Big Data Computing

Master's Degree in Computer Science

2024-2025

SAPIENZA
UNIVERSITÀ DI ROMA

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

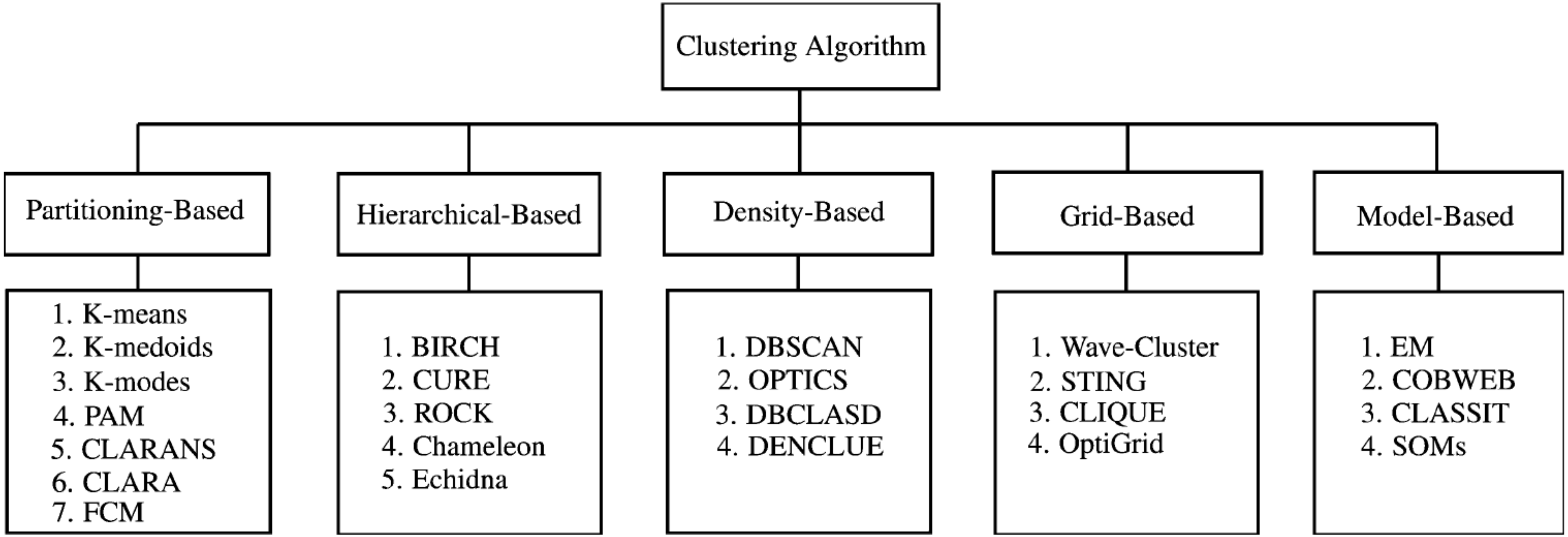
tolomei@di.uniroma1.it

Recap from Last Lecture(s)

- Clustering is an unsupervised learning technique to group "similar" data objects together
- Depends on:
  - **object representation**
  - **similarity measure**
- Harder when data dimensionality gets large (**curse of dimensionality**)
- Number of output clusters is part of the problem itself!

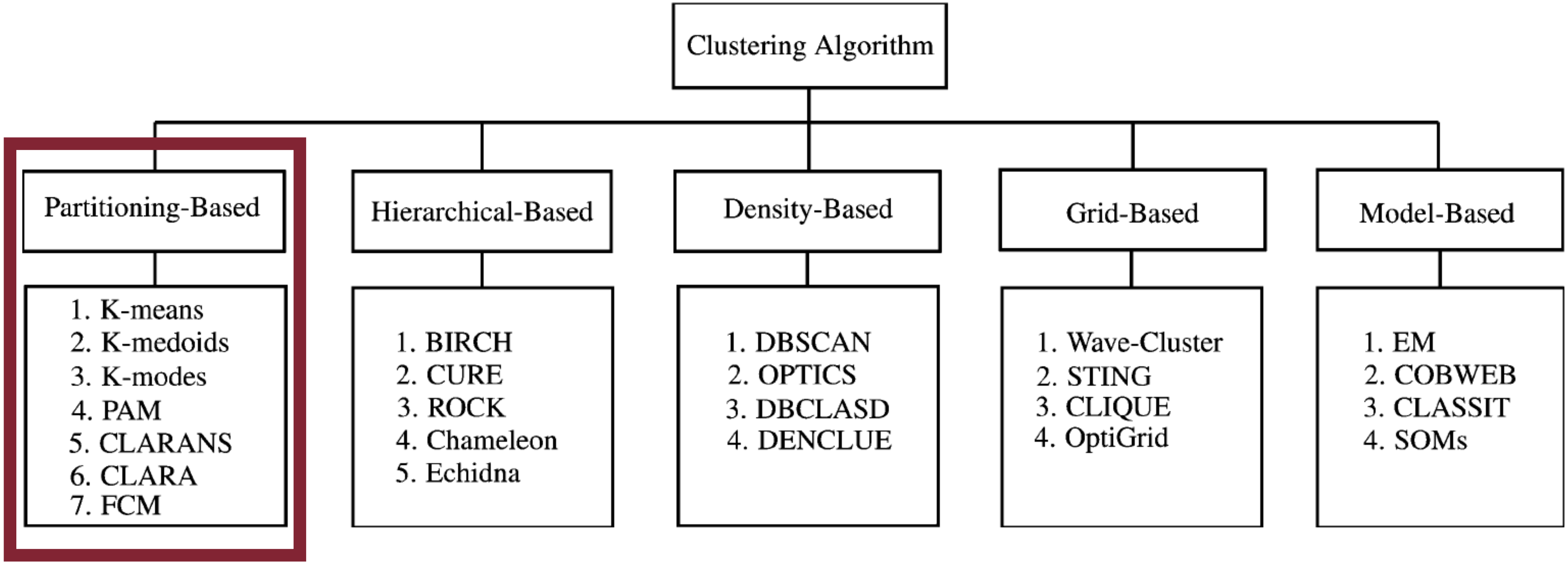
Clustering Algorithms

Clustering Algorithms: Taxonomy



source: https://www.computer.org/csdl/journal/ec/2014/03/06832486/13rRUEgs2xB

Clustering Algorithms: Taxonomy



source: https://www.computer.org/csdl/journal/ec/2014/03/06832486/13rRUEgs2xB

Partitioning: Hard Clustering

- **Input:** A set of N data points and a number K (K < N)

Partitioning: Hard Clustering

- **Input:** A set of N data points and a number K (K < N)
- **Output:** A partition of the N data points into K clusters

Partitioning: Hard Clustering

- **Input:** A set of N data points and a number K (K < N)
- **Output:** A partition of the N data points into K clusters
- **Goal:** Find the partition which optimizes a certain criterion

Partitioning: Intuition

Let K=2 for simplicity

Partitioning: Intuition

Let K=2 for simplicity

We can assign each of the N data points to **either** one of the K=2 clusters

Partitioning: Intuition

Let K=2 for simplicity

We can assign each of the N data points to **either** one of the K=2 clusters

We use an N-dimensional assignment vector C, where C[i] = {0, 1}

Partitioning: Intuition

Let K=2 for simplicity

We can assign each of the N data points to **either** one of the K=2 clusters

We use an N-dimensional assignment vector C, where C[i] = {0, 1}

Here is a possible assignment (i.e., clustering output):

	0	1	2			…				N-1
C	0	1	1	…	0	0	1	…	0	1

Partitioning: Intuition

Let K=2 for simplicity

We can assign each of the N data points to **either** one of the K=2 clusters

We use an N-dimensional assignment vector C, where C[i] = {0, 1}

Here is another one:

	0	1	2		…					N-1
C	1	1	0	…	0	0	1	…	1	1

Partitioning: Intuition

Let K=2 for simplicity

We can assign each of the N data points to **either** one of the K=2 clusters

We use an N-dimensional assignment vector C, where C[i] = {0, 1}

… And another one:

	0	1	2			…				N-1
C	1	1	0	…	0	1	1	…	1	0

Partitioning: Intuition

So, how many possible clustering outputs?

Partitioning: Intuition

So, how many possible clustering outputs?

Roughly, 2^N; More generally, K^N

0	1	2		…					N-1
{0..K-1}	{0..K-1}	{0..K-1}	…	{0..K-1}	{0..K-1}	{0..K-1}	…	{0..K-1}	{0..K-1}

Partitioning: Intuition

So, how many possible clustering outputs?

Roughly, 2^N; More generally, K^N

0	1	2		...					N-1
{0..K-1}	{0..K-1}	{0..K-1}	...	{0..K-1}	{0..K-1}	{0..K-1}	...	{0..K-1}	{0..K-1}

Actually, **slightly less** than that because we want each of the K clusters to contain at least one data point!

Partitioning: Intuition

So, how many possible clustering outputs?

Roughly, 2^N; More generally, K^N

0	1	2		...					N-1
{0..K-1}	{0..K-1}	{0..K-1}	...	{0..K-1}	{0..K-1}	{0..K-1}	...	{0..K-1}	{0..K-1}

Actually, **slightly less** than that because we want each of the K clusters to contain at least one data point!

In the previous example (K=2), the following is **not** allowed

0	1	2			...				N-1
0	0	0	...	0	0	0	...	0	0

0	1	2			...				N-1
1	1	1	...	1	1	1	...	1	1

Partitioning: Intuition

So, how many possible clustering outputs?

Roughly, 2^N; More generally, K^N

0	1	2		...					N-1
{0..K-1}	{0..K-1}	{0..K-1}	...	{0..K-1}	{0..K-1}	{0..K-1}	...	{0..K-1}	{0..K-1}

Actually, **slightly less** than that because we want each of the K clusters to contain at least one data point!

In the previous example (K=2), the following is **not** allowed

0	1	2			...				N-1
0	0	0	...	0	0	0	...	0	0

0	1	2			...				N-1
1	1	1	...	1	1	1	...	1	1

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Let K=2 be the desired number of output clusters

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Let K=2 be the desired number of output clusters

{{1}, {2,3,4,5}} → (0,1,1,1,1)
{{2}, {1,3,4,5}} → (1,0,1,1,1)
…
{{5}, {1,2,3,4}} → (1,1,1,1,0)

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Let K=2 be the desired number of output clusters

{{1}, {2,3,4,5}} → (0,1,1,1,1)
{{2}, {1,3,4,5}} → (1,0,1,1,1)
…
{{5}, {1,2,3,4}} → (1,1,1,1,0)

{{1,2}, {3,4,5}} → (0,0,1,1,1)
{{1,3}, {2,4,5}} → (0,1,0,1,1)
…
{{1,5}, {2,3,4}} → (0,1,1,1,0)

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Let K=2 be the desired number of output clusters

{{1}, {2,3,4,5}} → (0,1,1,1,1)	{{1,2}, {3,4,5}} → (0,0,1,1,1)	{{2,1}, {3,4,5}} → (0,0,1,1,1)
{{2}, {1,3,4,5}} → (1,0,1,1,1)	{{1,3}, {2,4,5}} → (0,1,0,1,1)	{{3,1}, {2,4,5}} → (0,1,0,1,1)
…	…	…
{{5}, {1,2,3,4}} → (1,1,1,1,0)	{{1,5}, {2,3,4}} → (0,1,1,1,0)	{{5,1}, {2,3,4}} → (0,1,1,1,0)

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Let K=2 be the desired number of output clusters

{{1}, {2,3,4,5}} → (0,1,1,1,1)	{{1,2}, {3,4,5}} → (0,0,1,1,1)	{{2,1}, {3,4,5}} → (0,0,1,1,1)
{{2}, {1,3,4,5}} → (1,0,1,1,1)	{{1,3}, {2,4,5}} → (0,1,0,1,1)	{{3,1}, {2,4,5}} → (0,1,0,1,1)
…	…	…
{{5}, {1,2,3,4}} → (1,1,1,1,0)	{{1,5}, {2,3,4}} → (0,1,1,1,0)	{{5,1}, {2,3,4}} → (0,1,1,1,0)

Some combinations are counted twice!

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Let K=2 be the desired number of output clusters

{{1}, {2,3,4,5}} → (0,1,1,1,1)	{{1,2}, {3,4,5}} → (0,0,1,1,1)	{{2,1}, {3,4,5}} → (0,0,1,1,1)
{{2}, {1,3,4,5}} → (1,0,1,1,1)	**{{1,3}, {2,4,5}} → (0,1,0,1,1)**	**{{3,1}, {2,4,5}} → (0,1,0,1,1)**
…	…	…
{{5}, {1,2,3,4}} → (1,1,1,1,0)	{{1,5}, {2,3,4}} → (0,1,1,1,0)	{{5,1}, {2,3,4}} → (0,1,1,1,0)

Some combinations are counted twice!

Partitioning: Intuition

Let D = {1, 2, 3, 4, 5} a set of N=5 data points

Let K=2 be the desired number of output clusters

{{1}, {2,3,4,5}} → (0,1,1,1,1)	{{1,2}, {3,4,5}} → (0,0,1,1,1)	{{2,1}, {3,4,5}} → (0,0,1,1,1)
{{2}, {1,3,4,5}} → (1,0,1,1,1)	{{1,3}, {2,4,5}} → (0,1,0,1,1)	{{3,1}, {2,4,5}} → (0,1,0,1,1)
…	…	…
{{5}, {1,2,3,4}} → (1,1,1,1,0)	{{1,5}, {2,3,4}} → (0,1,1,1,0)	{{5,1}, {2,3,4}} → (0,1,1,1,0)

Some combinations are counted twice!

Partitioning: NP-Hardness

- **Stirling Partition Number** → K-way non-empty partitions of N elements

Partitioning: NP-Hardness

- **Stirling Partition Number** → K-way non-empty partitions of N elements

$$S(K, N) \sim K^N/K! = O(K^N)$$

Partitioning: NP-Hardness

- **Stirling Partition Number** → K-way non-empty partitions of N elements

$$S(K, N) \sim K^N/K! = O(K^N)$$

- Finding the **global optimum** → Intractable for many objective function (enumerate all the possible partitions)*

*Kleinberg, J., "An Impossibility Theorem for Clustering" (NIPS 2002)

Partitioning: NP-Hardness

- **Stirling Partition Number** → K-way non-empty partitions of N elements

$$S(K, N) \sim K^N/K! = O(K^N)$$

- Finding the **global optimum** → Intractable for many objective function (enumerate all the possible partitions)*
- Effective heuristics → K-means, K-medoids, K-means++, etc.

*Kleinberg, J., "An Impossibility Theorem for Clustering" (NIPS 2002)

Flat Hard Clustering: General Framework

$\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ the set of N input data points

$\{C_1, \ldots, C_K\}$ the set of K output clusters

C_k the generic k-th cluster

$\boldsymbol{\theta}_k$ is the *representative* of cluster C_k

Flat Hard Clustering: General Framework

$\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ the set of N input data points

$\{C_1, \ldots, C_K\}$ the set of K output clusters

C_k the generic k-th cluster

$\boldsymbol{\theta}_k$ is the *representative* of cluster C_k

Note:

At this stage we haven't yet specified what a cluster representative actually is

Objective Function

$$L(A, \boldsymbol{\Theta}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$$

where:

- A is an $N \times K$ matrix s.t. $\alpha_{n,k} = 1$ iff $\mathbf{x}_n$ is assigned to cluster C_k, 0 otherwise
- $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ are the cluster representatives
- $\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$ is a function measuring the distance between $\mathbf{x}_n$ and $\boldsymbol{\theta}_k$

Objective Function

$$L(A, \mathbf{\Theta}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$$

$\forall n \; \exists! k$ such that $\alpha_{n,k} = 1 \; \wedge \; \alpha_{n,k'} = 0 \; \forall k' \neq k$

hard clustering

where:

- A is an $N \times K$ matrix s.t. $\alpha_{n,k} = 1$ iff $\mathbf{x}_n$ is assigned to cluster C_k, 0 otherwise
- $\mathbf{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ are the cluster representatives
- $\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$ is a function measuring the distance between $\mathbf{x}_n$ and $\boldsymbol{\theta}_k$

Objective Function

$$L(A, \boldsymbol{\Theta}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$$

$\forall n\ \exists! k$ such that $\alpha_{n,k} = 1\ \wedge\ \alpha_{n,k'} = 0\ \forall k' \neq k$

hard clustering

where:

- A is an $N \times K$ matrix s.t. $\alpha_{n,k} = 1$ iff $\mathbf{x}_n$ is assigned to cluster C_k, 0 otherwise
- $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ are the cluster representatives
- $\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$ is a function measuring the distance between $\mathbf{x}_n$ and $\boldsymbol{\theta}_k$

$$\boxed{A^*, \boldsymbol{\Theta}^* = \operatorname{argmin}_{A,\boldsymbol{\Theta}} \underbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}_{L(A,\boldsymbol{\Theta})}}$$

Objective Function

$$\boxed{A^*, \boldsymbol{\Theta}^* = \text{argmin}_{A,\boldsymbol{\Theta}} \underbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}_{L(A,\boldsymbol{\Theta})}}$$

Objective Function

$$A^*, \boldsymbol{\Theta}^* = \text{argmin}_{A,\boldsymbol{\Theta}} \underbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}_{L(A,\boldsymbol{\Theta})}$$

exact solution must explore exponential search space $S(K, N) \sim O(K^N)$

$\Rightarrow$ NP-hard

Objective Function

$$\boxed{A^*, \boldsymbol{\Theta}^* = \mathrm{argmin}_{A,\boldsymbol{\Theta}} \underbrace{\sum_{n=1}^{N}\sum_{k=1}^{K} \alpha_{n,k}\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}_{L(A,\boldsymbol{\Theta})}}$$

exact solution must explore exponential search space $S(K, N) \sim O(K^N)$ ⇒ NP-hard

non-convex due to the discrete assignment matrix A ⇒ multiple local minima

Iterative Solution: Lloyd-Forgy Algorithm

- **NP-hardness** doesn't allow us to compute the exact solution (i.e., global optimum)

Iterative Solution: Lloyd-Forgy Algorithm

- **NP-hardness** doesn't allow us to compute the exact solution (i.e., global optimum)
- **Non-convexity** doesn't allow us to rely on nice property of convex optimization (unique global optimum)

Iterative Solution: Lloyd-Forgy Algorithm

- **NP-hardness** doesn't allow us to compute the exact solution (i.e., global optimum)
- **Non-convexity** doesn't allow us to rely on nice property of convex optimization (unique global optimum)
- A convex objective can be (approximately) solved with numerical methods to find the global optimum

Iterative Solution: Lloyd-Forgy Algorithm

- **Lloyd-Forgy Algorithm:** 2-step **iterative** approximated solution

Iterative Solution: Lloyd-Forgy Algorithm

- **Lloyd-Forgy Algorithm:** 2-step **iterative** approximated solution
- Assignment step
- Update step

Iterative Solution: Lloyd-Forgy Algorithm

- **Lloyd-Forgy Algorithm:** 2-step **iterative** approximated solution
- Assignment step
- Update step

Does not guarantee to find the global optimum as it may stuck to a local optimum or a saddle point

2-Step Optimization: Assignment Step

$\boxed{\text{Minimize L w.r.t. A by fixing } \boldsymbol{\Theta}}$

$L(A|\boldsymbol{\Theta}) = L(A;\boldsymbol{\Theta}) = L$ is a function of A parametrized by $\boldsymbol{\Theta}$

2-Step Optimization: Assignment Step

Minimize L w.r.t. A by fixing $\boldsymbol{\Theta}$

$L(A|\boldsymbol{\Theta}) = L(A;\boldsymbol{\Theta}) = L$ is a function of A parametrized by $\boldsymbol{\Theta}$

Note:
Can't take the gradient of L w.r.t. A since A is discrete!

2-Step Optimization: Assignment Step

$\boxed{\text{Minimize L w.r.t. A by fixing } \boldsymbol{\Theta}}$

$$L(A|\boldsymbol{\Theta}) = L(A;\boldsymbol{\Theta}) = L \text{ is a function of } A \text{ parametrized by } \boldsymbol{\Theta}$$

Intuitively, given a set of fixed representatives, L is minimized if each data point is assigned to the closest cluster representative according to δ
(L is the sum of all the distances from each data point to its representative)

2-Step Optimization: Assignment Step

Minimize L w.r.t. A by fixing $\boldsymbol{\Theta}$

$L(A|\boldsymbol{\Theta}) = L(A;\boldsymbol{\Theta}) = L$ is a function of A parametrized by $\boldsymbol{\Theta}$

Intuitively, given a set of fixed representatives, L is minimized if each data point is assigned to the closest cluster representative according to δ

(L is the sum of all the distances from each data point to its representative)

$$\alpha_{n,k} = \begin{cases} 1 & \text{if } \delta(\mathbf{x}_n, \boldsymbol{\theta}_k) = \min_{1 \leq j \leq K}\{\delta(\mathbf{x}_n, \boldsymbol{\theta}_j)\} \\ 0 & \text{otherwise} \end{cases}$$

2-Step Optimization: Update Step

$\boxed{\text{Minimize L w.r.t. } \boldsymbol{\Theta} \text{ by fixing A}}$

$$L(\boldsymbol{\Theta}|A) = L(\boldsymbol{\Theta}; A) = L \text{ is a function of } \boldsymbol{\Theta} \text{ parametrized by } A$$

2-Step Optimization: Update Step

$\boxed{\text{Minimize L w.r.t. } \boldsymbol{\Theta} \text{ by fixing A}}$

$$L(\boldsymbol{\Theta}|A) = L(\boldsymbol{\Theta}; A) = L \text{ is a function of } \boldsymbol{\Theta} \text{ parametrized by } A$$

We can minimize L by taking the gradient of L w.r.t $\boldsymbol{\Theta}$ (i.e., the vector of partial derivatives), set it to 0 and solve it for $\boldsymbol{\Theta}$

2-Step Optimization: Update Step

$$\nabla L(\boldsymbol{\Theta}; A) = \left(\frac{\partial L(\boldsymbol{\Theta}; A)}{\partial \boldsymbol{\theta}_1}, \ldots, \frac{\partial L(\boldsymbol{\Theta}; A)}{\partial \boldsymbol{\theta}_K} \right)$$

2-Step Optimization: Update Step

$$\nabla L(\mathbf{\Theta}; A) = \left(\frac{\partial L(\mathbf{\Theta}; A)}{\partial \boldsymbol{\theta}_1}, \ldots, \frac{\partial L(\mathbf{\Theta}; A)}{\partial \boldsymbol{\theta}_K} \right)$$

$$\nabla L(\mathbf{\Theta}; A) = \left(\frac{\partial L(\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_1}, \ldots, \frac{\partial L(\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_K} \right)$$

2-Step Optimization: Update Step

$$\nabla L(\boldsymbol{\Theta}; A) = \left(\frac{\partial L(\boldsymbol{\Theta}; A)}{\partial \boldsymbol{\theta}_1}, \ldots, \frac{\partial L(\boldsymbol{\Theta}; A)}{\partial \boldsymbol{\theta}_K} \right)$$

$$\nabla L(\boldsymbol{\Theta}; A) = \left(\frac{\partial L(\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_1}, \ldots, \frac{\partial L(\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_K} \right)$$

$$\boxed{\frac{\partial L(\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_j}}$$

The general j-th partial derivative

2-Step Optimization: Update Step

$$\nabla L(\boldsymbol{\Theta}; A) = \mathbf{0} \Leftrightarrow \frac{\partial L(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_j} = 0 \; \forall j \in \{1, \ldots, K\}$$

2-Step Optimization: Update Step

$$\nabla L(\boldsymbol{\Theta}; A) = \mathbf{0} \Leftrightarrow \frac{\partial L(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_j} = 0 \; \forall j \in \{1, \ldots, K\}$$

$$\frac{\partial L(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_j} = \boxed{\frac{\partial}{\partial \boldsymbol{\theta}_j} \left[\sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k) \right]}$$

2-Step Optimization: Update Step

$$\nabla L(\boldsymbol{\Theta}; A) = \mathbf{0} \Leftrightarrow \frac{\partial L(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_j} = 0 \; \forall j \in \{1, \ldots, K\}$$

$$\frac{\partial L(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K; A)}{\partial \boldsymbol{\theta}_j} = \boxed{\frac{\partial}{\partial \boldsymbol{\theta}_j} \left[\sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k) \right]}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_j}$$

To make the notation easier!

2-Step Optimization: Update Step

$$\frac{\partial L}{\partial \boldsymbol{\theta}_j} = \frac{\partial}{\partial \boldsymbol{\theta}_j}\left[\sum_{n=1}^{N}\sum_{k=1}^{K}\alpha_{n,k}\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)\right] = 0$$

2-Step Optimization: Update Step

$$\frac{\partial L}{\partial \boldsymbol{\theta}_j} = \frac{\partial}{\partial \boldsymbol{\theta}_j}\left[\sum_{n=1}^{N} \boxed{\sum_{k=1}^{K}} \alpha_{n,k}\,\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)\right] = 0$$

When computing the partial derivative w.r.t. $\boldsymbol{\theta}_j$ any other term $\boldsymbol{\theta}_k$ of the inner summation is treated as constant!

2-Step Optimization: Update Step

$$\frac{\partial L}{\partial \boldsymbol{\theta}_j} = \frac{\partial}{\partial \boldsymbol{\theta}_j} \left[\sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k) \right] = 0$$

$$\boxed{= \frac{\partial}{\partial \boldsymbol{\theta}_j} \left[\sum_{n=1}^{N} \alpha_{n,j} \delta(\mathbf{x}_n, \boldsymbol{\theta}_j) \right] = 0}$$

2-Step Optimization: Update Step

$$\frac{\partial L}{\partial \boldsymbol{\theta}_j} = \frac{\partial}{\partial \boldsymbol{\theta}_j}\left[\sum_{n=1}^{N}\sum_{k=1}^{K}\alpha_{n,k}\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)\right] = 0$$

$$\boxed{= \frac{\partial}{\partial \boldsymbol{\theta}_j}\left[\sum_{n=1}^{N}\alpha_{n,j}\delta(\mathbf{x}_n, \boldsymbol{\theta}_j)\right] = 0}$$

Solve for each $\boldsymbol{\theta}_j$ independently

Depends on the distance function δ

Take-Home Message of Today

- Focus on hard partitioning clustering

Take-Home Message of Today

- Focus on hard partitioning clustering
- Formulate hard partitioning clustering as a (**`non-convex`**) optimization problem
  - Minimizing "some" aggregated internal cluster distance

Take-Home Message of Today

- Focus on hard partitioning clustering
- Formulate hard partitioning clustering as a (**non-convex**) optimization problem
  - Minimizing "some" aggregated internal cluster distance
- Computing exact solution is **NP-hard** due to exponential search space

Take-Home Message of Today

- Focus on hard partitioning clustering
- Formulate hard partitioning clustering as a (**non-convex**) optimization problem
  - Minimizing "some" aggregated internal cluster distance
- Computing exact solution is **NP-hard** due to exponential search space
- Use an iterative (approximate) solution