What is a cluster?

Computer cluster

Asteroid cluster

Activity clusters

Winter cluster

Consonant cluster

A group of entities which are alike. Entities from different clusters are not alike

Clustering

The task of grouping a set of objects in such a way that objects in the same group (**a cluster**) are more **similar** (in some sense) to each other than to those in other groups (clusters).

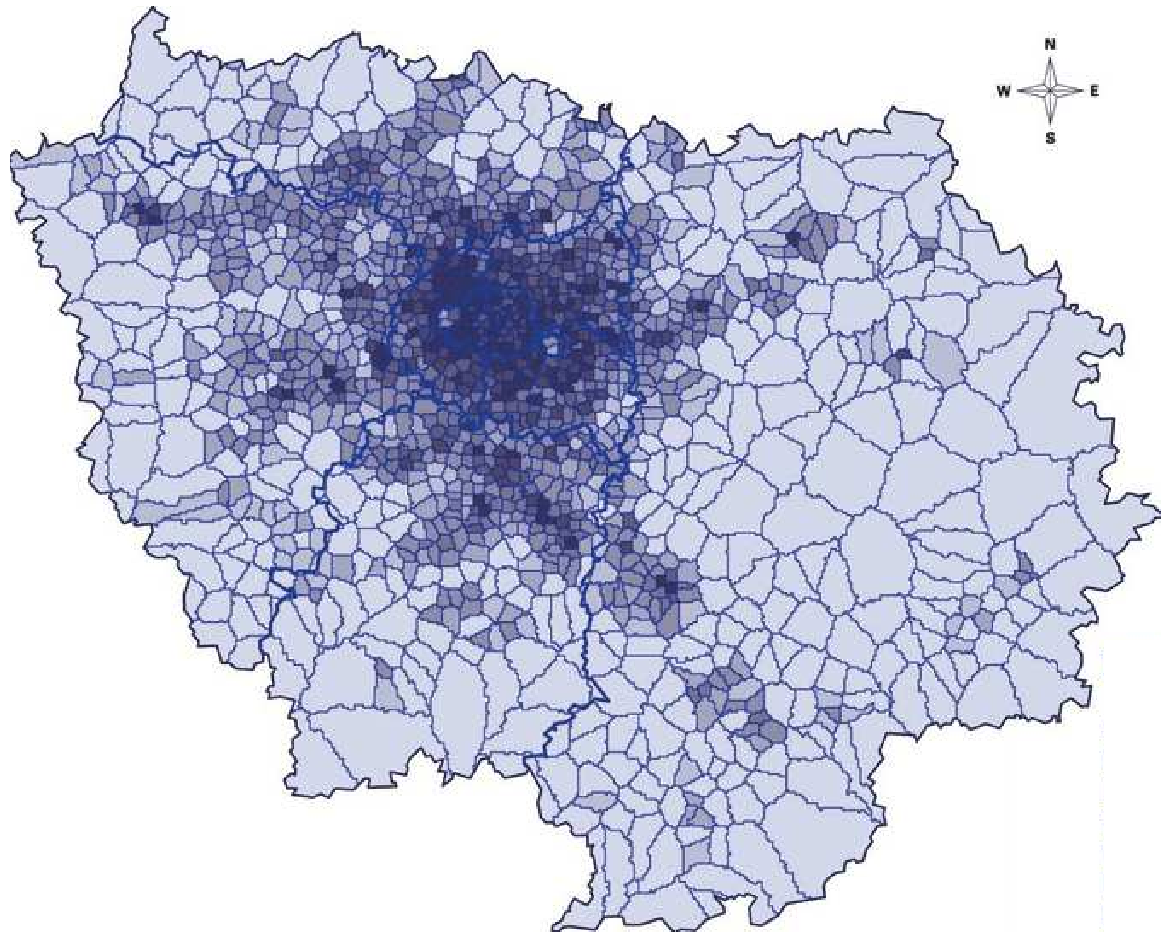
Main task of exploratory data mining, and a common technique for statistical data analysis

Fields of use: machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

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

Examples in healthcare

Gene expression
Shamir et al.
BMC Bioinformatics 2014



Population study
Lefèvre et al. PlosOne 2014

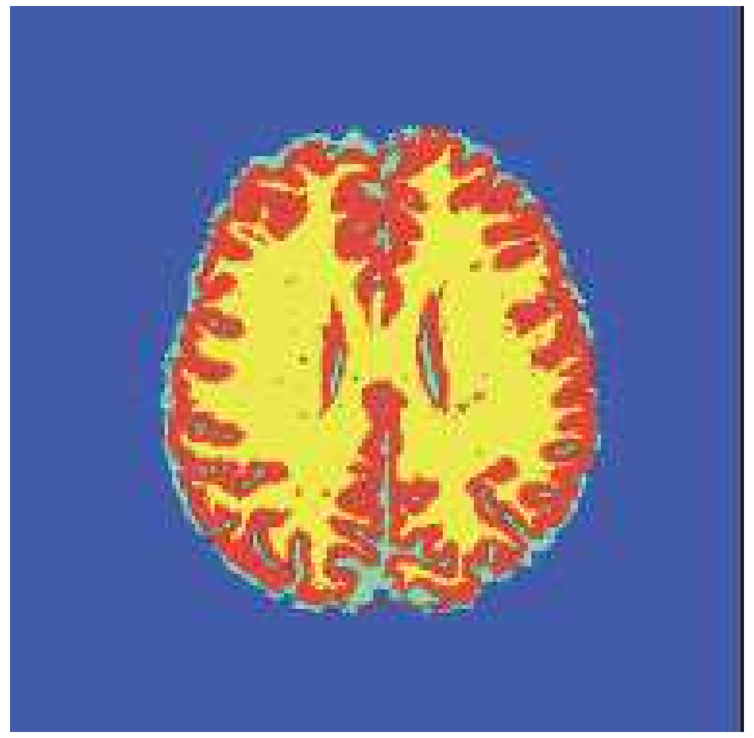
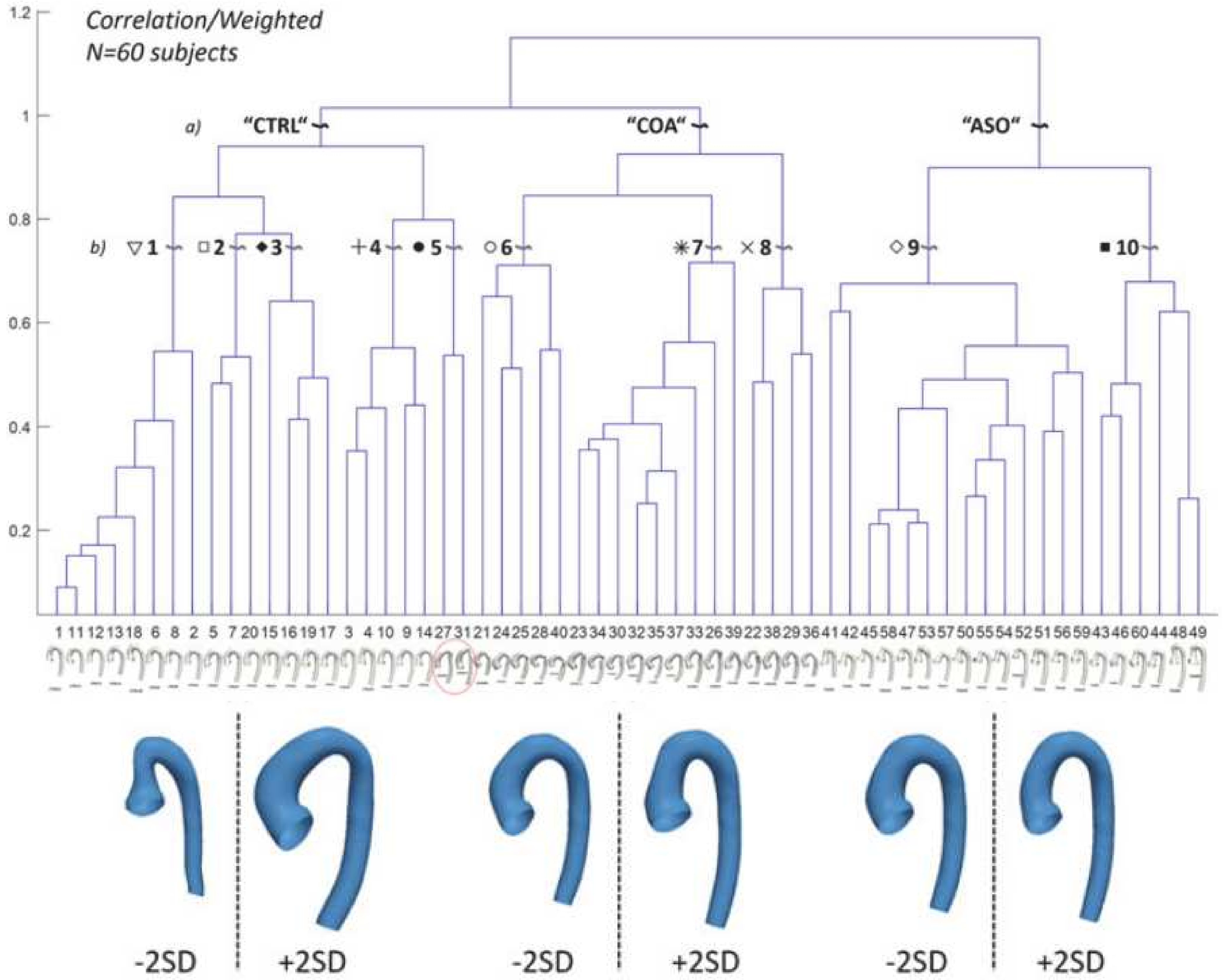


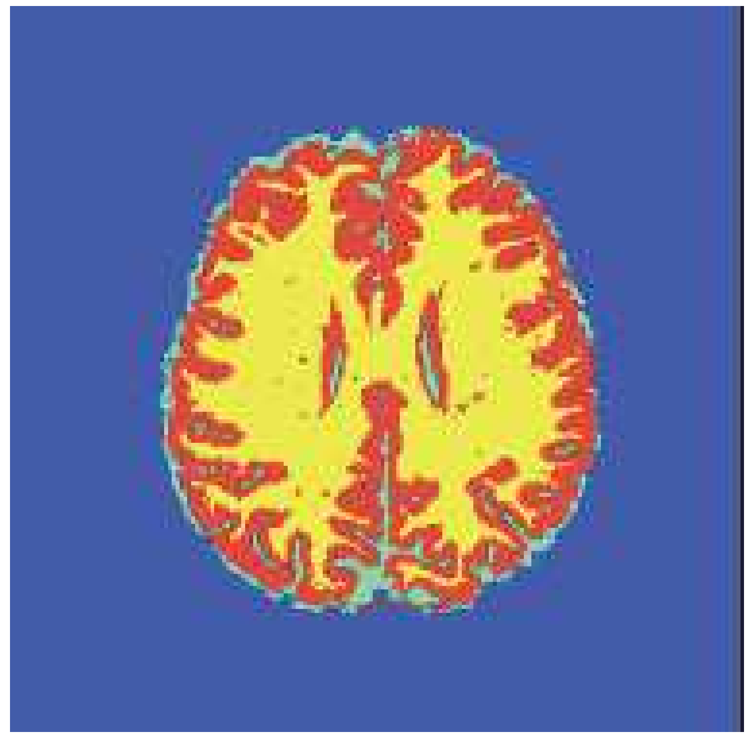
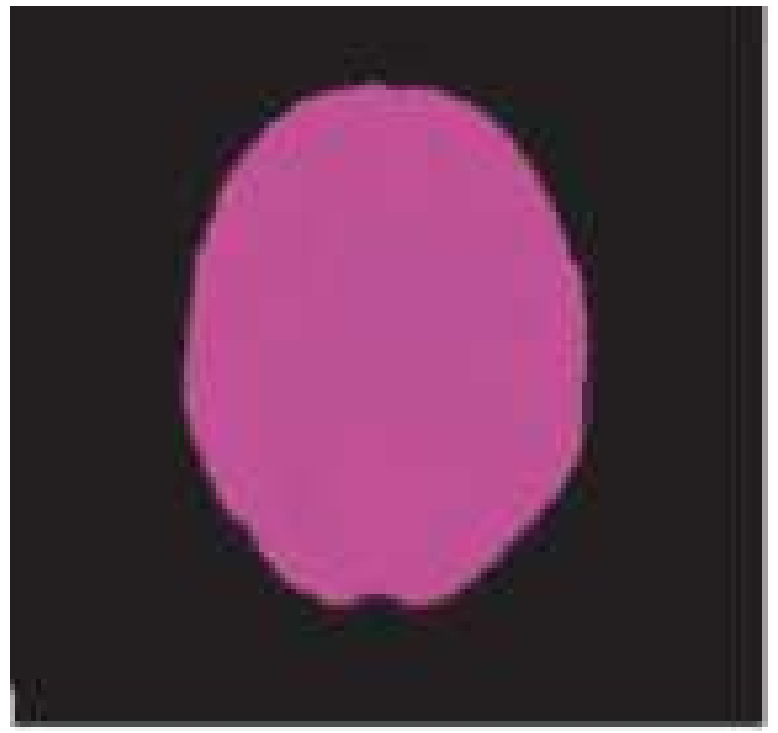
Image segmentation
Ashburner & Friston Neuroimage 2005



Shape clusters
Bruse et al. IEEE TBME 2017

Clustering: Key elements

- Similarity criteria
- Number of groups
- Features

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

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

K-means: Formalisation

K-means: Formalisation

- Dataset: $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbf{R}^D$

K-means: Formalisation

- Dataset: $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbf{R}^D$
- **Goal v1.0:** Partition the dataset into K clusters

K-means: Formalisation

- Dataset: $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbf{R}^D$
- **Goal v1.0:** Partition the dataset into K clusters
- D-dimensional vector $\mu_k, \; k = 1, \ldots, K$, a prototype associated to the k^{th} cluster.

K-means: Formalisation

- Dataset: $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbf{R}^D$
- **Goal v1.0:** Partition the dataset into K clusters
- D-dimensional vector $\mu_k,\ k = 1,\ldots,K$, a prototype associated to the k^{th} cluster.
- **Goal v2.0:** Find an assignment of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ to clusters, and a set of vectors $\{\mu_k\}$, which minimise the sum of square distances of each point to its closest vector

K-means: Formalisation

K-means: Formalisation

- To designate membership of each $\mathbf{x}_i$, use a binary indicator variable:

$$r_{ik} \in \{0,1\} \qquad \begin{array}{l} k = 1,\ldots,K \\ i = 1,\ldots,N \end{array}$$

K-means: Formalisation

- To designate membership of each $\mathbf{x}_i$, use a binary indicator variable:

$$r_{ik} \in \{0,1\} \qquad \begin{array}{l} k = 1,\ldots,K \\ i = 1,\ldots,N \end{array}$$

- $r_{ik} = 1$ when the point x_i belongs to cluster k. $r_{ij} = 0$ for $j \neq k$

K-means: Formalisation

- To designate membership of each $\mathbf{x}_i$, use a binary indicator variable:

$$r_{ik} \in \{0,1\} \qquad \begin{array}{l} k = 1,\ldots,K \\ i = 1,\ldots,N \end{array}$$

- $r_{ik} = 1$ when the point x_i belongs to cluster k. $r_{ij} = 0$ for $j \neq k$

x	k
1	GM
2	WM
3	CSF
4	WM
5	WM

Clusters = {GM, WM, CSF}

x	r
1	(1,0,0)
2	(0,1,0)
3	(0,0,1)
4	(0,1,0)
5	(0,1,0)

This representation scheme is denoted 1-of-K coding.
It is better know as one hot encoding

K-means: Goal v3.0

- Find k prototype vectors $\{\mu_k\}$ and a 1-of-K coding scheme, represented by r_{ik}, that minimises the *distortion measure* given by

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} ||\mathbf{x}_i - \mu_k||^2$$

K-means: Goal v3.0

- Find k prototype vectors $\{\mu_k\}$ and a 1-of-K coding scheme, represented by r_{ik}, that minimises the *distortion measure* given by

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} ||\mathbf{x}_i - \mu_k||^2$$

Sum of the squares of the distances of each data point to vector μ_k

Optimisation algorithm

- Iterative procedure involving optimisation of r_{ik}, μ_k

```
Initialise μ_k

Iterate:

  Minimise J w.r.t r_ik keeping fixed μ_k

  Minimise J w.r.t μ_k keeping fixed r_ik

repeat until convergence
```

Optimisation algorithm: r_{ik}

- Minimisation of J w.r.t r_{ik}

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

Optimisation algorithm: r_{ik}

- Minimisation of J w.r.t r_{ik}

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \underbrace{\|\mathbf{x}_i - \mu_k\|^2}$$

Independent for each i
Only one term is non-zero

Optimisation algorithm: r_{ik}

- Minimisation of J w.r.t r_{ik}

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \underbrace{||\mathbf{x}_i - \mu_k||^2}_{}$$

Independent for each i
Only one term is non-zero

$$r_{i1}||\mathbf{x}_i - \mu_1||^2, r_{i2}||\mathbf{x}_i - \mu_2||^2, r_{i3}||\mathbf{x}_i - \mu_3||^2, \ldots, r_{iK}||\mathbf{x}_i - \mu_K||^2$$

Optimisation algorithm: r_{ik}

- Minimisation of J w.r.t r_{ik}

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \underbrace{||\mathbf{x}_i - \mu_k||^2}_{}$$

Independent for each i
Only one term is non-zero

$$r_{i1}||\mathbf{x}_i - \mu_1||^2, r_{i2}||\mathbf{x}_i - \mu_2||^2, r_{i3}||\mathbf{x}_i - \mu_3||^2, \ldots, r_{iK}||\mathbf{x}_i - \mu_K||^2$$

$$\boxed{r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\mathbf{x}_i - \mu_j||^2 \\ 0 & \text{otherwise} \end{cases}}$$

Optimisation algorithm: μ_k

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

Optimisation algorithm: μ_k

- Minimisation of J w.r.t μ_k

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

Optimisation algorithm: μ_k

- Minimisation of J w.r.t μ_k

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} ||\mathbf{x}_i - \mu_k||^2$$

- Derive J w.r.t μ_k

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} r_{ik} ||\mathbf{x}_i - \mu_k||^2$$

Optimisation algorithm: μ_k

- Minimisation of J w.r.t μ_k

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

- Derive J w.r.t μ_k

$$J = \sum_{k=1}^{K} \underbrace{\sum_{i=1}^{N} r_{ik} \|\mathbf{x}_i - \mu_k\|^2}_{\mathbf{J_k}}$$

Optimisation algorithm: μ_k

- Minimisation of J w.r.t μ_k

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

- Derive J w.r.t μ_k

$$J = \sum_{k=1}^{K} \underbrace{\sum_{i=1}^{N} r_{ik} \|\mathbf{x}_i - \mu_k\|^2}_{\mathbf{J_k}}$$

$$2 \sum_{i=1}^{N} r_{ik}(\mathbf{x}_i - \mu_k) = 0 \quad \Rightarrow \quad \boxed{\mu_k = \frac{\sum_i r_{ik} \mathbf{x}_i}{\sum_i r_{ik}}}$$

Optimisation algorithm: μ_k

- Minimisation of J w.r.t μ_k

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

- Derive J w.r.t μ_k

$$J = \sum_{k=1}^{K} \underbrace{\sum_{i=1}^{N} r_{ik} \|\mathbf{x}_i - \mu_k\|^2}_{\mathbf{J_k}}$$

$$2 \sum_{i=1}^{N} r_{ik}(\mathbf{x}_i - \mu_k) = 0$$

$$\boxed{\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}}$$

Optimisation algorithm: μ_k

- Minimisation of J w.r.t μ_k

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

- Derive J w.r.t μ_k

J_k

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} r_{ik} \|\mathbf{x}_i - \mu_k\|^2$$

$$2 \sum_{i=1}^{N} r_{ik}(\mathbf{x}_i - \mu_k) = 0 \quad \Rightarrow \quad \boxed{\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}}$$

K-means

K-means algorithm

Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

for each μ_k:

$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

until μ_k converges

K-means algorithm

Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

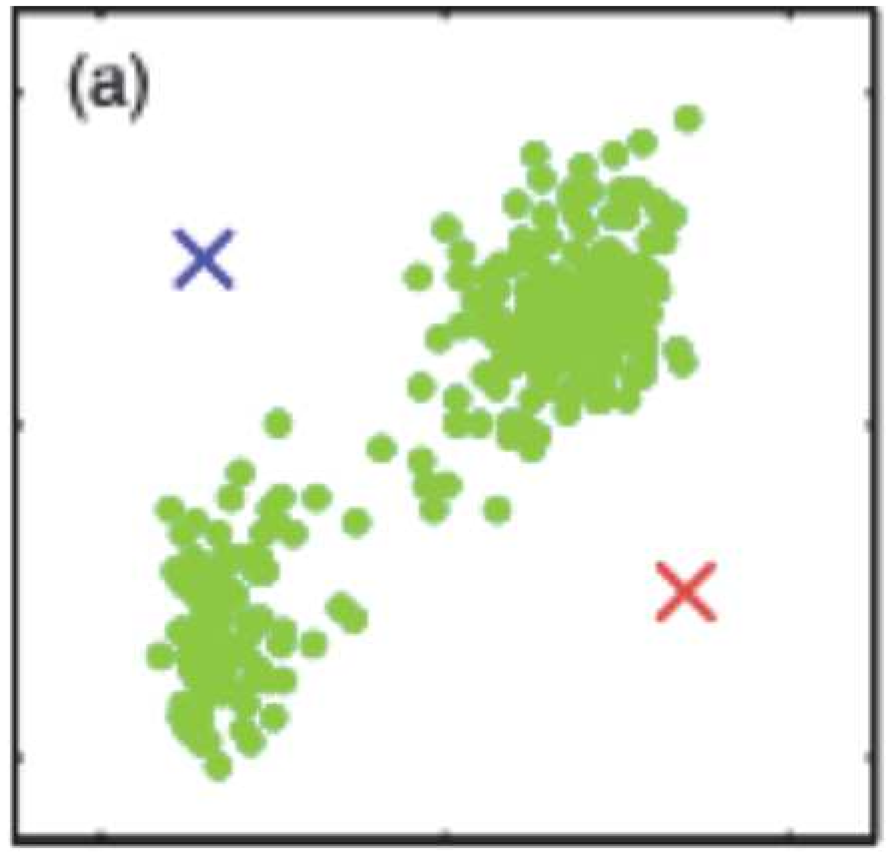
for each μ_k:

$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

K-means algorithm



- Initialise $\mu_k, k = 1,\ldots,K$

  Repeat

    for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

    for each μ_k:

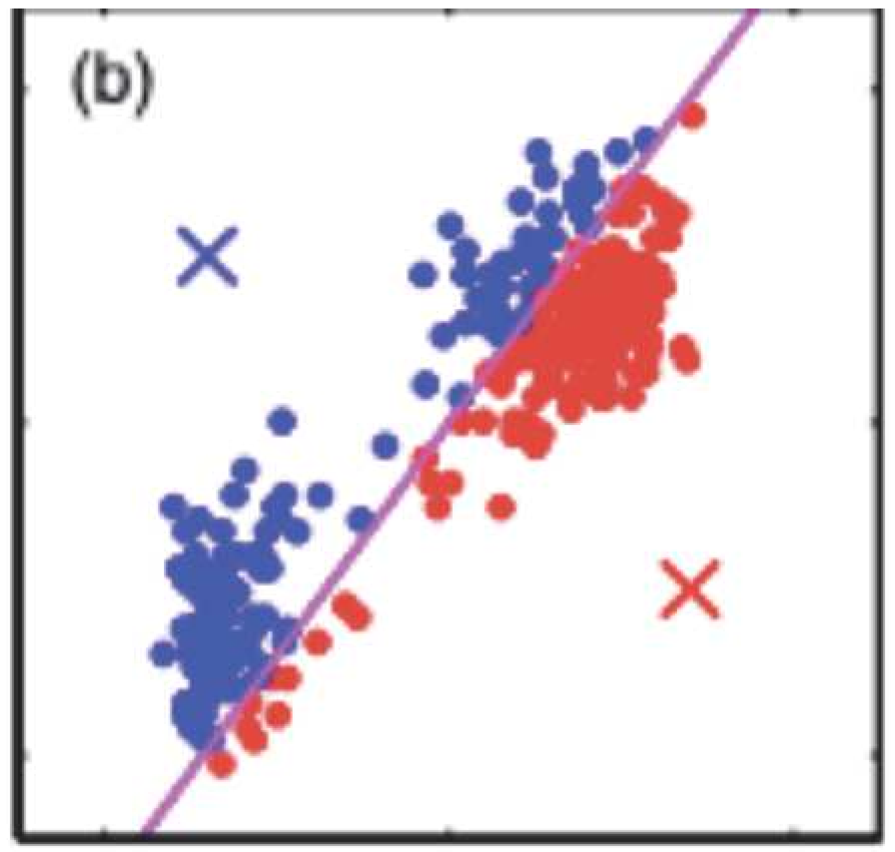
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

  until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

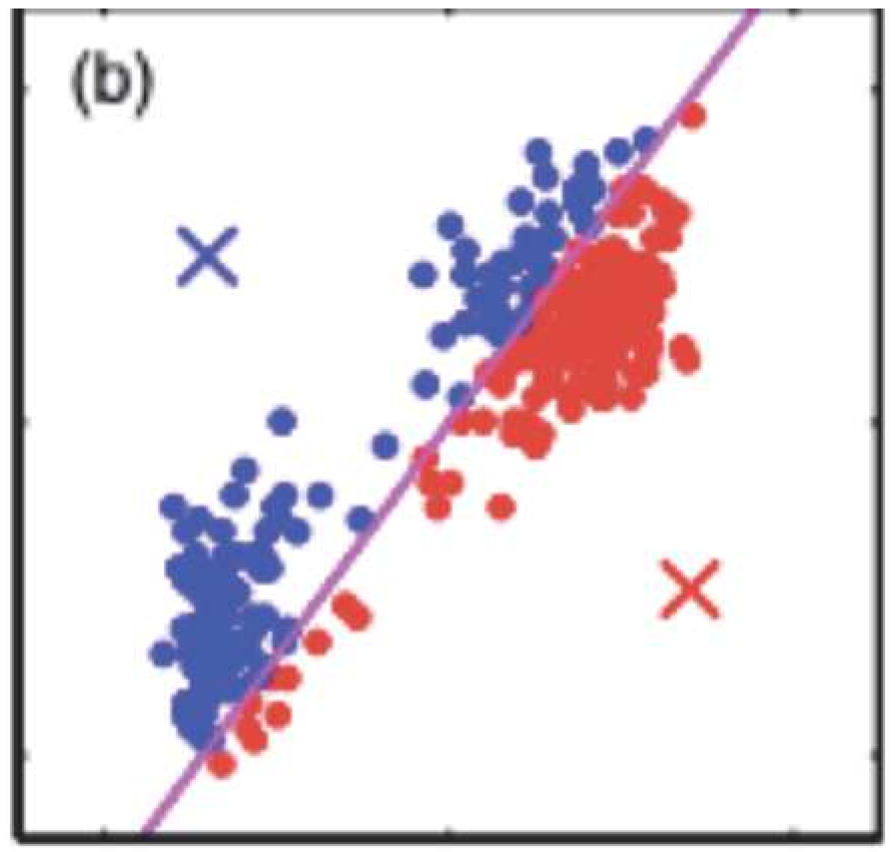
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

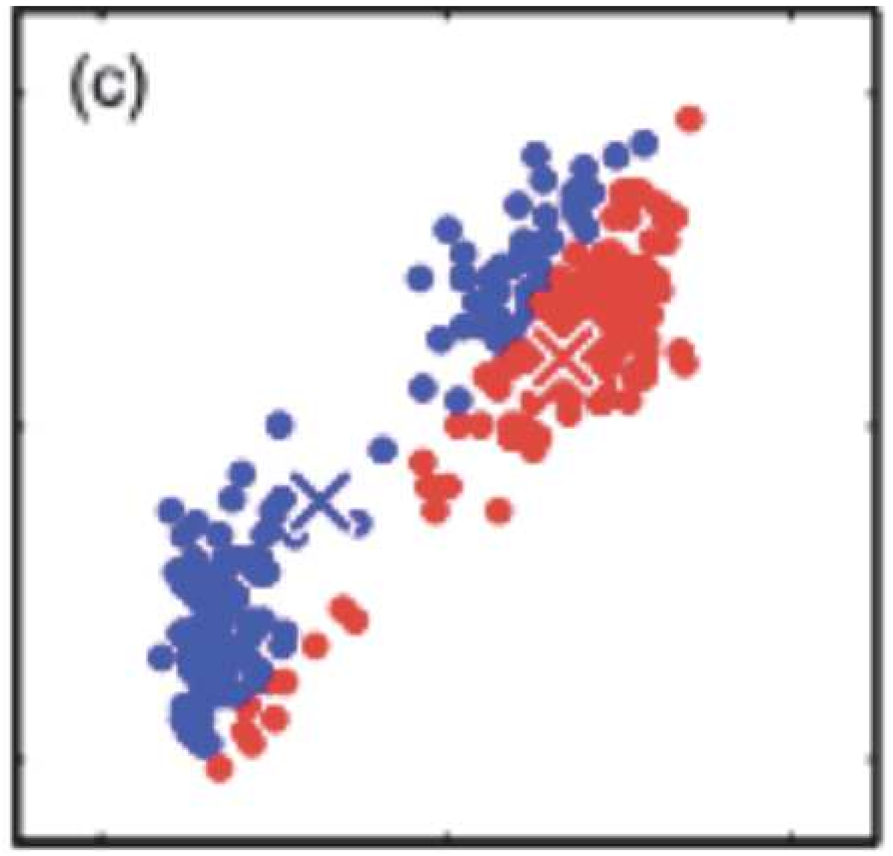
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

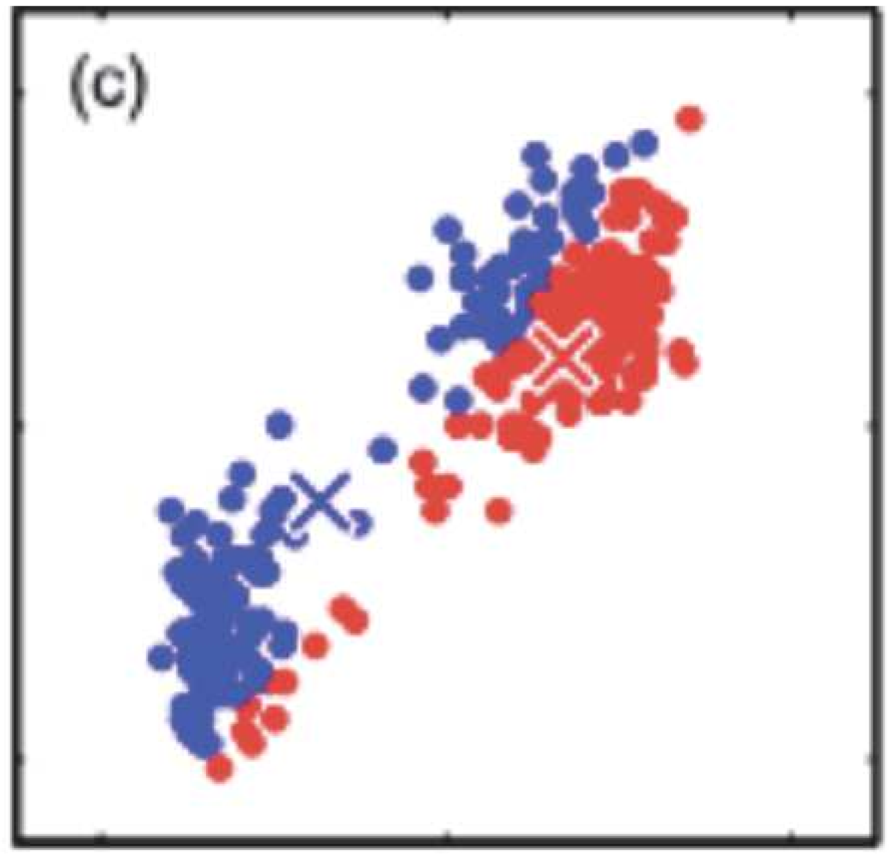
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

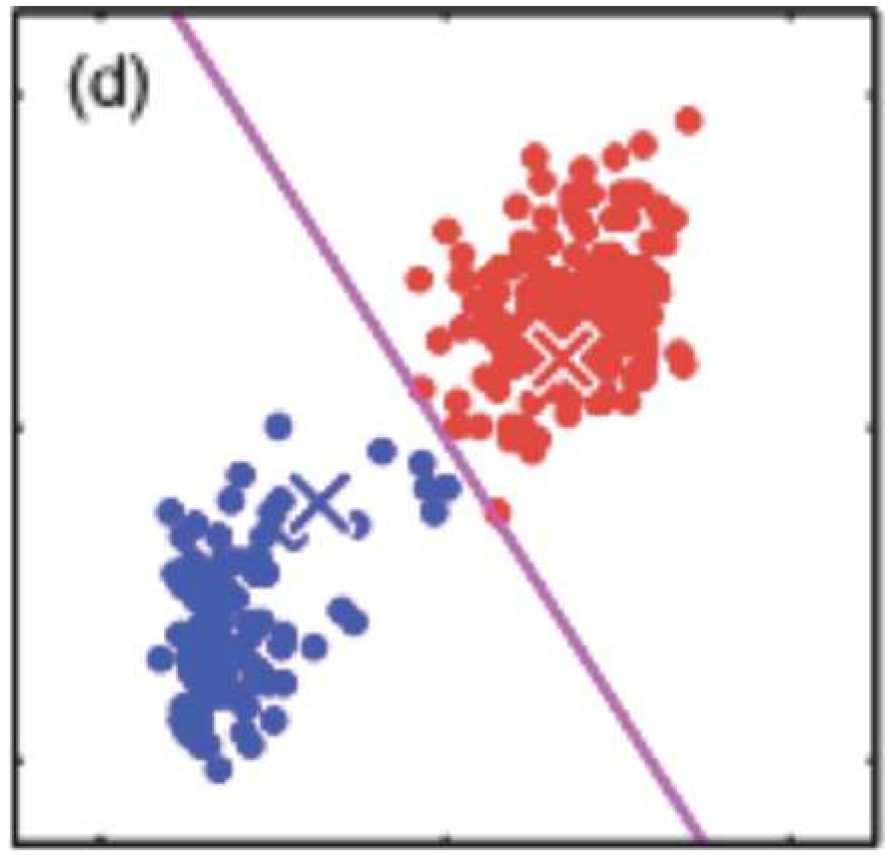
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

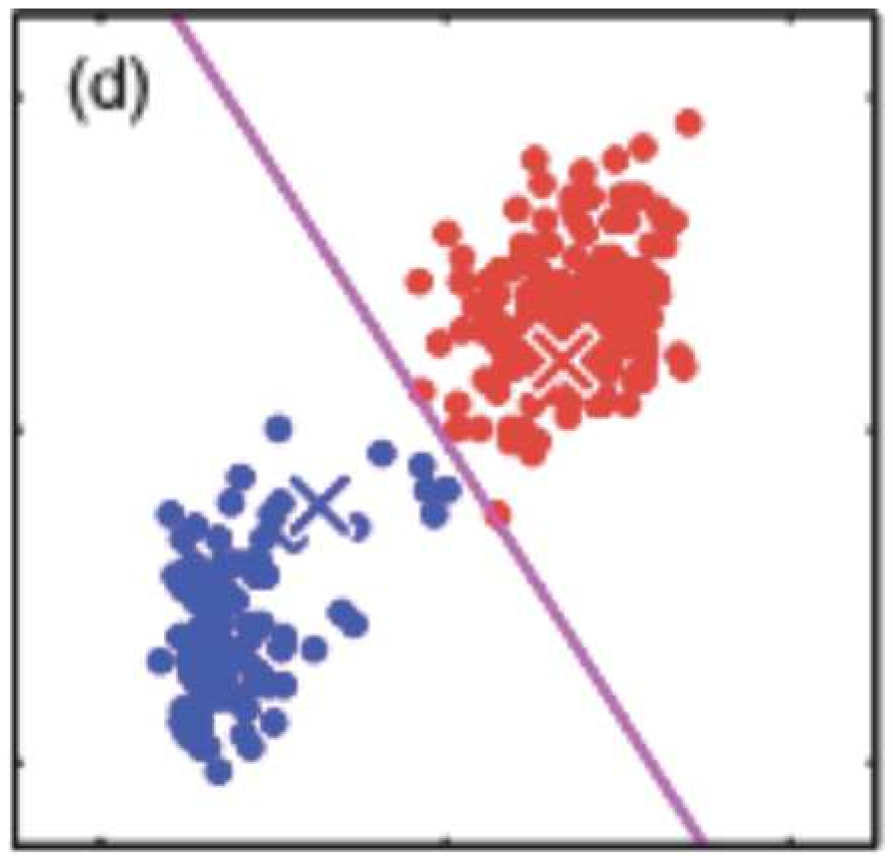
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

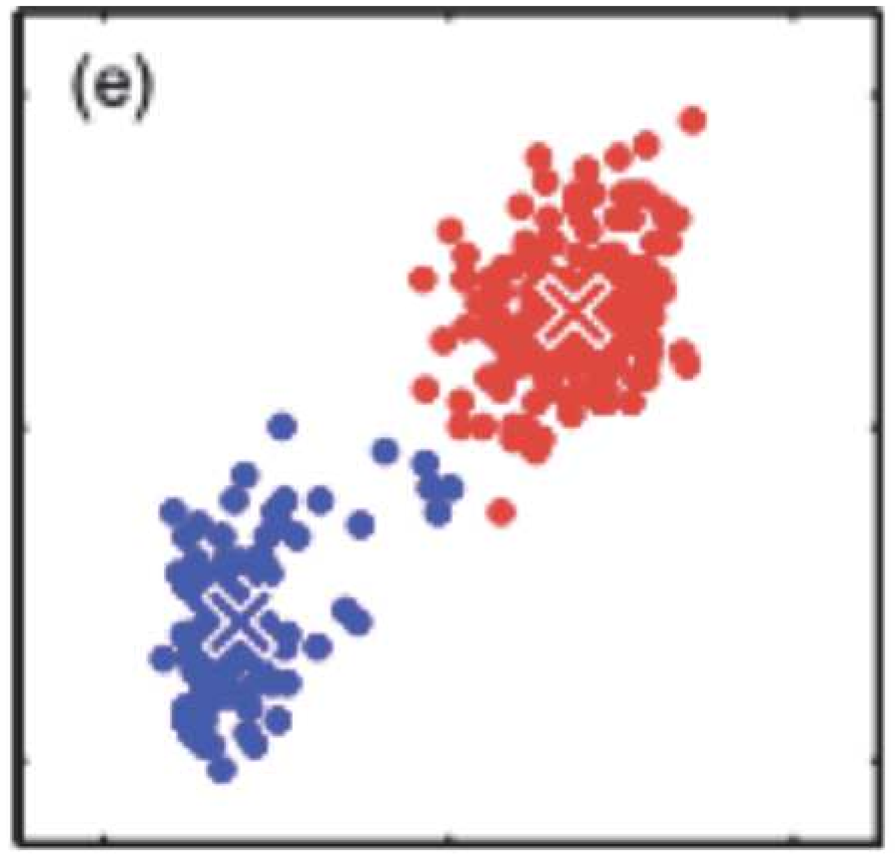
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

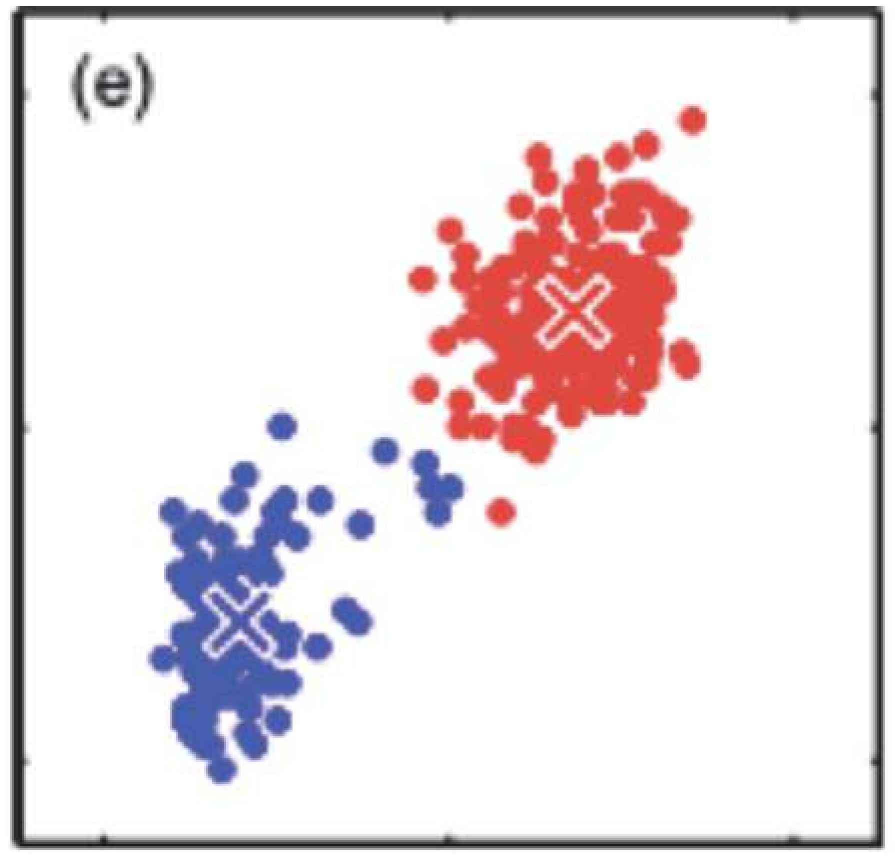
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

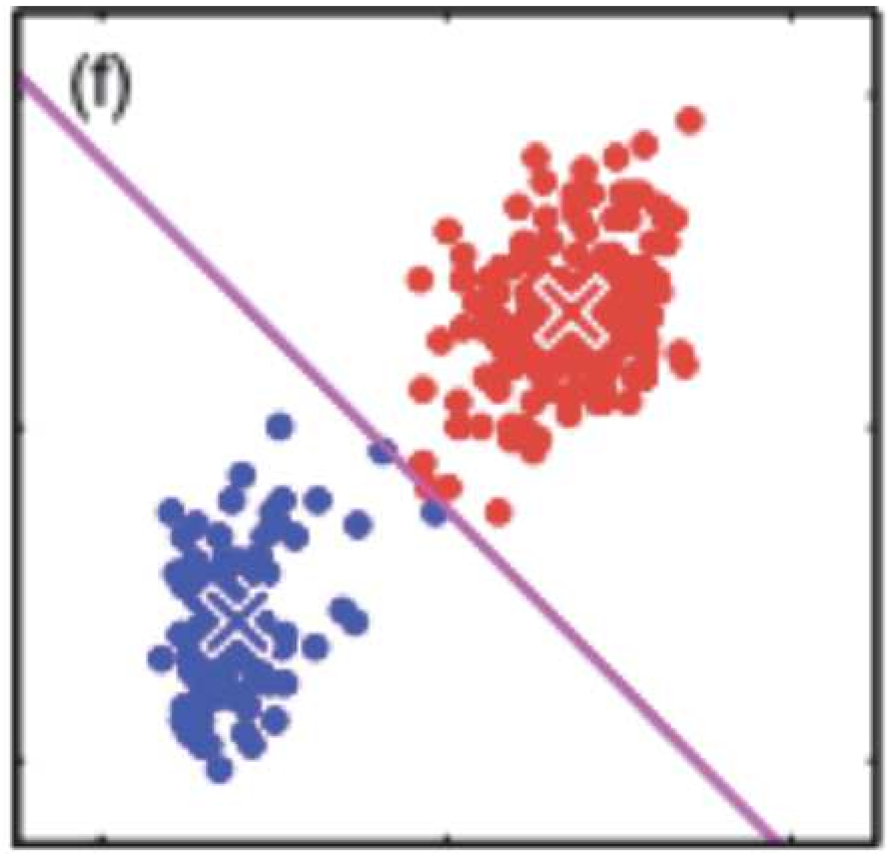
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

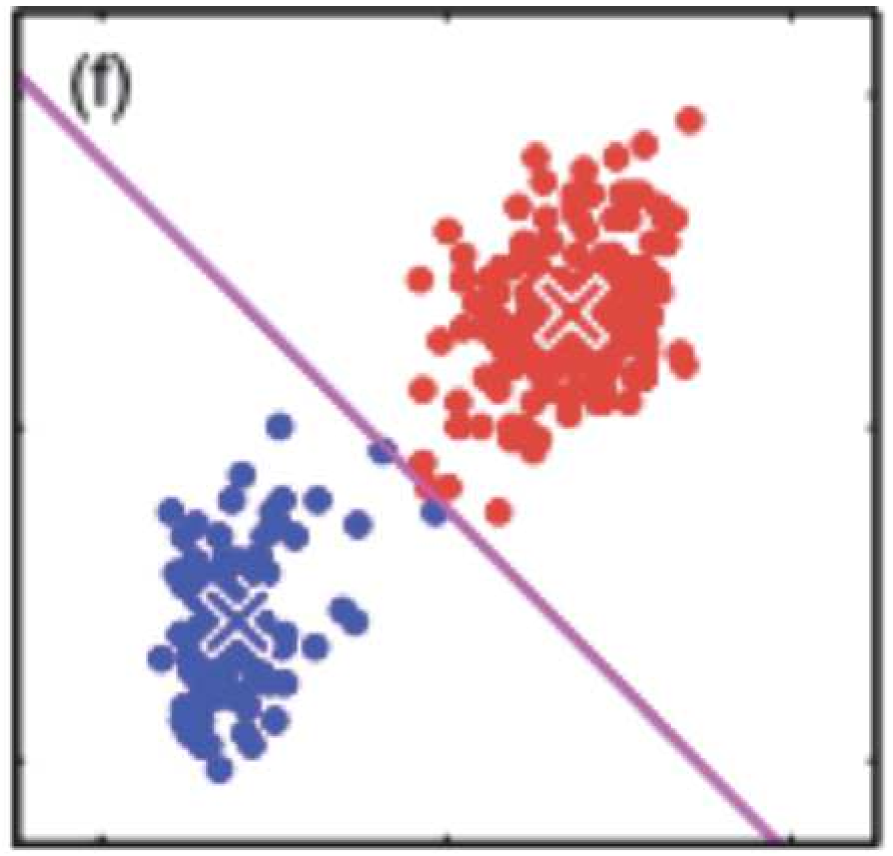
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

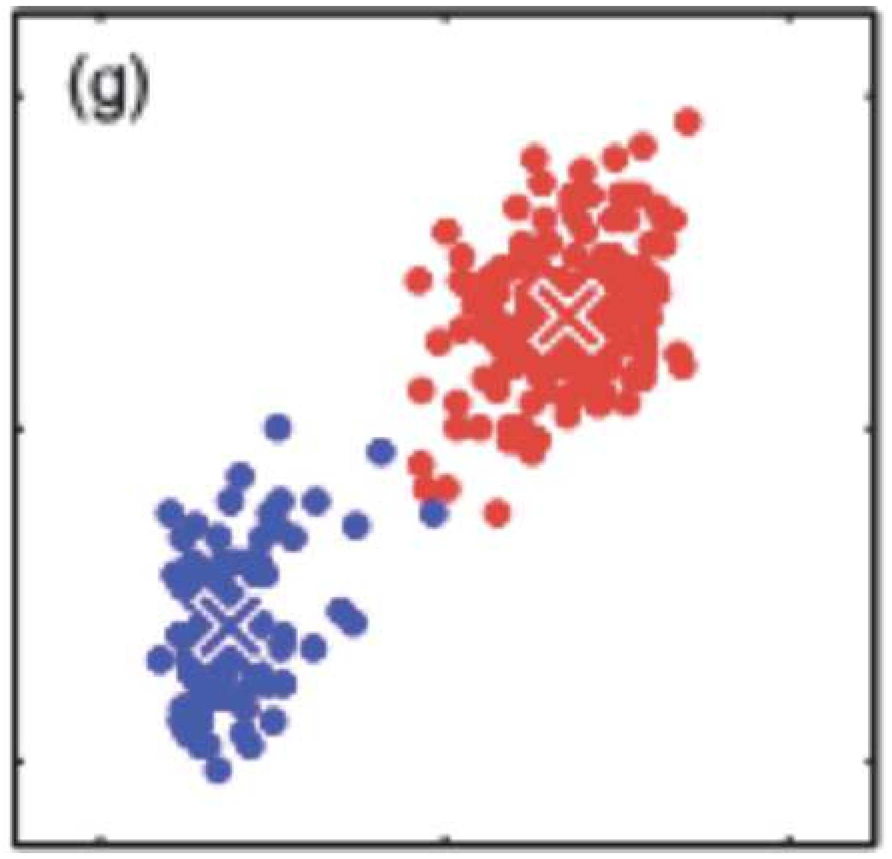
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

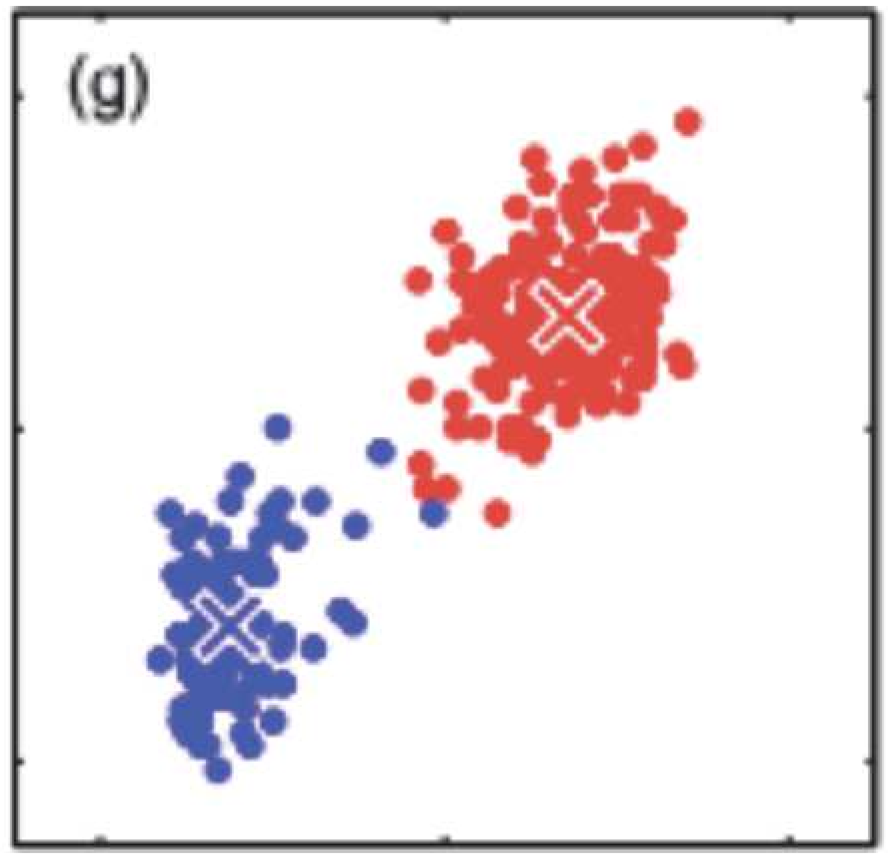
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

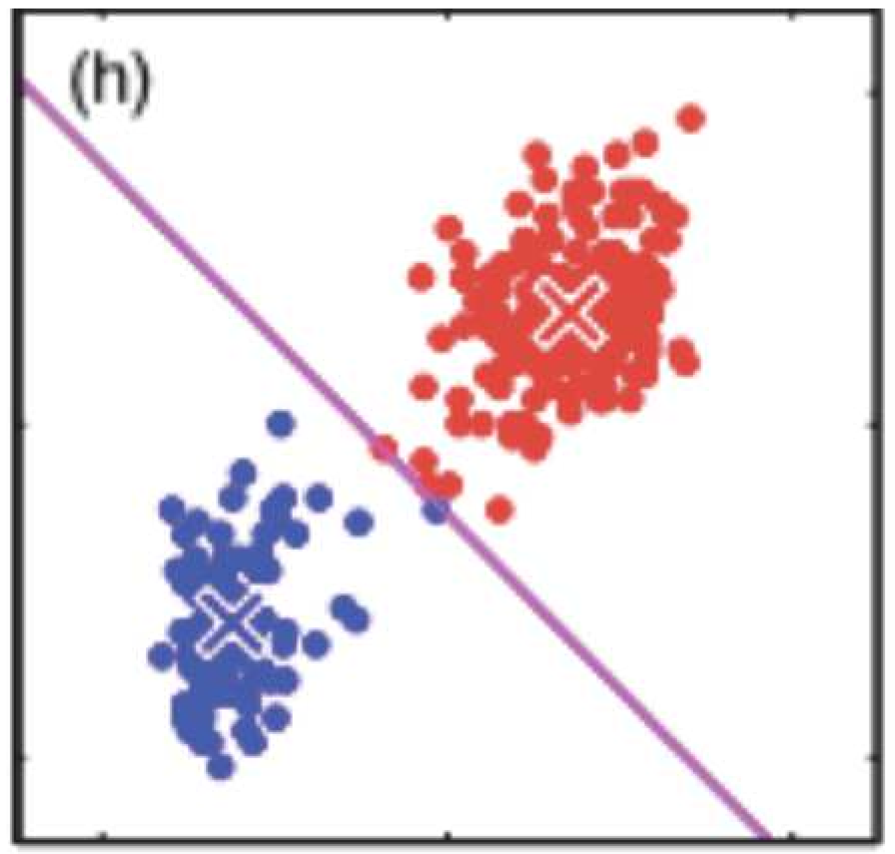
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

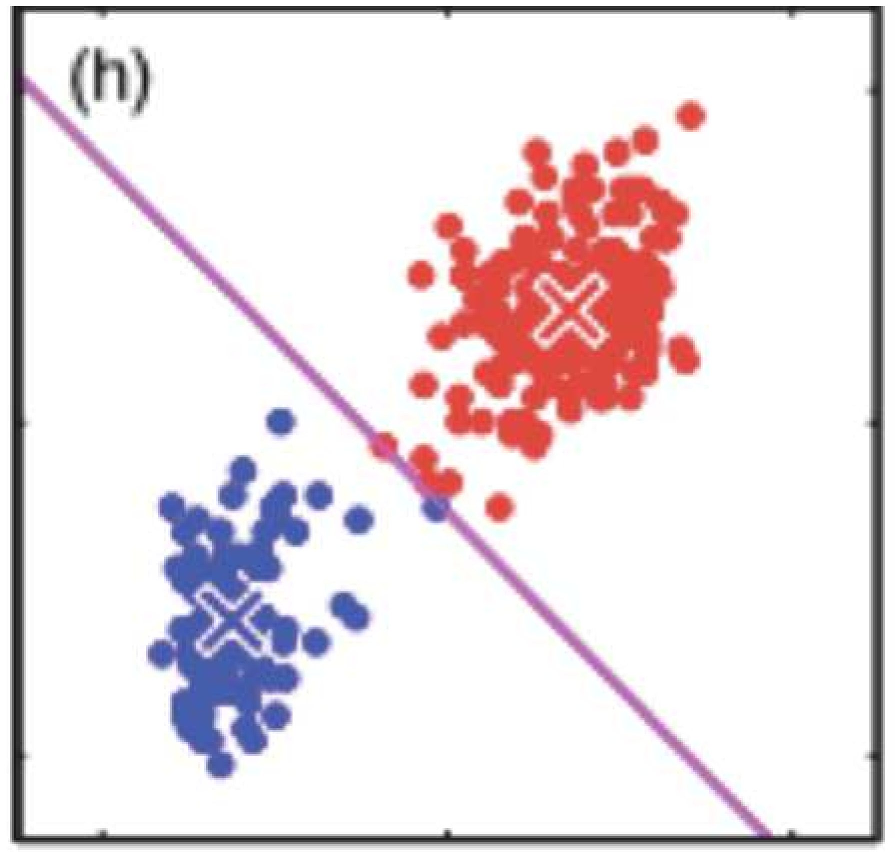
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

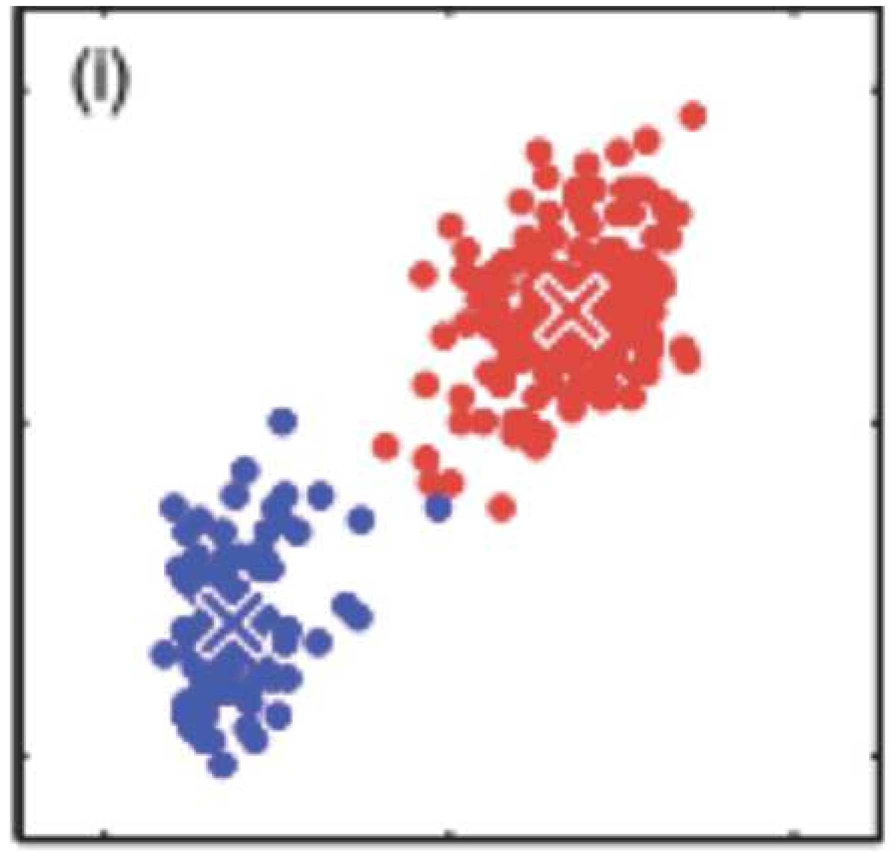
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

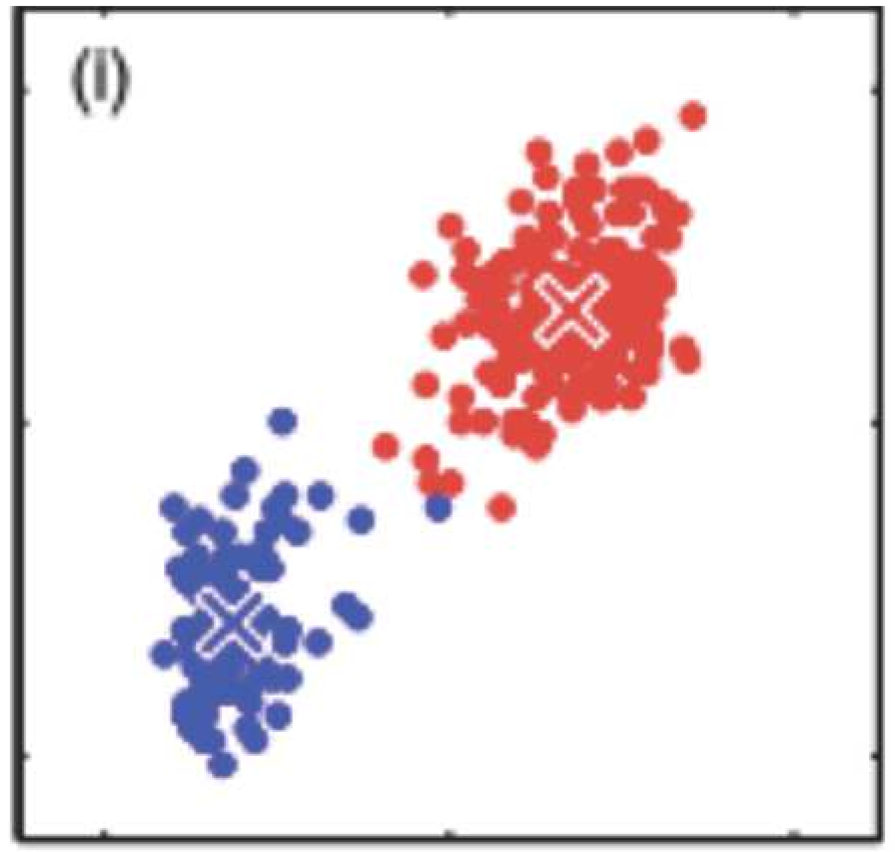
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

E-step (expectation)
1- Find nearest centroid for each point
2- Assign points to clusters (update)

for each μ_k:

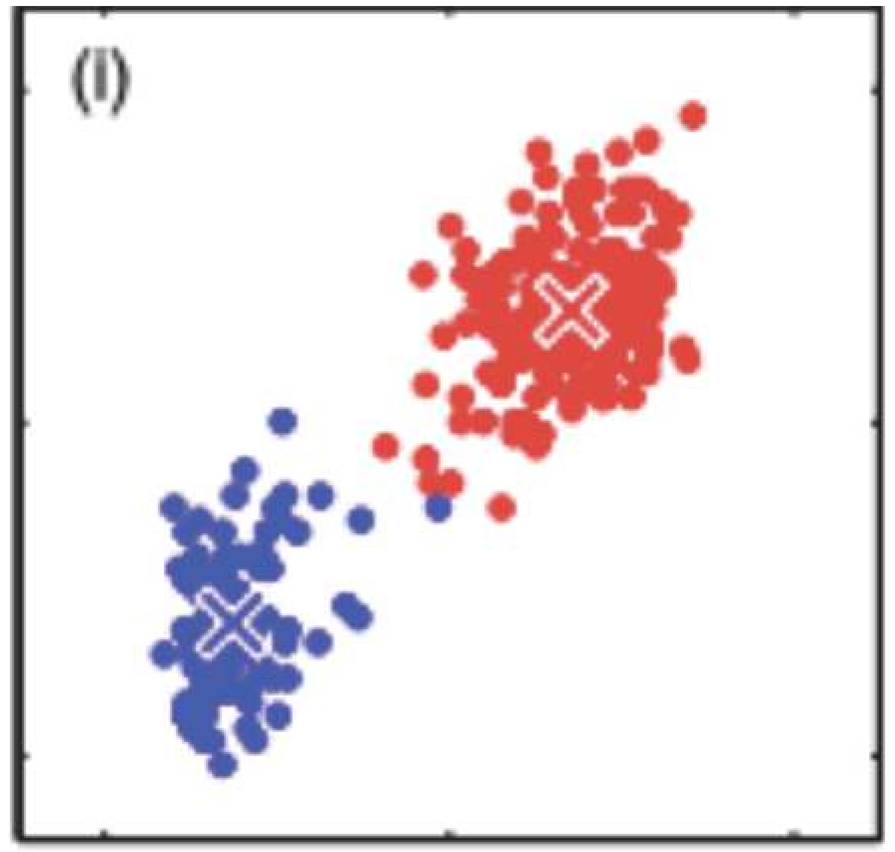
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

M-step (maximisation)
Compute centroid for each cluster

until μ_k converges

Example adapted from Fig 9.1, 9,2 - Bishop

K-means algorithm



Initialise $\mu_k, k = 1, \ldots, K$

Repeat

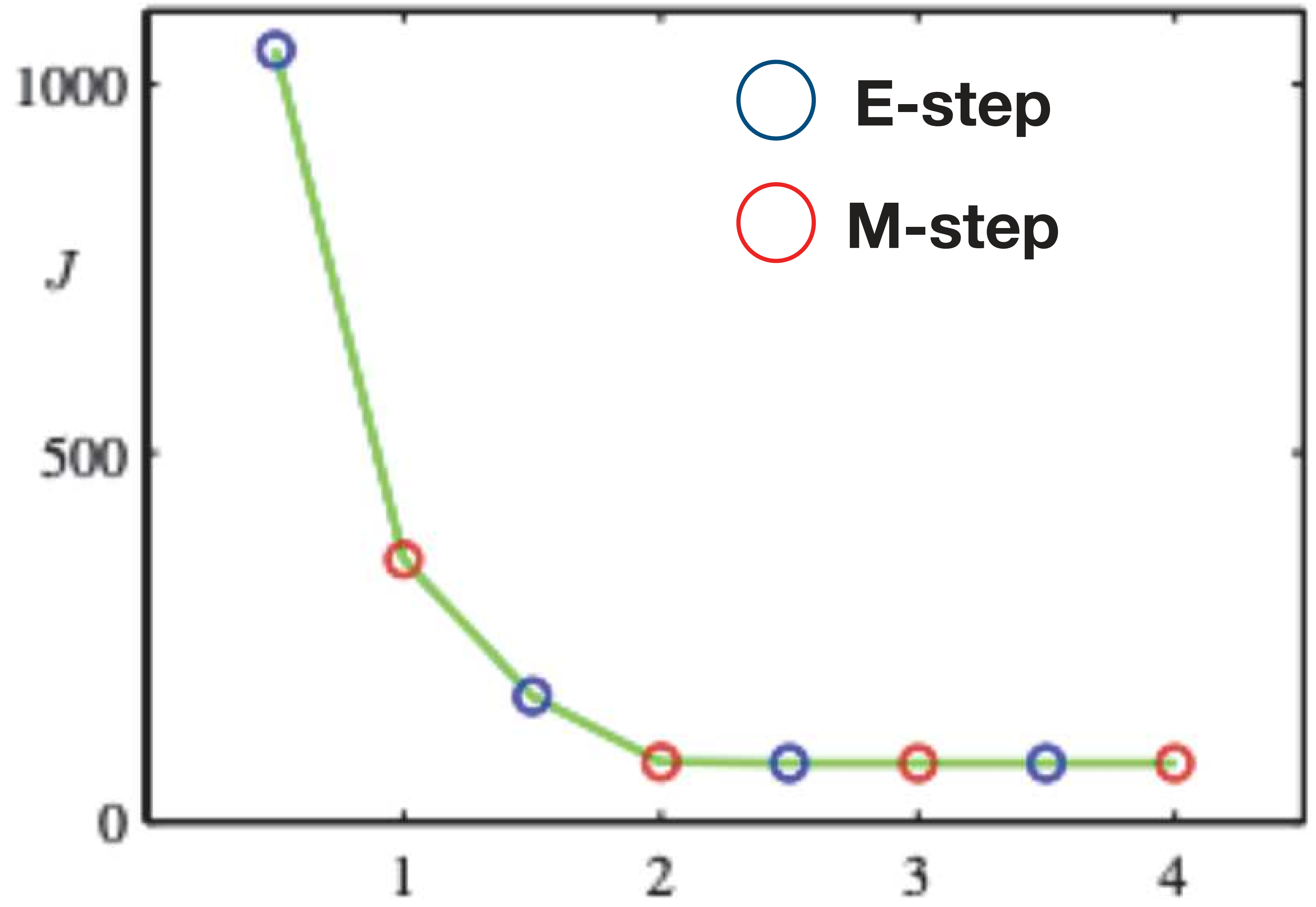
for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

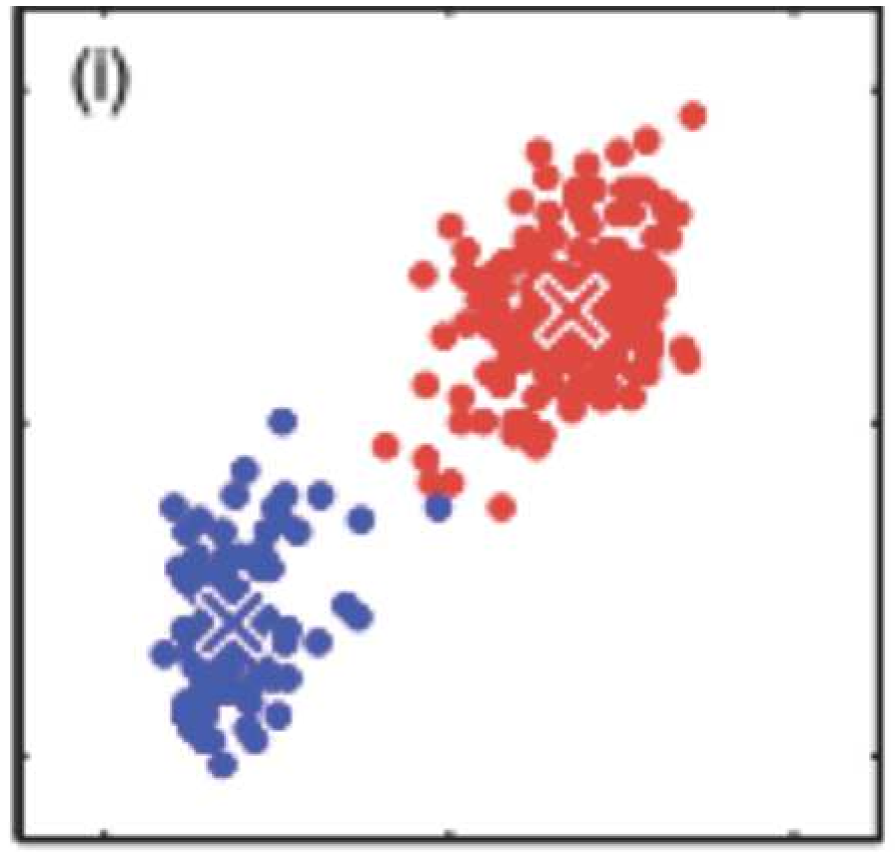
for each μ_k:

$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

until μ_k converges



K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

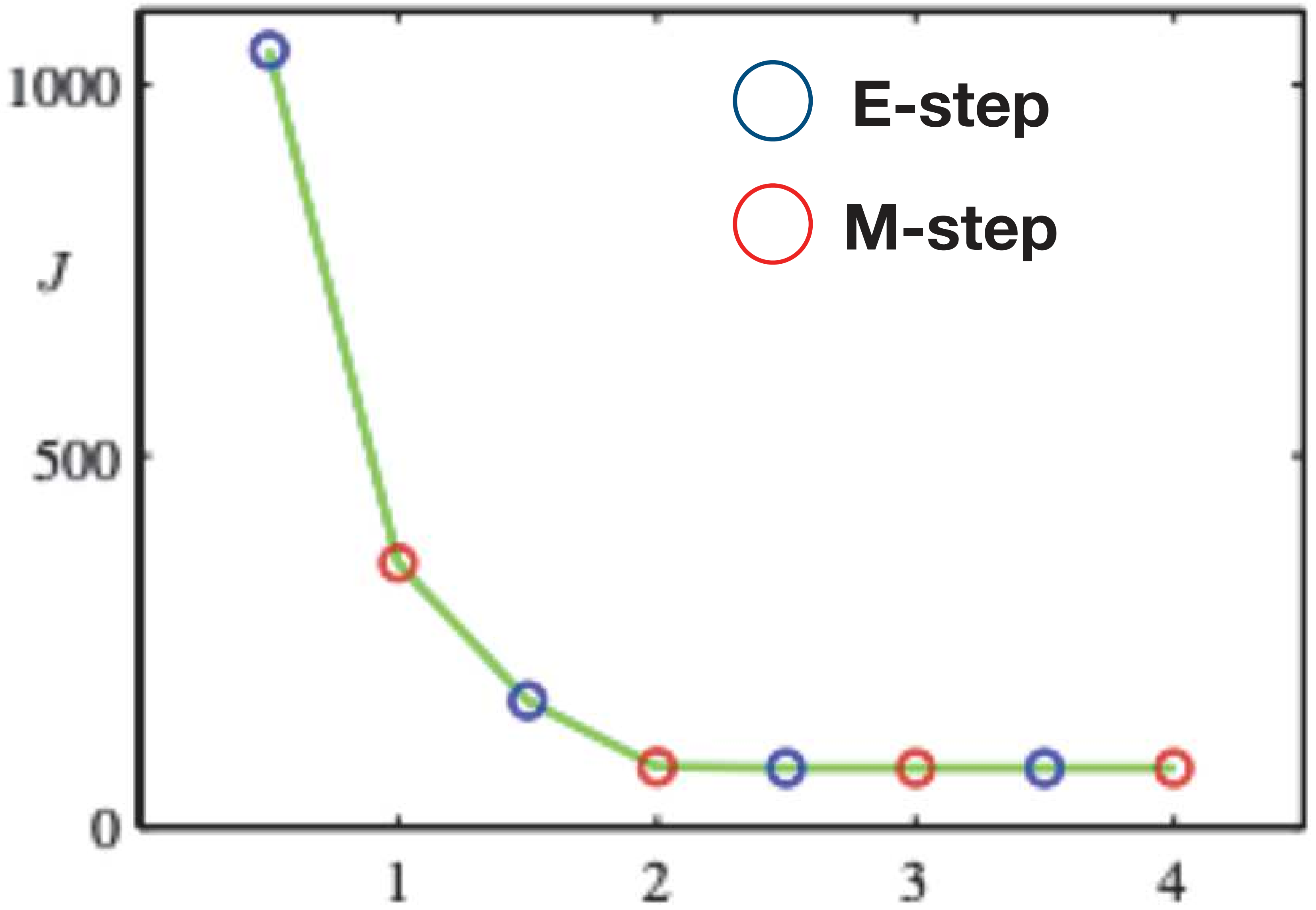
for each $\mathbf{x}_i$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

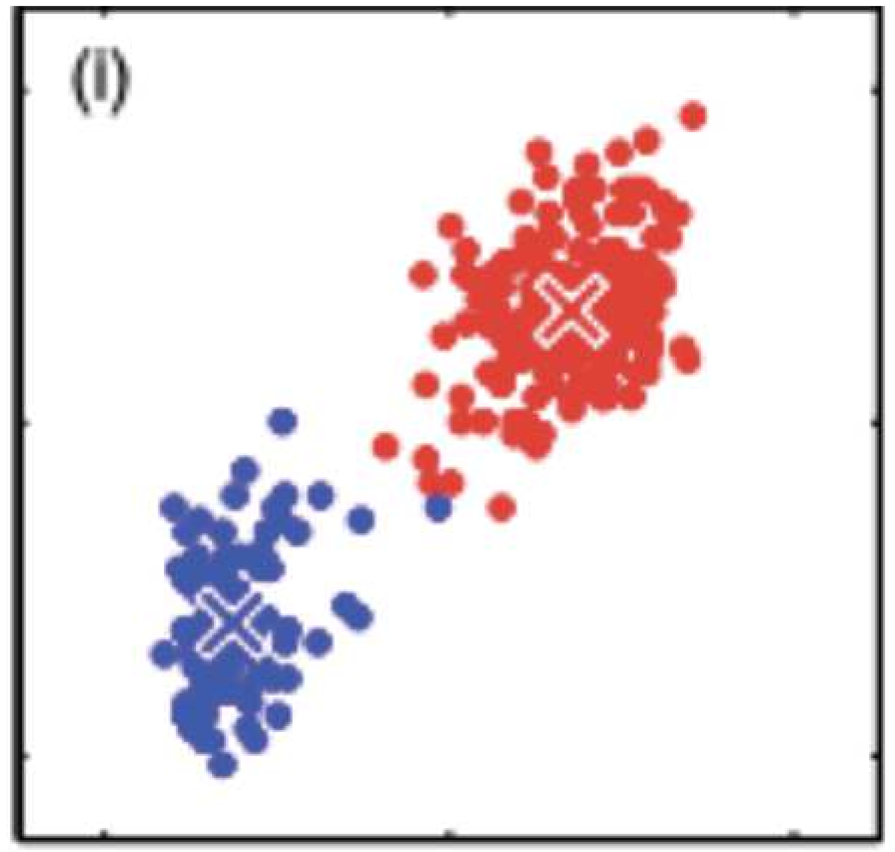
for each μ_k:

$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

- until μ_k converges



K-means algorithm



Initialise $\mu_k, k = 1,\ldots,K$

Repeat

for each $\mathbf{x}_i$:

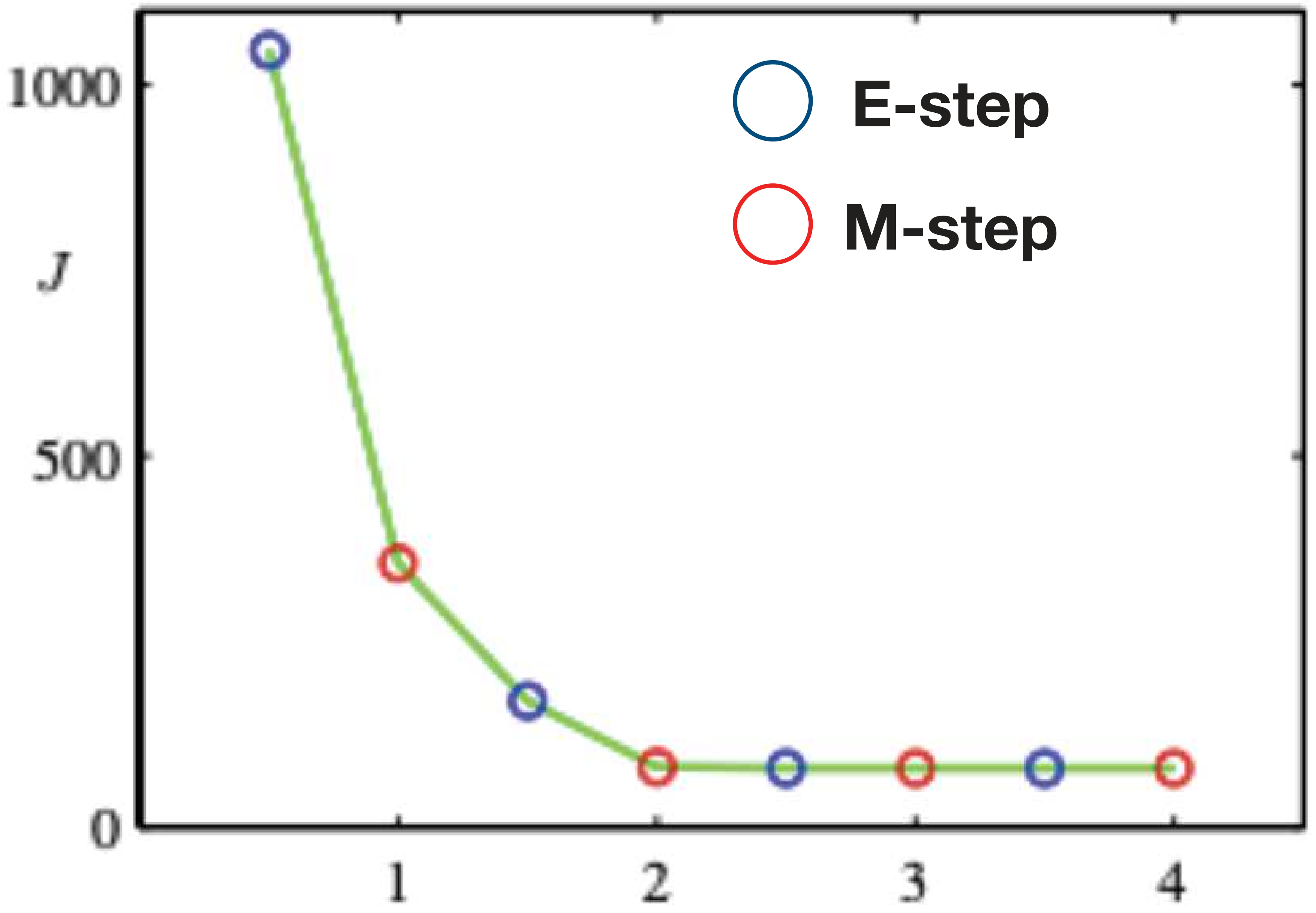
$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

for each μ_k:

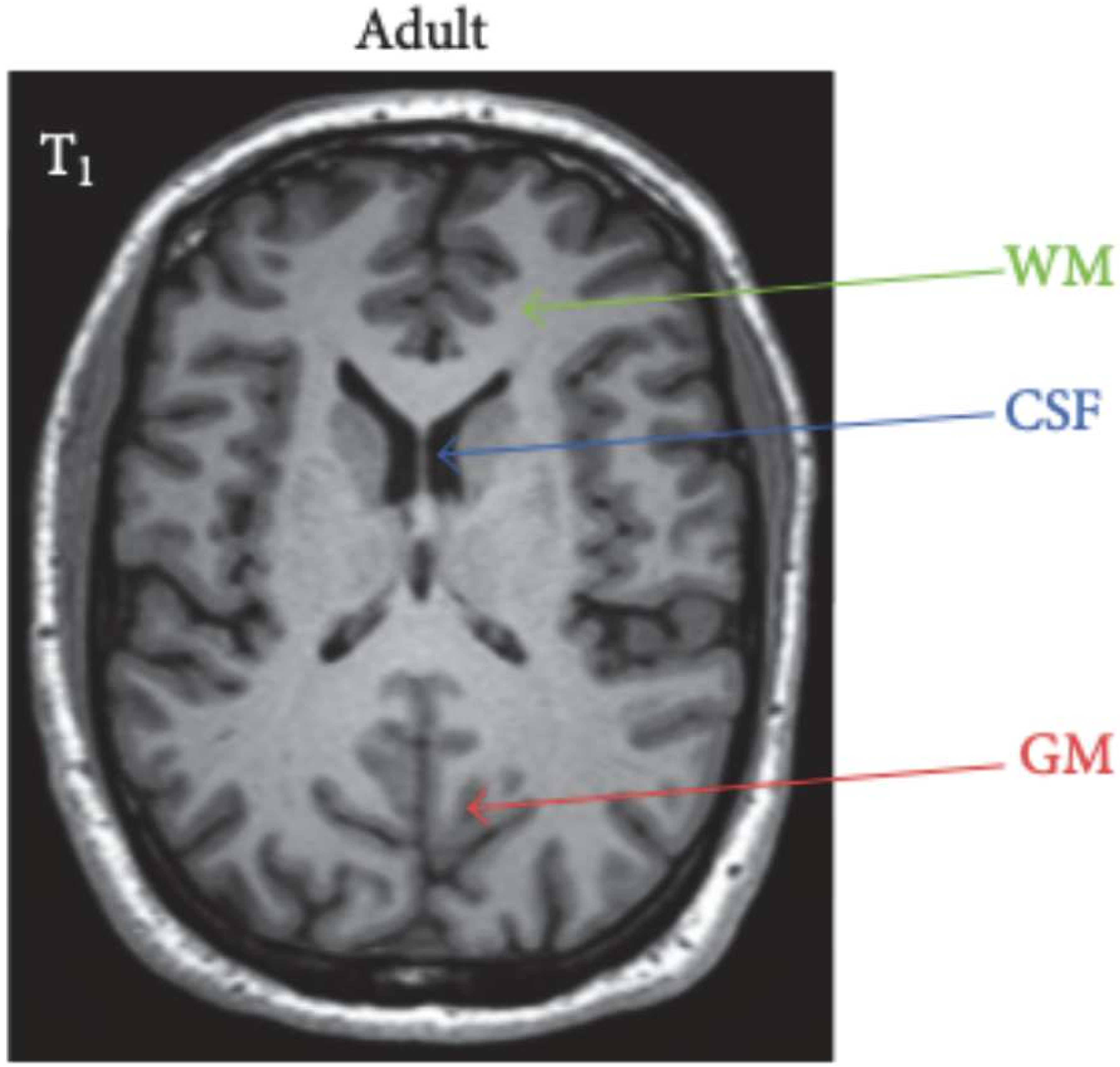
$$\mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

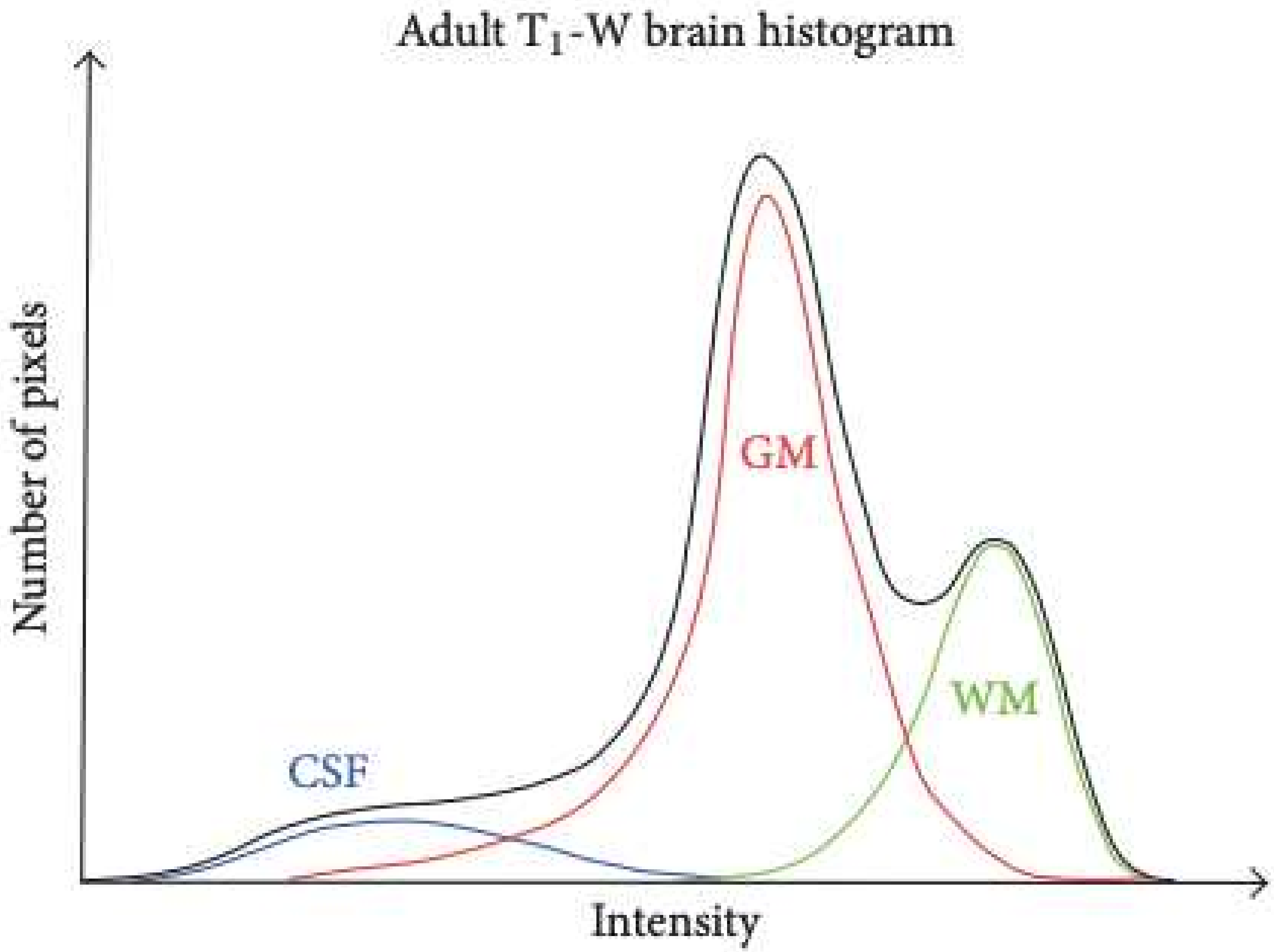
until μ_k converges

Algorithm complexity?



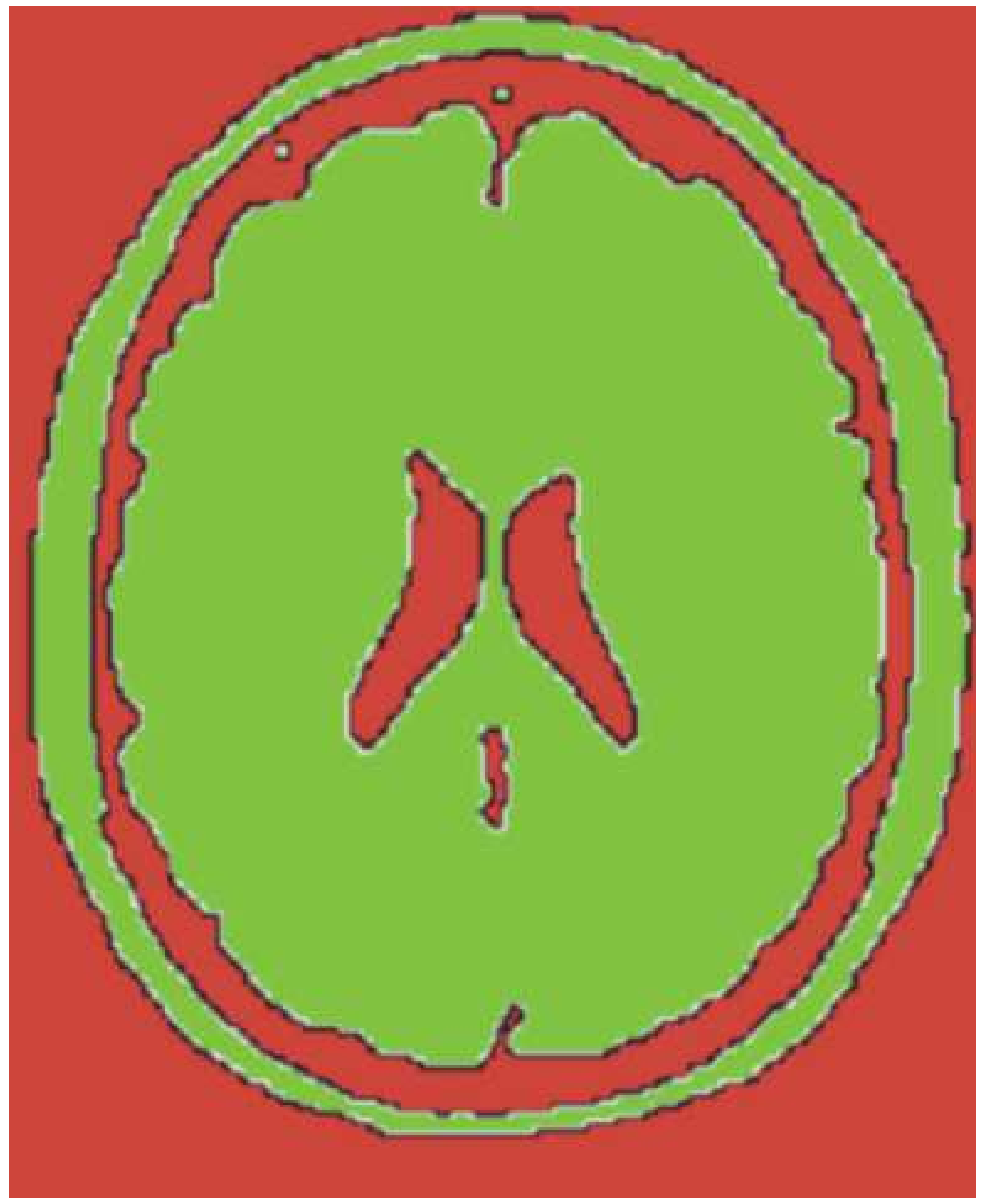
Real example: Brain segmentation





Despotovic et al. **MRI Segmentation of the Human Brain: Challenges, Methods, and Applications.** CMMM 2015

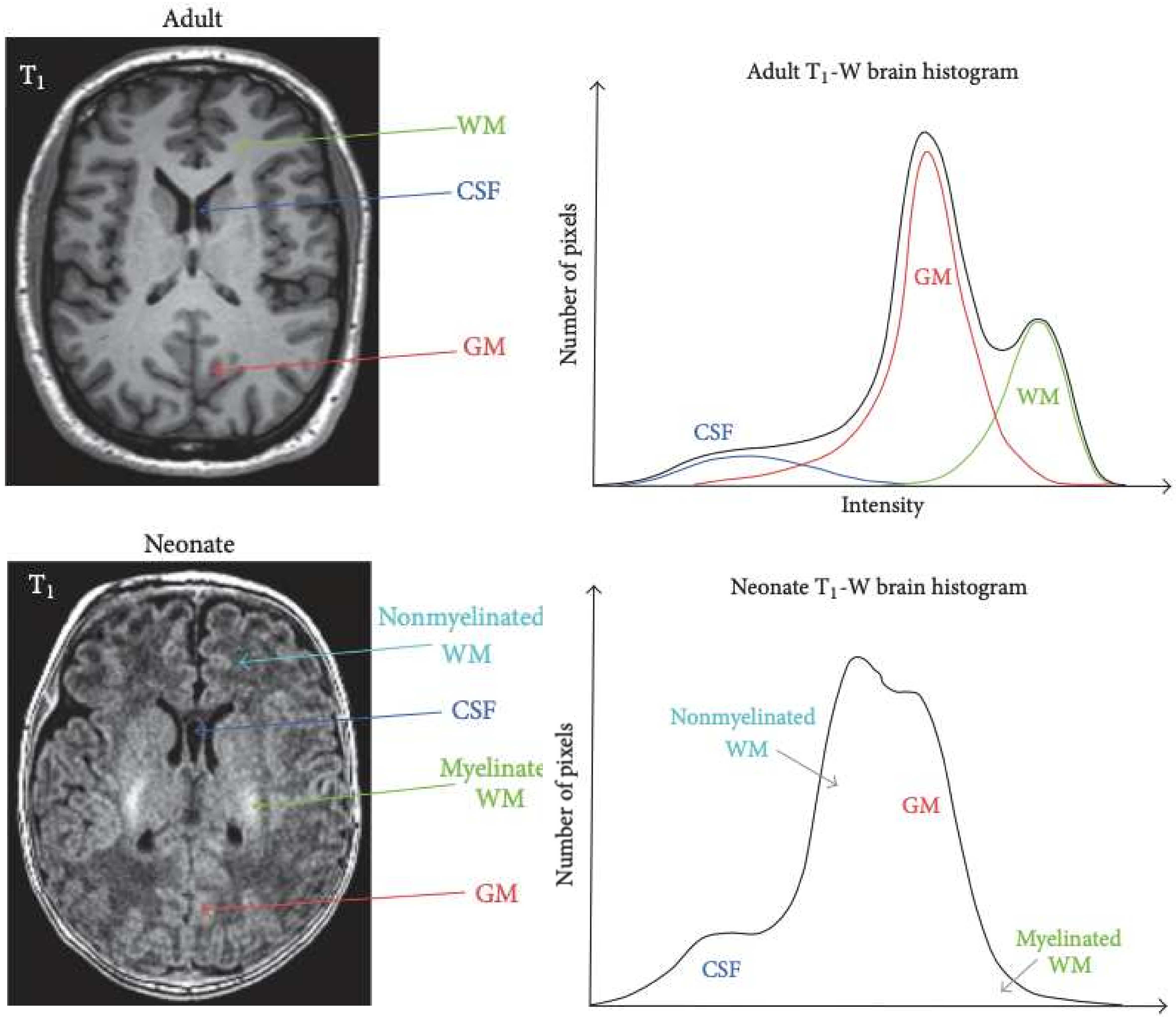
Real example



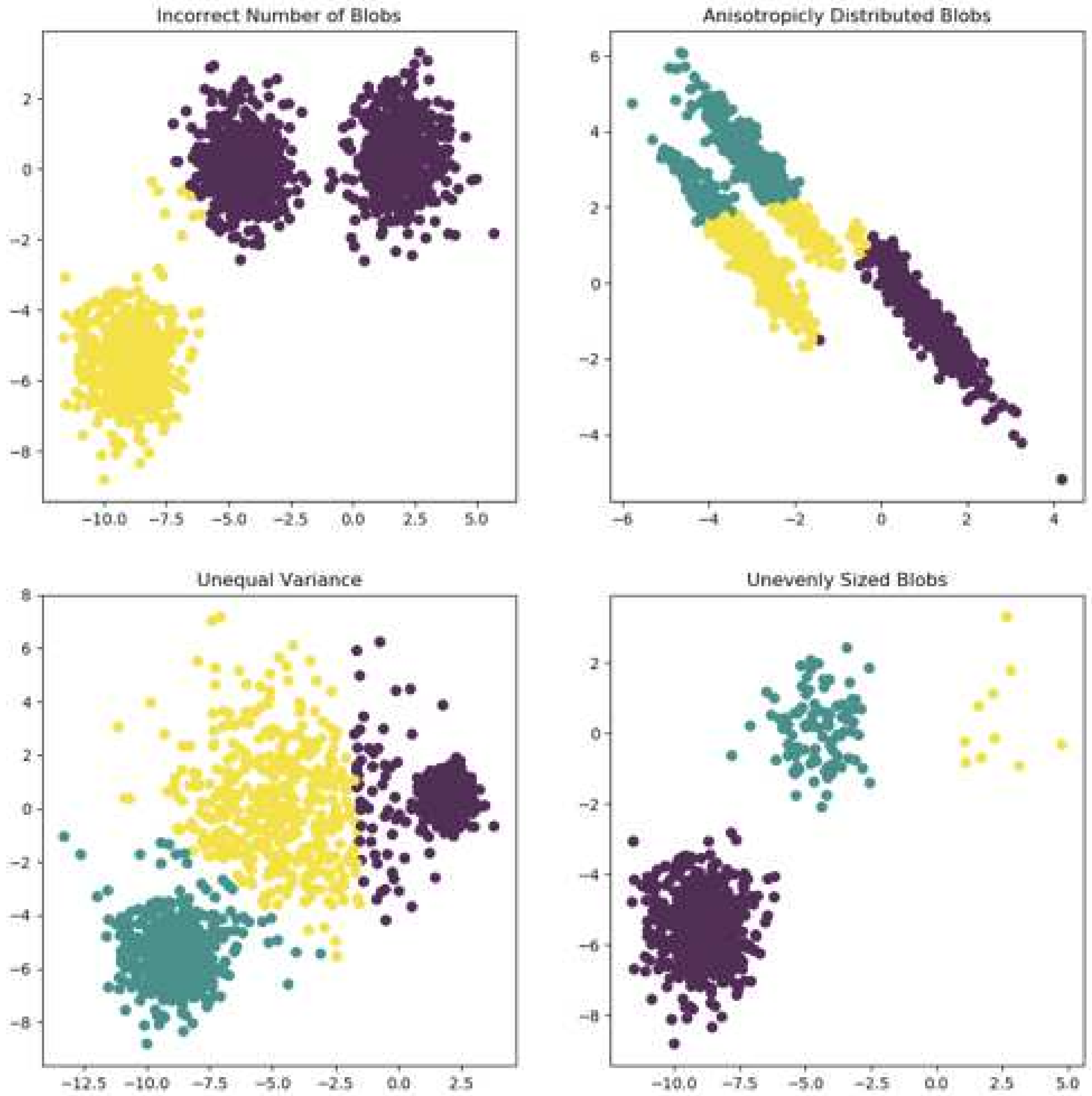
K= 2

K= 3

Limitations

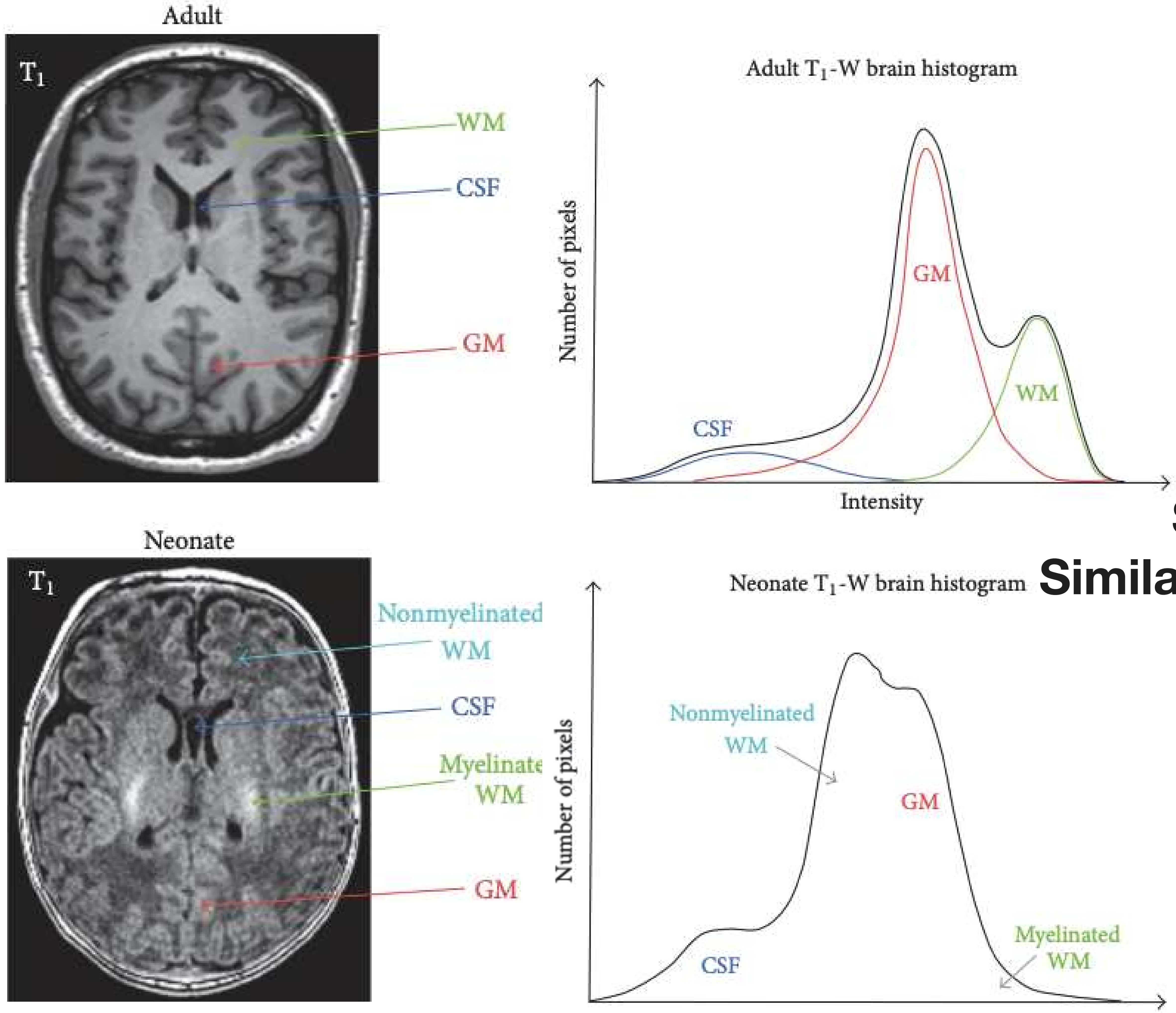


Despotovic et al. CMMM 2015



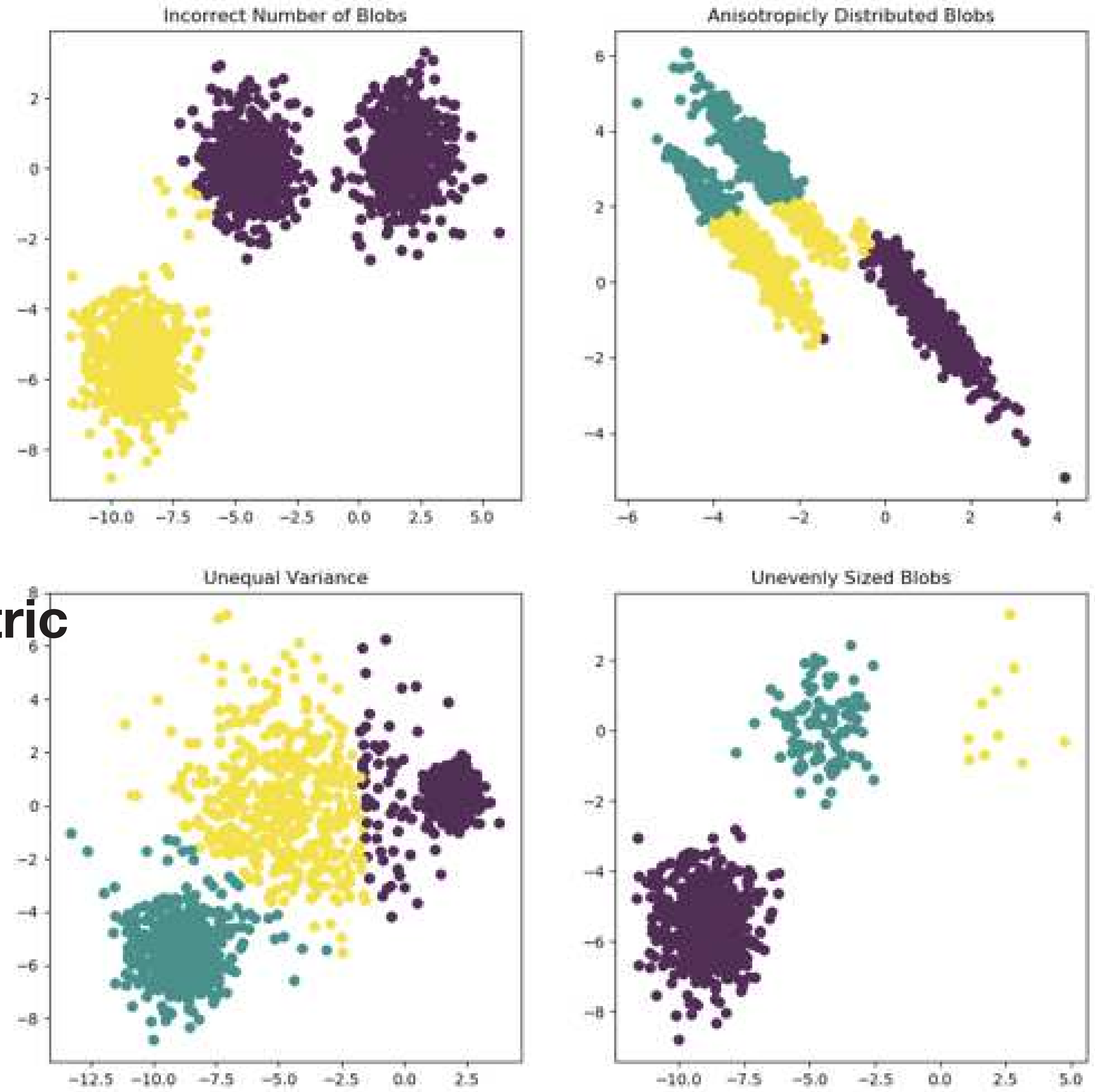
Source: scikit-learn

Limitations



Slow

Similarity metric



Despotovic et al. CMMM 2015

Source: scikit-learn

Further reading

- AK. Jain. Data clustering: 50 years beyond K-means. Pattern Recognition Letters. 2010
- L. Kaufman & P. Rosseuw. Clustering by means of Medoids. Statistical Data Analysis Based on the –Norm and Related Methods. 1987
- J. Newling & F. Fleuret. K-medoids for K-means seeding. NIPS 2017
- S. Sohely-Kahn et al. Generalized k-means-based clustering for temporal data under weighted and kernel time warp. Pattern Recognition Letters. 2016

This week

- Exercises 9.1, 9.2 Bishop
- Implement own version of K-means (no scikit-learn, no ITK)
  - Apply it to brain image segmentation
  - Experiment with different initialisation strategies
  - Bonus: Computationally efficient implementation

Next session: Gaussian mixtures & Expectation maximisation algorithm