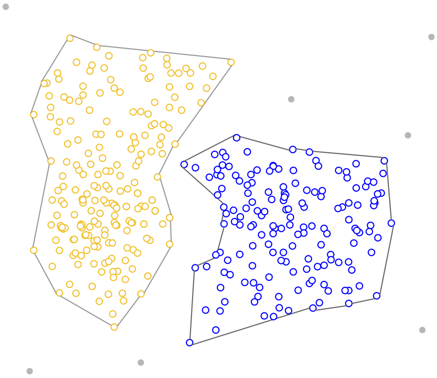
Centroid-based clustering Clusters are summarized using a single representative point, and points are assigned to clusters based on their distance to this *centroid* (k-means and its direct variants).



One can try to divide existing clustering approaches into several categories:

Centroid-based clustering Clusters are summarized using a single representative point, and points are assigned to clusters based on their distance to this *centroid* (k-means and its direct variants).

Density-based clustering Clusters are defined as set of dense points in the feature space (mean shift clustering, DB-SCAN, ...)

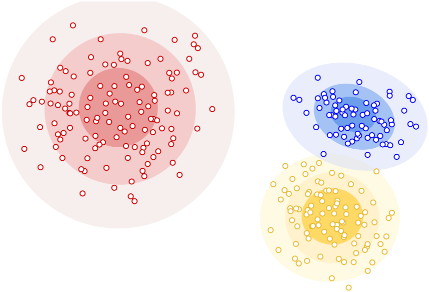


One can try to divide existing clustering approaches into several categories:

Centroid-based clustering Clusters are summarized using a single representative point, and points are assigned to clusters based on their distance to this *centroid* (k-means and its direct variants).

Density-based clustering Clusters are defined as set of dense points in the feature space (mean shift clustering, DB-SCAN, ...)

Distribution-based clustering Clusters are defined based on the likelihood of points to belong to the same probability distribution (Gaussian mixture models, ...)



Families of clustering approaches

Clusteringception

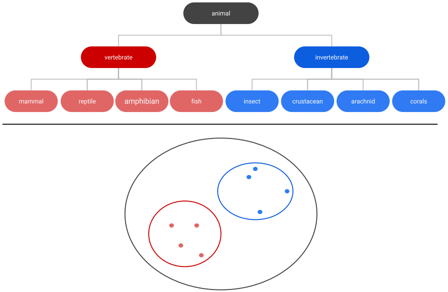
One can try to divide existing clustering approaches into several categories:

Centroid-based clustering Clusters are summarized using a single representative point, and points are assigned to clusters based on their distance to this *centroid* (k-means and its direct variants).

Density-based clustering Clusters are defined as set of dense points in the feature space (mean shift clustering, DB-SCAN, ...)

Distribution-based clustering Clusters are defined based on the likelihood of points to belong to the same probability distribution (Gaussian mixture models, ...)

Hierarchical clustering Clusters are organized in a hierarchical way (Hiearchical Agglomerative Clustering, Recursive K-Means).



One can try to divide existing clustering approaches into several categories:

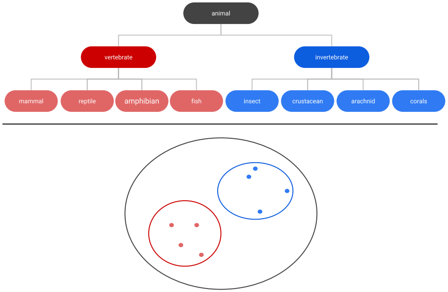
Centroid-based clustering Clusters are summarized using a single representative point, and points are assigned to clusters based on their distance to this *centroid* (k-means and its direct variants).

Density-based clustering Clusters are defined as set of dense points in the feature space (mean shift clustering, DB-SCAN, ...)

Distribution-based clustering Clusters are defined based on the likelihood of points to belong to the same probability distribution (Gaussian mixture models, ...)

Hierarchical clustering Clusters are organized in a hierarchical way (Hiearchical Agglomerative Clustering, Recursive K-Means).

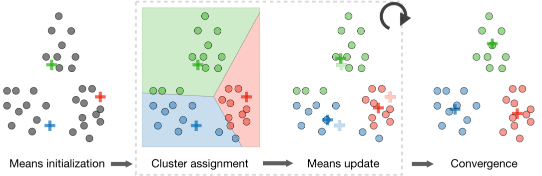
And so on ...



Partition n observations $\mathbf{x}_1, \dots, \mathbf{x}_n$ into k clusters $\mathbf{C} = \{C_1, \dots C_k\}$ where each observation $\mathbf{x}_i$ belongs to the clusters $C_{j^\star}$ whose mean $\boldsymbol{\mu}_{j^\star}$ is the closest: $\mathbf{x}_i \in S_{j^\star}$ with $j^\star = \arg\min_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2$.

Algorithm Basic K-means algorithm.

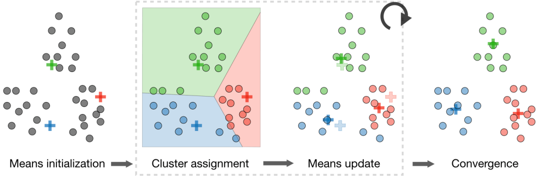
1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



Partition n observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ into k clusters $\mathbf{C} = \{C_1, \ldots C_k\}$ where each observation $\mathbf{x}_i$ belongs to the clusters $C_{j^\star}$ whose mean $\boldsymbol{\mu}_{j^\star}$ is the closest: $\mathbf{x}_i \in S_{j^\star}$ with $j^\star = \arg\min_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2$.

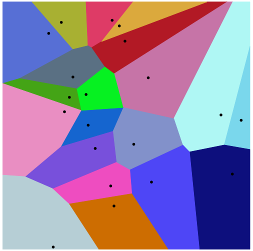
Algorithm Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



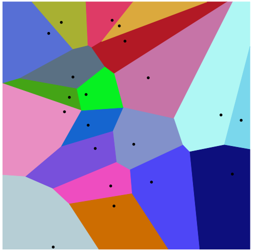
- $\rightarrow$ Minimizes within-cluster sum of squares (variance)
- $\rightarrow$ Overall optimization problem: $$\underset{\mathbf{C}=\{C_1,\ldots C_k\}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}\in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$
- $\rightarrow$ NP-hard problem, no guarantee to find the global optimum
- $\rightarrow$ Stochastic and very sensitive to initial conditions
- $\rightarrow$ Sensitive to outliers (thank you, L_2 norm...)
- $\rightarrow$ Yet it's probably the most used clustering algorithm out there

k-means and Voronoi tesselation



Voronoi tesselation: partition of the Euclidean space relatively to discrete points/seeds. Each region/Voronoi cell is composed of all the points in the space that are closer to the cell seed than to any other seed

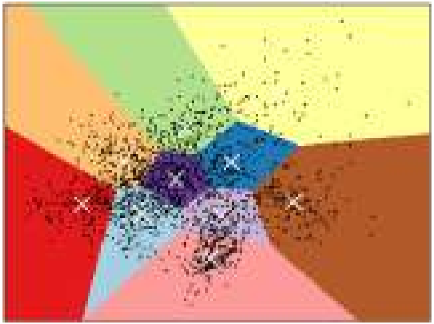
k-means and Voronoi tesselation



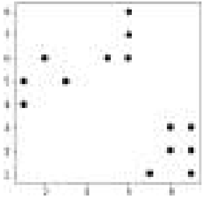
Voronoi tesselation: partition of the Euclidean space relatively to discrete points/seeds. Each region/Voronoi cell is composed of all the points in the space that are closer to the cell seed than to any other seed

→ k-means provides a way to obtain a Voronoi tesselation of the input space, where seeds are the final cluster means.

→ Alternatively, one can use some pre-computed Voronoi tesselation seeds as initial clusters for k-means



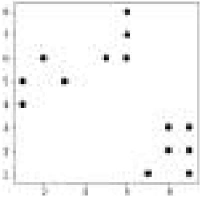
How many clusters for this data set?

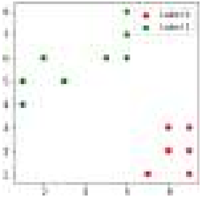


Determining the optimal number of clusters with the elbow method

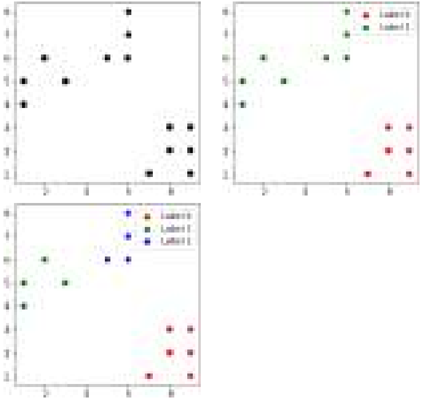
Empirical af, but still better than nothing...

How many clusters for this data set?

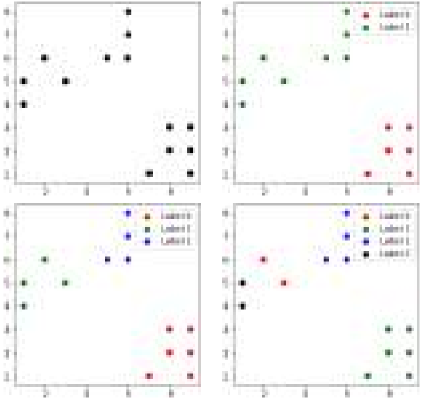




How many clusters for this data set?



How many clusters for this data set?



Determining the optimal number of clusters with the elbow method

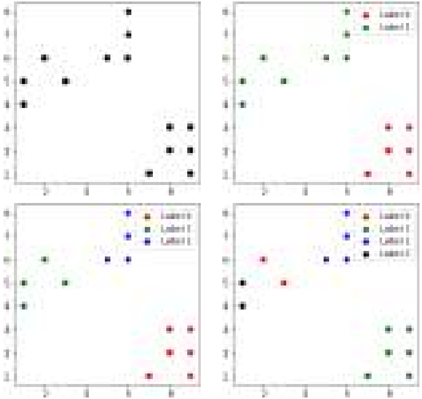
Empirical af, but still better than nothing. . .

How many clusters for this data set?

→ Compute explained variance for an increasing number of clusters k

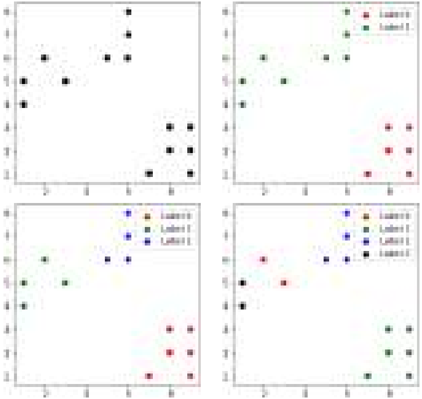
$$\mathrm{Var}(\mathbf{C} = \{C_1, \dots C_k\}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

→ Plot and find the bend of the elbow



Determining the optimal number of clusters with the elbow method

Empirical af, but still better than nothing...

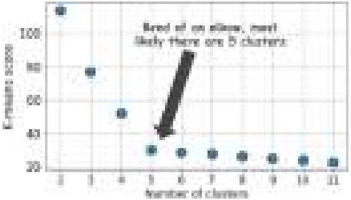


How many clusters for this data set?

- $\rightarrow$ Compute explained variance for an increasing number of clusters k

$$\mathrm{Var}(\mathbf{C} = \{C_1, \dots C_k\}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

- $\rightarrow$ Plot and find the bend of the elbow



Determining the optimal number of clusters with the elbow method

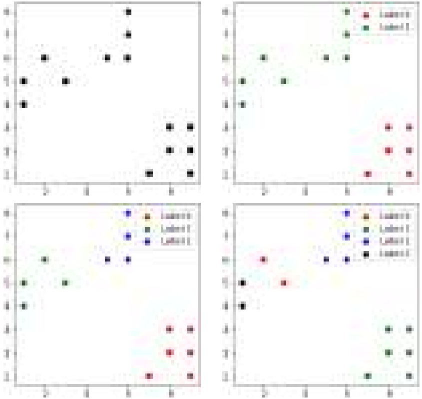
Empirical af, but still better than nothing. . .

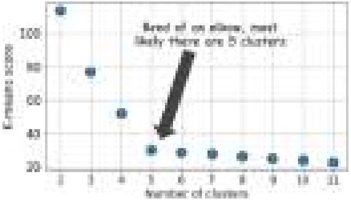
How many clusters for this data set?

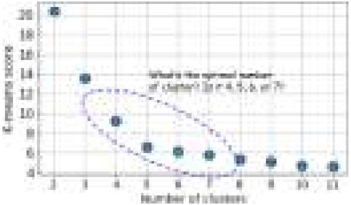
→ Compute explained variance for an increasing number of clusters k

$$\mathrm{Var}(\mathbf{C} = \{C_1, \dots C_k\}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

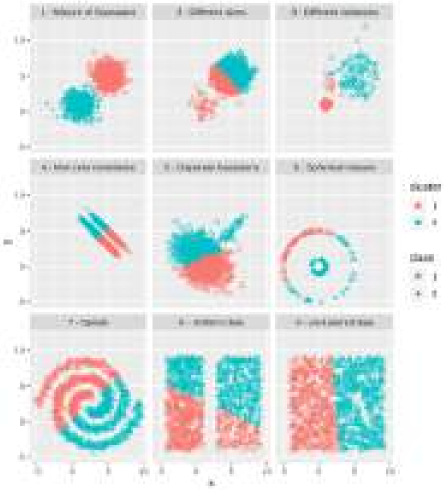
→ Plot and find the bend of the elbow







But most of the time, not as expected. And it is probably because of the L_2 norm that k-means tries to minimize:



→ Sensible to curse of dimensonality

→ Form "normalized Gaussian" clusters (spherical)

→ Does not adapt to manifold geometry

→ Sensible to class imbalance

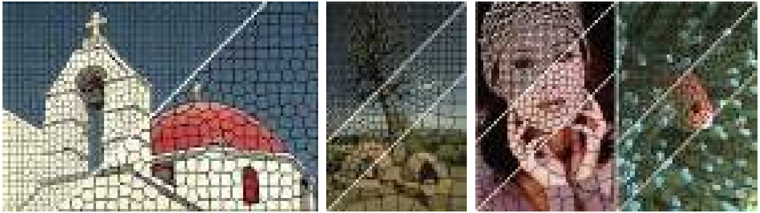
→ Sensible to outliers

SLIC superpixels uses a modified k-means clustering in the *Labxy* space to produce *k* clusters regularly sampled and perceptually coherent from a color point of view.

Algorithm SLIC superpixel segmentation

Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
repeat
  for each cluster center C_k **do**
    Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (Eq. 1).
  end for
  Compute new cluster centers and residual error E {L1 distance between previous centers and recomputed centers}
until $E \le$ threshold
Enforce connectivity.

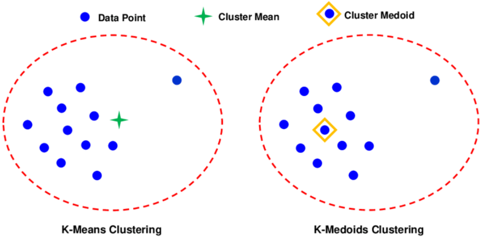
$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$
$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \quad (1)$$
$$D_s = d_{lab} + \frac{m}{S} d_{xy},$$



k-medoids clustering

A possible extension to k-means

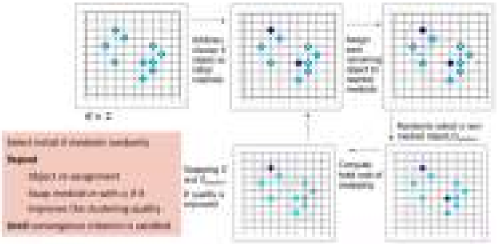
Clusters centroids are not initial data points $\rightarrow$ can be problematic



$\Rightarrow$ Replace centroid by medoid (point with the smallest distance to all other points in the cluster)

$$\mathbf{m}_{\mathcal{C}} = \arg\min_{\mathbf{x} \in \mathcal{C}} \sum_{\mathbf{x}_i \in \mathcal{C}} d(\mathbf{x}, \mathbf{x}_i)$$

$\Rightarrow$ k-medoids algorithm



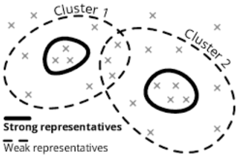
Overall objective: find k medoids $\mathbf{m}_1, \ldots, \mathbf{m}_k$ that minimize the partitioning cost

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_i} d(\mathbf{x}, \mathbf{m}_i)$$

Fuzzy c-means clustering
Let it fuzz

k-means is a *hard* clustering method → each data point 100% belongs to its cluster.

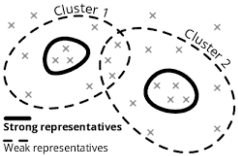
Soft (aka *fuzzy*) clustering methods allow each data point to belong to several clusters with various degrees of membership.



Fuzzy c-means clustering
Let it fuzz

k-means is a *hard* clustering method $\rightarrow$ each data point 100% belongs to its cluster.

Soft (aka *fuzzy*) clustering methods allow each data point to belong to several clusters with various degrees of membership.

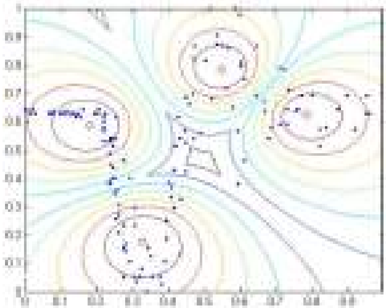


FCM clustering: outputs clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ and membership matrix $\mathbf{W}_{n \times k}$ ($w_{ij} = \%(\mathbf{x}_i \in \mathcal{C}_j)$)

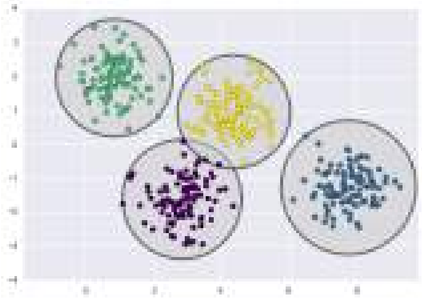
$\Rightarrow \arg\min_{\mathbf{C}=\{C_1,\ldots C_k\}} \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}^{m} \left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|^2$

$\Rightarrow$ Alternatively update $\boldsymbol{\mu}_j = \frac{\sum_{\mathbf{x}_i} w_{ij}^m \mathbf{x}_i}{\sum_{\mathbf{x}_i} {w_{ij}}^m}$ and

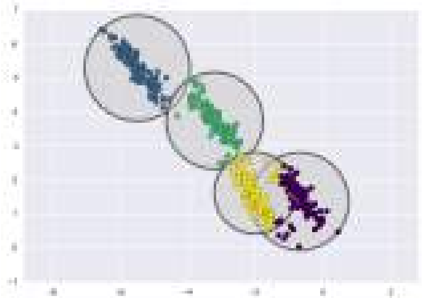
$$w_{ij} = \frac{1}{\sum_{l=1}^{k} \left(\frac{\left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|}{\left\|\mathbf{x}_i - \boldsymbol{\mu}_l\right\|} \right)^{\frac{2}{m-1}}}$$



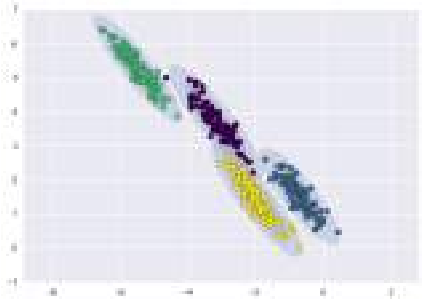
k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian



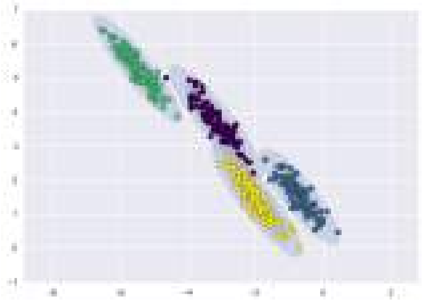
k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian

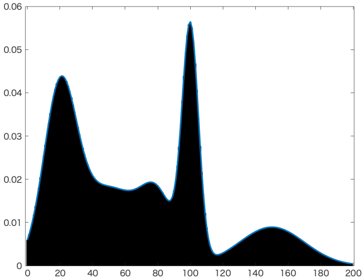


k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian



k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian



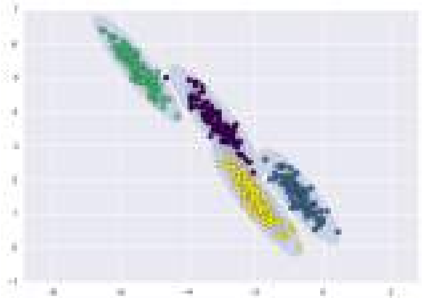


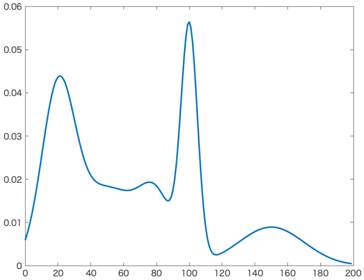
Gaussian mixture models

k-means on steroids

k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder

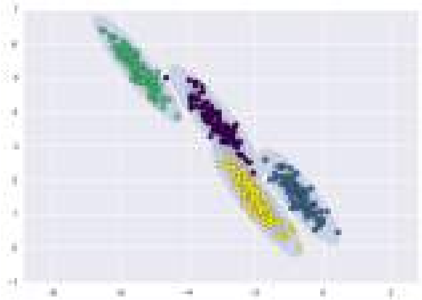
Model probability density function f of data as a mixture of multivariate Gaussian

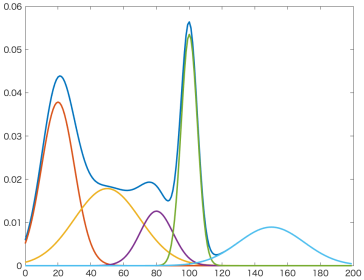




Gaussian mixture models
k-means on steroids

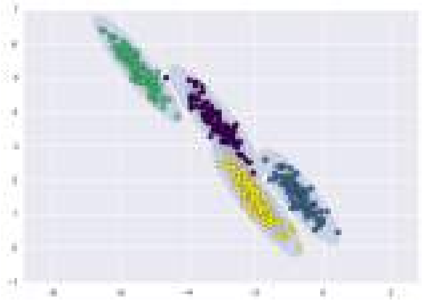
k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian

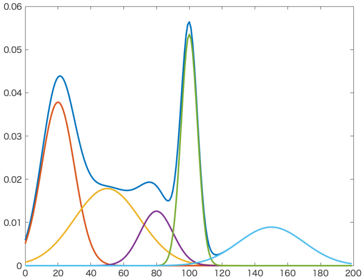




k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder

Model probability density function f of data as a mixture of multivariate Gaussian

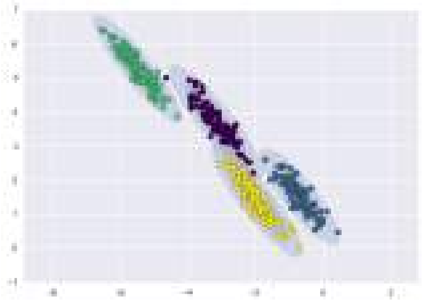


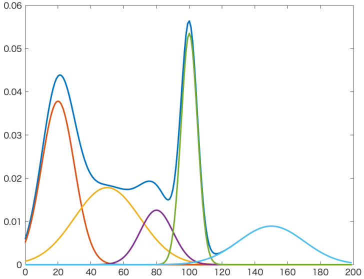


Gaussian mixture model: $f(\mathbf{x}) = \sum_{i=1}^{k} \phi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$

recall that $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^N \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)$ for $\mathbf{x} \in \mathbb{R}^N$

k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian



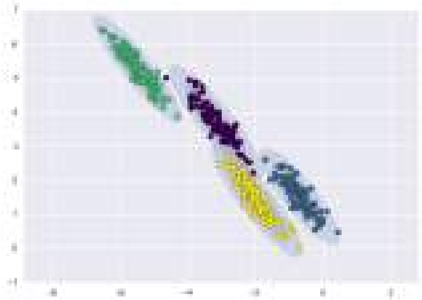


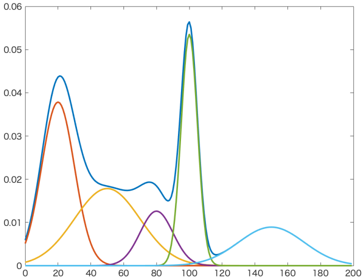
Gaussian mixture model: $f(\mathbf{x}) = \sum_{i=1}^{k} \phi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$

recall that $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^N \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$ for $\mathbf{x} \in \mathbb{R}^N$

$\rightarrow$ ϕ_i are mixture component weights ($\sum_{i=1}^{k} \phi_i = 1$)

k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian





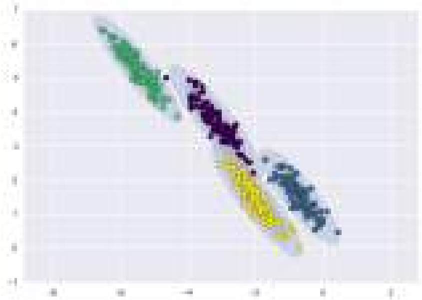
Gaussian mixture model: $f(\mathbf{x}) = \sum_{i=1}^{k} \phi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$

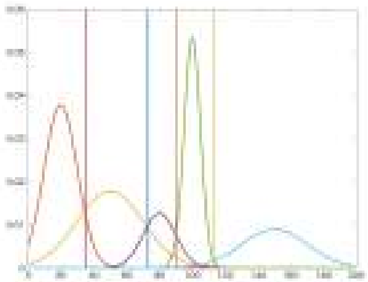
recall that $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^N \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$ for $\mathbf{x} \in \mathbb{R}^N$

$\rightarrow$ ϕ_i are mixture component weights ($\sum_{i=1}^{k} \phi_i = 1$)

$\rightarrow$ How to estimate $\phi_i, \boldsymbol{\mu}_i, \Sigma_i \ \forall i = 1, \ldots, k$?

k-means works for spherical clusters, but fails in any other case $\Rightarrow$ try harder
Model probability density function f of data as a mixture of multivariate Gaussian



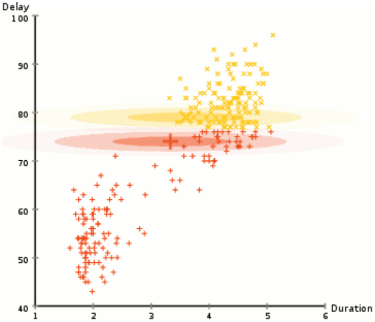


Gaussian mixture model: $f(\mathbf{x}) = \sum_{i=1}^{k} \phi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$

recall that $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^N \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)$ for $\mathbf{x} \in \mathbb{R}^N$

$\rightarrow$ ϕ_i are mixture component weights ($\sum_{i=1}^{k} \phi_i = 1$)

$\rightarrow$ How to estimate $\phi_i, \boldsymbol{\mu}_i, \Sigma_i \ \forall i = 1, \ldots, k$?



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ → $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

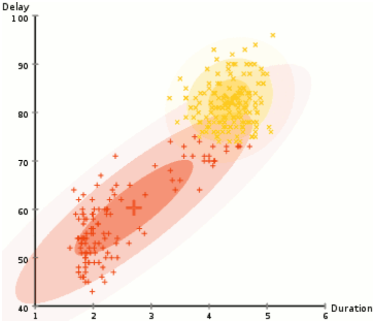
Maximization step

→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$

The EM algorithm

The nightmare of any EECS grad student around the world



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \dots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \dots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \dots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

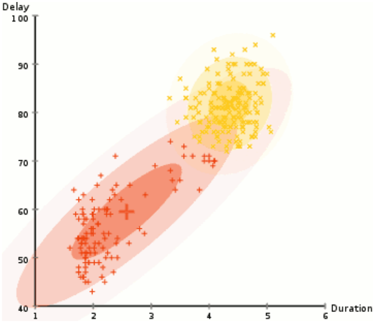
$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ → $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ → $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

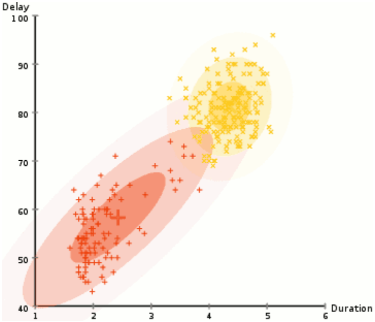
Maximization step

→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$

The EM algorithm

The nightmare of any EECS grad student around the world



Initialization

- $\rightarrow$ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$
- $\rightarrow$ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$
- $\rightarrow$ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

- $\rightarrow$ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

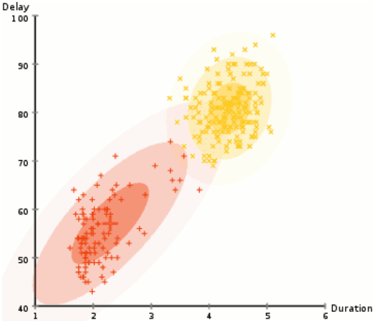
$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ $\rightarrow$ $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

- $\rightarrow$ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\quad \hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\quad \hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \dots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \dots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \dots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

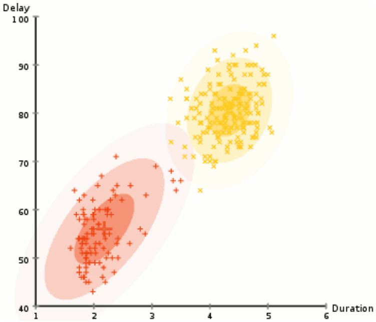
$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ → $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\quad \hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\quad \hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

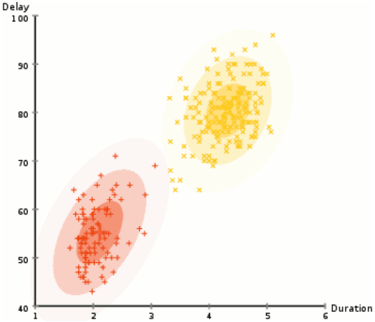
$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ → $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\quad \hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\quad \hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

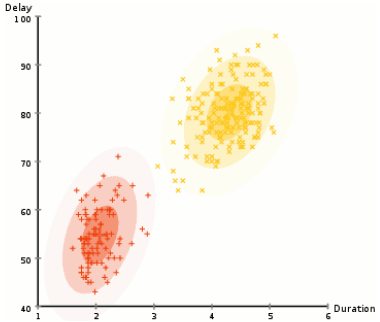
$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ → $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i \rightarrow \sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

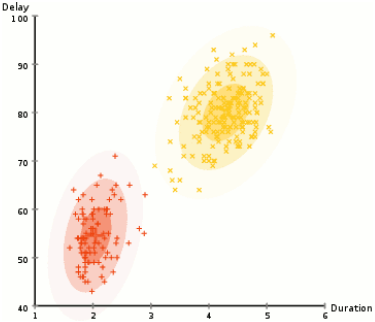
Maximization step

→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\quad \hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\quad \hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$

The EM algorithm

The nightmare of any EECS grad student around the world



Initialization

- $\rightarrow$ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \dots, \hat{\boldsymbol{\mu}}_k$
- $\rightarrow$ Init all covariance matrices $\hat{\Sigma}_1, \dots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$
- $\rightarrow$ Set uniform mixture weights $\hat{\phi}_1, \dots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

- $\rightarrow$ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

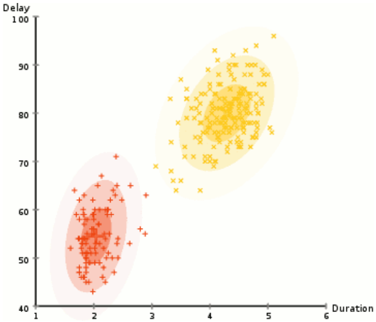
$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i \rightarrow \sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

- $\rightarrow$ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\quad \hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\quad \hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$



Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$

Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i$ → $\sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

→ Update weights (in that order)

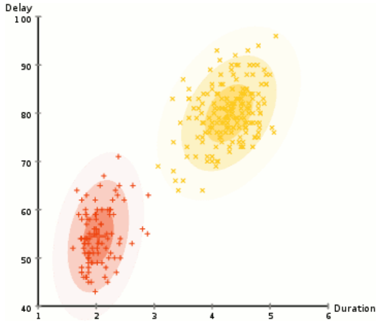
$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\quad \hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\quad \hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$

Initialization

→ Select k random points as initial means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_k$

→ Init all covariance matrices $\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$ as whole data sample covariance matrix $\hat{\Sigma}$

→ Set uniform mixture weights $\hat{\phi}_1, \ldots, \hat{\phi}_k = \frac{1}{k}$



Alternate until convergence

Expectation step

→ Compute membership weight $\hat{\gamma}_{ij}$ of $\mathbf{x}_i$ with respect to j$^{\text{th}}$ component $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)$

$$\hat{\gamma}_{ij} = \frac{\hat{\phi}_j \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\Sigma}_j)}{\sum_{m=1}^{k} \hat{\phi}_m \mathcal{N}(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_m, \hat{\Sigma}_m)}$$

$\hat{\gamma}_{ij} \equiv$ posterior probability of j$^{\text{th}}$ component given data $\mathbf{x}_i \rightarrow \sum_{j=1}^{k} \hat{\gamma}_{ij} = 1$

Maximization step

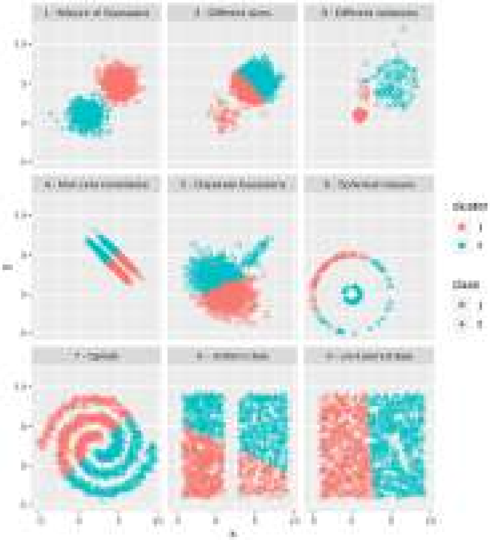
→ Update weights (in that order)

$\hat{\phi}_j = \frac{N_j}{n}$ with $N_j = \sum_i \hat{\gamma}_{ij}$; $\quad \hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} \mathbf{x}_i$; $\quad \hat{\Sigma}_j = \frac{1}{N_j} \sum_i \hat{\gamma}_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T$

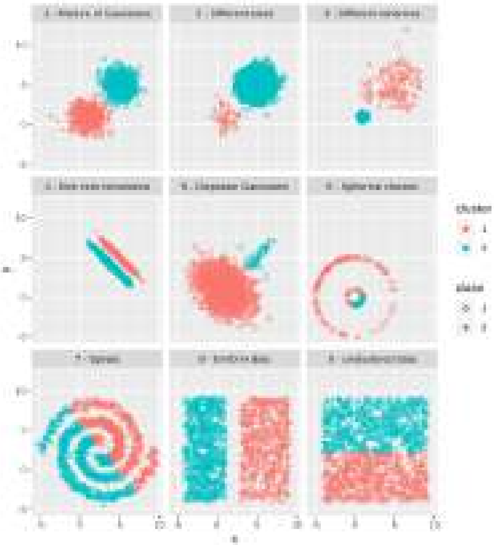
k-means vs GMM

Let the fight begin

k-means

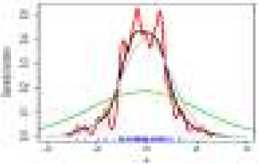


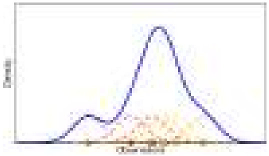
GMM

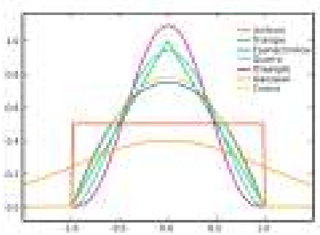


Worth the pain, right?

Goal: Estimate probability density function f based on observation $x_1 \dots, x_n$ only, assumed to derive from f (otherwise wtf are we doing here?)





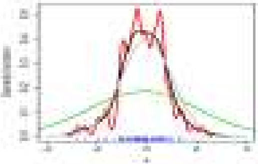


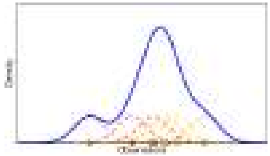
The kernel density estimator with bandwidth h at a given point x is given by

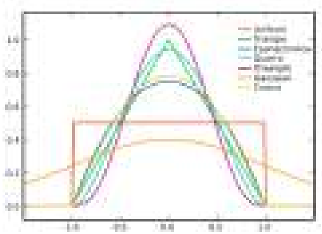
$$\widehat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\Big(\frac{x - x_i}{h}\Big),$$

Kernel density estimation
Nonparametric estimation

Goal: Estimate probability density function f based on observation $x_1 \dots, x_n$ only, assumed to derive from f (otherwise wtf are we doing here?)







The kernel density estimator with bandwidth h at a given point x is given by

$$\widehat{f_h}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$