Clustering Heterogeneous and Multi-View Data

Lorenzo Mannocci

a.a. 2026/2027



UNIVERSITÀ DI PISA

Clustering beyond numerical tabular data

Most clustering algorithms are introduced assuming that every observation is represented by a numerical feature vector: $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$

Examples:

- K-Means uses Euclidean distance.
- DBSCAN requires a distance measure.
- Hierarchical clustering relies on pairwise distances.

However, real datasets often contain:

- numerical variables,
- categorical variables,
- time series,
- text,
- multiple data sources describing the same object.

How can we cluster observations when the data are heterogeneous?

Main challenge: Defining Similairity

Clustering is fundamentally based on the idea of similarity or distance.
For numerical variables, a distance such as Euclidean distance is straightforward:

$$d(x_i, x_j) = \sqrt{\sum_k (x_{ik} - x_{jk})^2}$$

Age	Income	Country	Occupation
35	42,000	Italy	Engineer
37	45,000	France	Engineer

For age and income, differences are numerical.
But what is the distance between:

- Italy and France?
- Engineer and Teacher?

Before choosing a clustering algorithm, we need a meaningful representation of similarity for each type of data

Mixed numerical and categorical data

Ordinary K-Means is not naturally designed for this situation because its centroid is defined using the arithmetic mean.

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

But the "mean" of: *Italy, France, Italy, Germany* has no meaningful interpretation.

- A boolean attribute corresponding to a single item in a transaction, if that item appears, the boolean attribute is set to '1' or '0' otherwise.
- A categorical attribute may have several values, each value can be treated as an item and represented by a boolean attribute.

$Italy \rightarrow 0$
$France \rightarrow 1$
$Italy \rightarrow 2$
$Germany \rightarrow 3$

Also with One Hot Encoding, you cannot use traditional distances designed for numerical data!

K-modes
Categorical Data

K-modes: K-means for categorical data

$$\text{Minimise} \quad P(W,Q) = \sum_{l=1}^{k}\sum_{i=1}^{n} w_{i,l}\ d(X_i, Q_l)$$ → Minimise the total distance between each object and the mode of the cluster it is assigned to

$$\text{subject to} \quad \sum_{l=1}^{k} w_{i,l} = 1, \qquad 1 \le i \le n$$ → Each object must belong to exactly one cluster

$$w_{i,l} \in \{0,1\}, \qquad 1 \le i \le n, \qquad 1 \le l \le k$$ → Membership is binary: an object either belongs to a cluster or it does not

- $X = X_1, \dots, X_n$: dataset of n objects
- $X_i = [x_1, \dots, x_m]$: object i, described by m categorical attributes
- $Q = \{Q_1, \dots, Q_k\}$: set of representative objects (**mode**) for the k clusters.
- $D(X_i, Q_l)$: distance function between object X_i and the mode Q_l
- $$w_{i,l} = \begin{cases} 1, & \text{if } X_i \text{ belongs to cluster } l \\ 0, & \text{otherwise} \end{cases}$$

Huang, Z. (1998). **Extensions to the k-means algorithm for clustering large data sets with categorical values**. *Data mining and knowledge discovery*, 2(3), 283-304.

K-modes: Distancance and Mode

K-Means as distance uses Euclidean distance:

$$d(X,Y) = \sqrt{\sum_{j=1}^{m} (x_j - y_j)^2}$$

K-Modes as distance uses the number of mismatches between the attributes of two objects:

$$d_1(X,Y) = \sum_{j=1}^{m} \delta(x_j, y_j)$$

$$\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases}$$

- K-Modes uses the mode as representative object of a cluster
- Given the set of objects in the cluster C_l, the mode is get computing the max frequency for each attribute:

$$f_r(A_j = c_{l,j} \mid X_l) = \frac{n_c^{l,k}}{n}$$

Huang, Z. (1998). **Extensions to the k-means algorithm for clustering large data sets with categorical values**. *Data mining and knowledge discovery*, 2(3), 283-304.

K-modes algorithm

1. Randomly select the initial objects as modes
2. Scan of the data to assign each object to the closer cluster identified by the mode
3. Re-compute the mode of each cluster
4. Repeat the steps 2 and 3 until no object changes the assigned cluster

Mixed Data:
Categorical and Numerical

Distance measure

K-Means works with numerical data.
K-Modes works with categorical data.

Cluster prototype contains:

- **Means** for numerical attributes
- **Modes** for categorical attributes

Same optimization problem

$$\text{Minimise} \quad P(W,Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l}\ d(X_i, Q_l)$$

Q_l=[32.5, 45000, Female, Engineer]

$$d(X_i, Q_l) = \sum_{j \in Num} (x_{ij} - q_{lj})^2 + \gamma \sum_{j \in Cat} \delta(x_{ij}, q_{lj})$$

$$\delta(x_{ij}, q_{lj}) = \begin{cases} 0 & (x_{ij} = q_{lj}) \\ 1 & (x_{ij} \neq q_{lj}) \end{cases}$$

Numerical distance

γ: weight controlling the importance of categorical attributes

Categorical mismatch

Huang, Z. (1998). **Extensions to the k-means algorithm for clustering large data sets with categorical values**. *Data mining and knowledge discovery*, 2(3), 283-304.

K-prototypes algorithm

1. Choose number of clusters k
2. Initialize k prototypes
3. Assign each object to the nearest prototype
4. Update prototypes:
    1. numerical attributes → mean
    2. categorical attributes → mode
5. Repeat until assignments stop changing

Huang, Z. (1998). **Extensions to the k-means algorithm for clustering large data sets with categorical values**. *Data mining and knowledge discovery*, 2(3), 283-304.

Gower distance

K-Prototypes changes the clustering algorithm.
Gower distance is designed for mixed-type observations.
For each feature k, compute a feature-specific dissimilarity:

$$d(i,j) = \frac{\sum_k w_{ijk}\, d_{ijk}}{\sum_k w_{ijk}}$$

Where w_{ijk} are non-negative weights usually set to 1 and d_{ijk}:

- For a numerical feature:

$$d_{ijk} = \frac{|x_{ik} - x_{jk}|}{\text{range}_k}$$

range of $k-th$ variable, ensuring $0 \leq s_{ijk} \leq 1$

- For a categorical feature:

$$d_{ijk} = \begin{cases} 0, & x_{ik} = x_{jk} \\ 1, & x_{ik} \neq x_{jk} \end{cases}$$

Each feature contributes a value between 0 and 1.
Then you can use clustering algorithms that accept arbitrary distances.

- K-Prototypes is a natural extension of K-Means for mixed data
- Gower distance is more general because it separates the definition of similarity from the clustering algorithm

Gower, J. C. (1971). **A general coefficient of similarity and some of its properties**. *Biometrics*, 857-871.

Multi-View Clustering (MVC)

From mixed-type data to multi-view data

Audio



Image



Video



Text



K-Prototypes assumes a single representation: $X \in \mathbb{R}^{n \times d}$, possibly containing numerical and categorical attributes.

Multi-view learning instead assumes several representations of the same objects: $X^{(1)}, X^{(2)}, \ldots, X^{(V)}$, with $X^{(v)} \in \mathbb{R}^{n \times d_v}$ where all views refer to the same n objects.

Examples:

- $X^{(1)} = \text{visual features}$
- $X^{(2)} = \text{textual features}$
- $X^{(3)} = \text{behavioral features}$

This distinction is important: mixed-type data and multi-view data are not the same problem.

Why multiple views?

Audio



Image



Video



Text

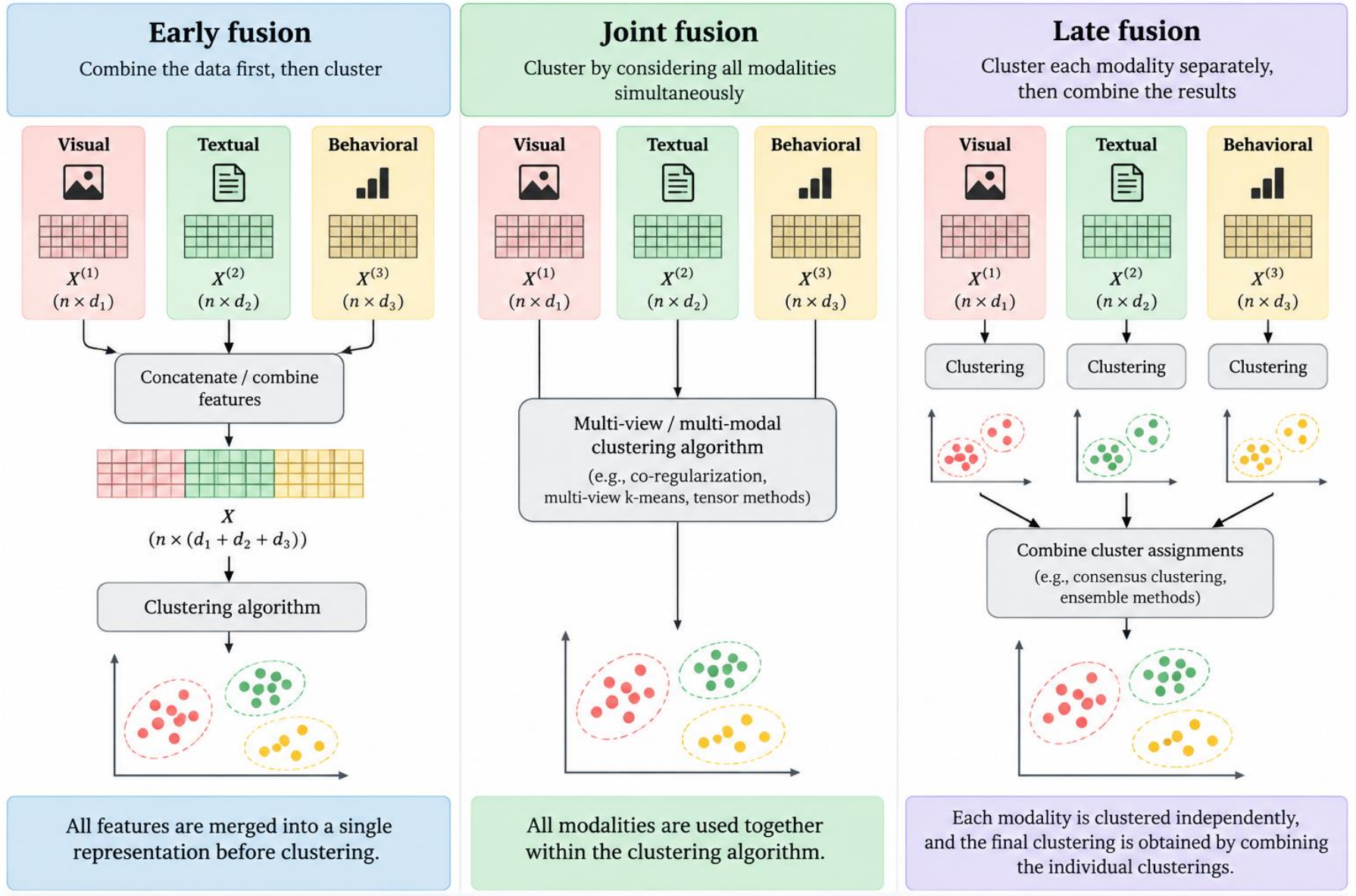


Different views may capture:

- complementary information
- different modalities
- different feature spaces
- different levels of abstraction

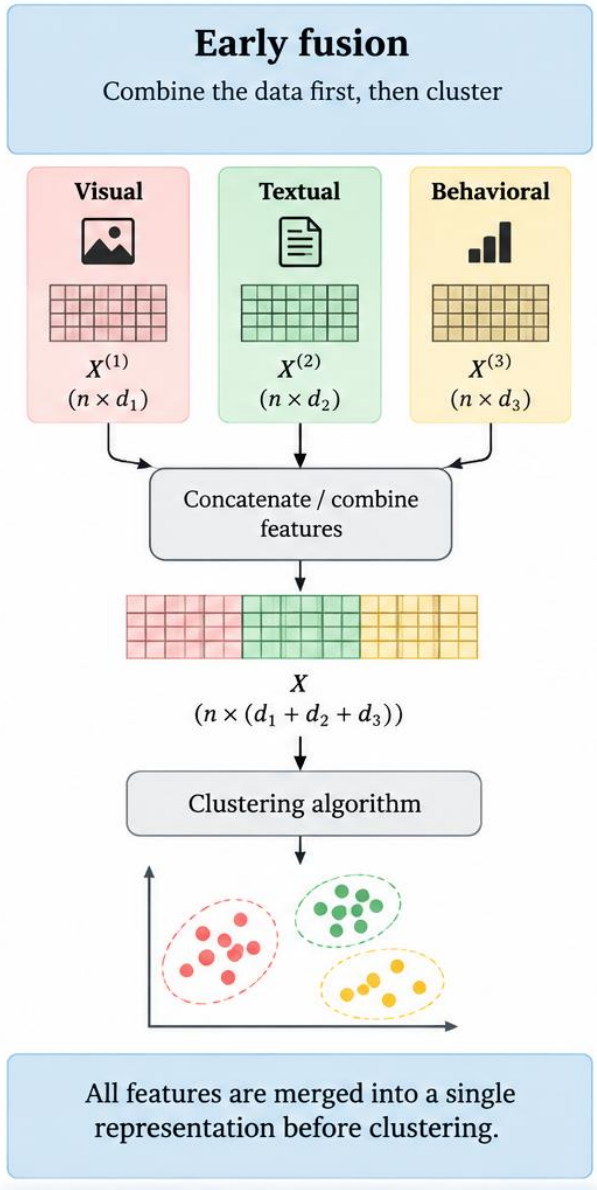
The goal is to obtain a clustering that exploits information across views rather than analyzing each representation in isolation.

Three fusion strategies



Early Fusion

Early Fusion

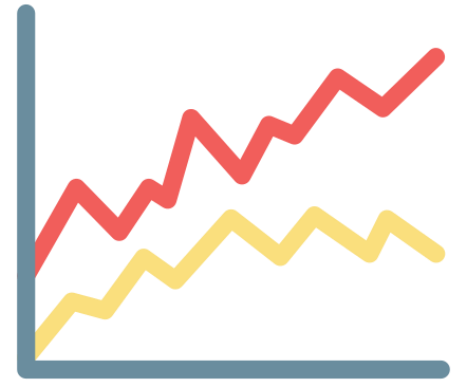


Each data source is first converted into a numerical feature representation.
Then the representations are concatenated:

$$z_i = \left[x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(M)}\right]$$

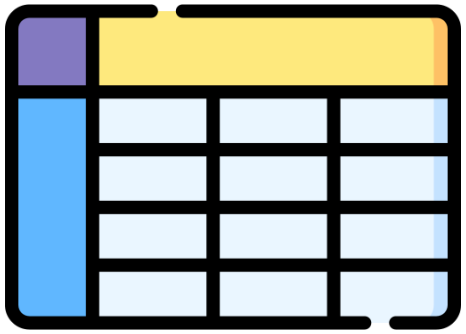
The resulting vector can be clustered using standard algorithms.

Early Fusion: Example



tsfresh

Statistical
feature vector



Tabular
feature vector

1. Scaling
2. Dimensionality Reduction
3. Clustering

The main challenge is how the different sources are represented and combined.

Early Fusion: One modality can dominate

- static table: 5 features
- tsfresh: 400 features

If all features are concatenated directly, the time-series representation may dominate the distance.

Similarly, variables with very different scales can dominate Euclidean distance. Therefore the fused representation may require:

- feature scaling
- feature selection
- dimensionality reduction
- **modality weighting** ⟶

$$d(i,j) = w_{tab} d_{tab}(i,j) + w_{TS} d_{TS}(i,j)$$

where:

- d_{tab} distance in tabular representation
- d_{TS} distance in time-series representation
- w_{tab} and w_{TS} control their relative importance

How much should each modality contribute?

Representation-Level Fusion

In early fusion, each modality is converted into handcrafted features and the resulting feature vectors are concatenated.

For object i: $x_i^{(1)} \rightarrow z_i^{(1)}$, $x_i^{(2)} \rightarrow z_i^{(2)}$

Then combine the representations:

$$z_i = \left[z_i^{(1)}, z_i^{(2)}\right]$$

and apply a standard clustering algorithm to z_i.

Learn a compact numerical representation for each modality before combining them.

Different modalities are transformed into comparable numerical embeddings before clustering.

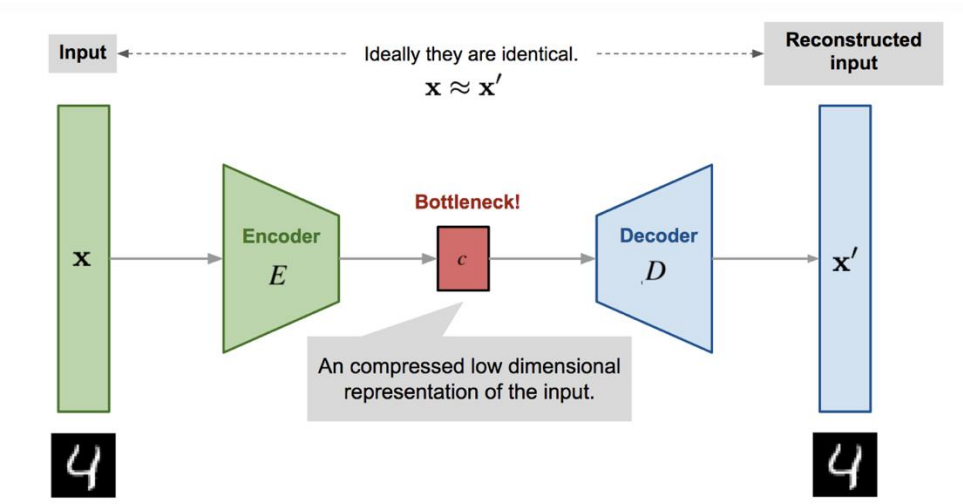
Autoencoders as Representation Learners

An Autoencoder can be used to learn a compact representation:

$$x \rightarrow Encoder \rightarrow \mathrm{c} \rightarrow Decoder \rightarrow \mathrm{x}'$$

The model is trained to reconstruct the original input.

After training, the decoder can be discarded and the latent vector z can be used as a compact feature representation.



Representations can capture nonlinear structures that handcrafted features or PCA may miss.

Autoencoders as Representation Learners

Advantages:

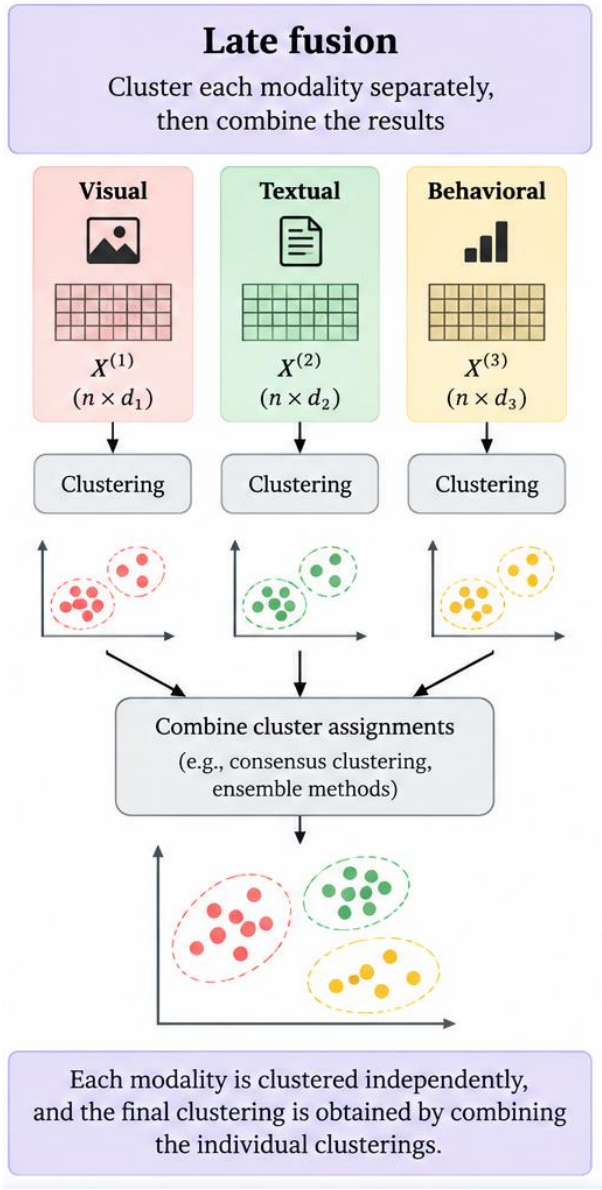
- simple
- computationally efficient
- easy to implement

Limitations:

- assumes views can be meaningfully combined
- sensitive to differences in scale
- noisy or irrelevant views may dominate
- does not explicitly model inter-view relationships

Late Fusion

Late Fusion



Each view is clustered independently:

$$X^{(1)} \rightarrow C^{(1)}$$
$$X^{(2)} \rightarrow C^{(2)}$$
$$\vdots$$
$$X^{(V)} \rightarrow C^{(V)}$$

Then the clusterings are combined:

$$\{C^{(1)}, \ldots, C^{(V)}\} \rightarrow C^{*}$$

The fusion is performed on the **clustering results**, not on the original features.

Late Fusion

Advantages:

- different algorithms can be used for different views
- naturally handles heterogeneous feature spaces

Limitation:

- views are learned independently, so cross-view interactions are not exploited during clustering

Consensus clustering

Given multiple partitions:

$$\Pi = \{C^{(1)}, C^{(2)}, \ldots, C^{(V)}\}$$

find a consensus partition:

$$C^*$$

that summarizes the common structure across them.

Note that **clusters labels are arbitrary**, so we cannot directly average or vote on labels.

From graphs and hypergraphs

An ordinary undirected graph is defined as $G(V, E)$ where:

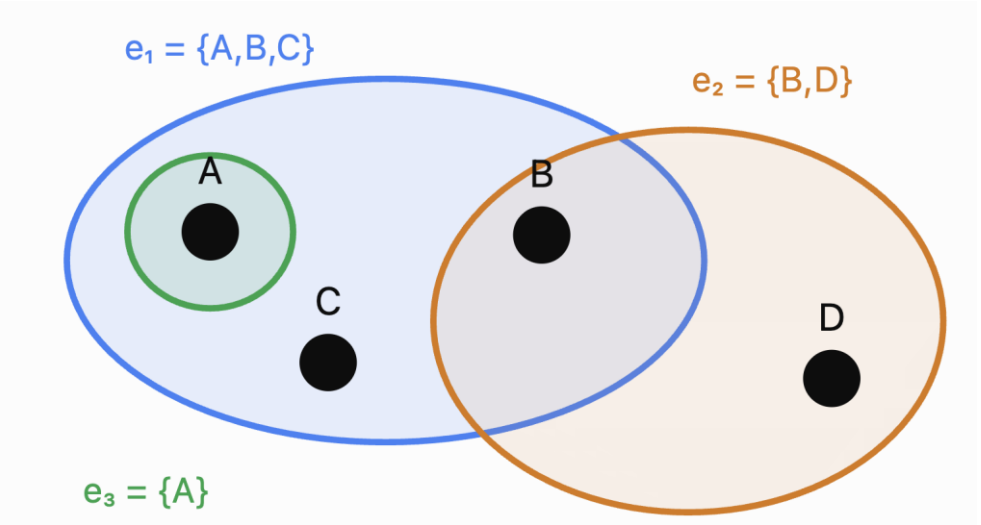
- $V = \{v_1, \dots, v_n\}$ is the set of vertices;
- E is the set of edges;
- each edge connects a pair of vertices $e = \{v_i, v_j\}$.

$$e \in E \Rightarrow e \subseteq V, |e| = 2$$

A hypergraph generalizes this idea $H = (V, E)$ where:

- $V = \{v_1, \dots, v_n\}$ is the set of vertices;
- $E = \{e_1, \dots, e_m\}$ is the set of hyperedges;
- Each hyperedge is a subset of vertices $e_a \subseteq V$.

Unlike an ordinary edge, a hyperedge can connect more than two vertices: $|e_a| \geq 1$.
For example: $e_1 = \{v_1, v_2, v_4, v_5\}$ is a hyperedge simultaneously connects four vertices.



Conceptual hyperdge with three hyperedges, including a singleton hyperedge $\{A\}$

Meta-Clustering Algorithm (MCLA)

Input: Q partitions of n objects; desired number of consensus clusters k.

1. Hypergraph representation: Represent each base cluster as a hyperedge and construct the incidence matrix H.
2. Meta-graph construction: Construct a weighted graph G_M, where vertices represent hyperedges and edge weights are their Jaccard similarities.
3. Meta-graph partitioning: Use METIS to partition G_M into k approximately balanced meta-clusters, minimizing the total weight of edges connecting different groups.
4. Collapse meta-clusters: Average the indicator vectors of the hyperedges in each meta-cluster to obtain k meta-hyperedges and the association matrix A.
5. Compete for objects: Assign each object to the meta-cluster with the highest association.

Output: Final consensus partition C^*.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Cluster Ensemble as a Hypergraph

Let $X = \{x_1, \dots, x_n\}$ be the set of n objects to cluster.
Suppose we have Q different input clusterings, also called partitions $\Pi = \{\lambda^{(1)}, \lambda^{(2)}, \dots, \lambda^{(Q)}\}$, where

- $q = 1, \dots, Q$ indexes the input partitions;
- $\lambda^{(q)} \in \mathbb{N}^n$ is the label vector produced by partition q;
- $k^{(q)}$ is the number of clusters in partition q.

For example, $\lambda_i^{(q)}$ is the cluster label assigned to object x_i by partition q

For each partition q, construct the binary membership matrix $H^{(q)} \in \{0,1\}^{n \times k^{(q)}}$ with entries:

$$H_{ij}^{(q)} = \begin{cases} 1 & \text{if } x_i \text{ belongs to cluster } j \text{ in partition } q \\ 0 & \text{otherwise} \end{cases}$$

Where:

- $i = 1, \dots, n$ indexes objects;
- $j = 1, \dots, k^{(q)}$ indexes clusters inside partition q

For hard clustering,

$$\sum_{j=1}^{k^{(q)}} H_{ij}^{(q)} = 1$$

Summing values of a row, along all columns, i.e., clusters in partition q, x_i belongs to one cluster.

for every object x_i, each object belongs to exactly one cluster in each partition.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Hypergraph interpretation

Each column of $H^{(q)}$ represents one cluster. For example,

$$h_a = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

means that the corresponding cluster contains objects $\{x_1, x_2, x_4\}$.

In the hypergraph representation:

- the n original objects are the vertices;
- every cluster is a hyperedge;
- a hyperedge can connect any number of vertices.

A cluster is naturally represented as a hyperedge: it connects all objects belonging to that cluster.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Combining all partitions

Concatenate all the membership matrices horizontally: $H = \left[H^{(1)} \mid H^{(2)} \mid \cdots \mid H^{(\mathrm{Q})}\right]$
Therefore, $H \in \{0,1\}^{n \times m}$ where

$$m = \sum_{q=1}^{\mathrm{Q}} k^{(q)}$$

is the total number of clusters produced by all input partitions. Thus $H = [h_1, h_2, \ldots, h_m]$ and every column h_a is one hyperedge, with $a = 1, \ldots, m$.

$$\text{n objects} \longrightarrow \text{n hypergraph vertices}$$

$$m = \sum_{q=1}^{r} k^{(q)} \text{ input clusters} \longrightarrow m \text{ hyperedges}$$

This concatenated matrix H is the hypergraph incidence matrix.

$$H_{ia} = \begin{cases} 1 & \text{if vertex/object } x_i \text{ belongs to hyperedge } e_a \\ 0 & \text{otherwise} \end{cases}$$

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

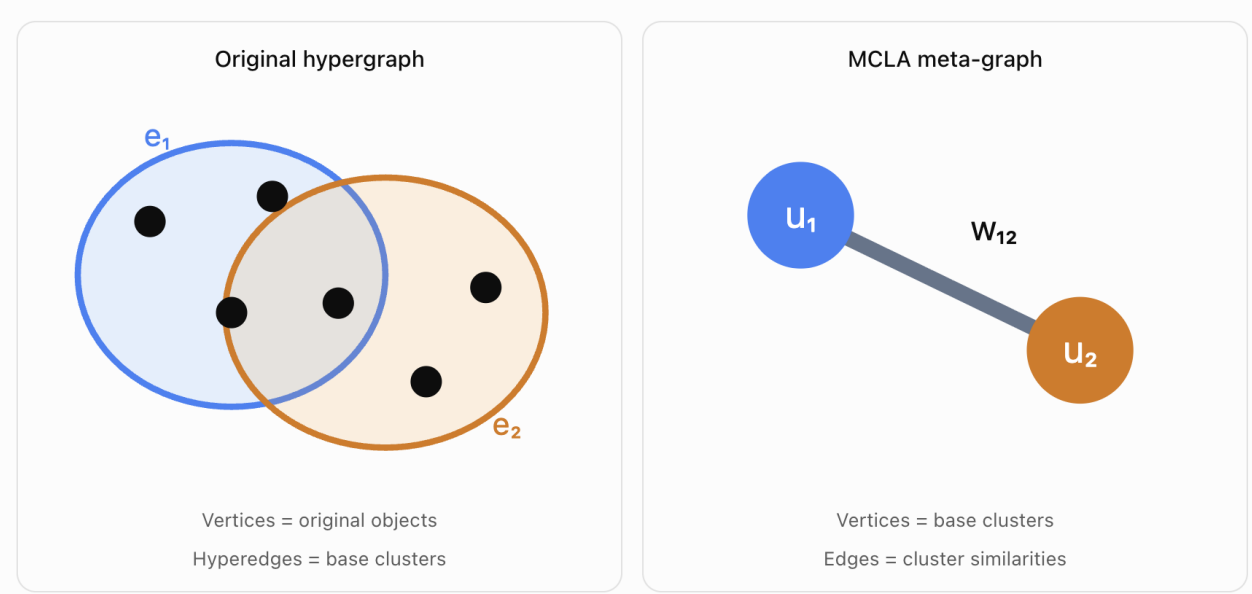
MCLA – Constructing MCLA Meta-Graph

Starting from the hypergraph incidence matrix $H = [\mathbf{h}_1, \ldots, \mathbf{h}_m] \in \{0,1\}^{n \times m}$ MCLA constructs an ordinary weighted undirected meta-graph $G_M = (V_M, E_M, W)$, where:

- Vertices: $V_M = \{u_1, \ldots, u_m\}$, where each u_a represents a hyperedge e_a, i.e., a cluster from an input partition.
- Edges: $E_M \subseteq \{\{u_a, u_b\} : a \neq b\}$, connecting pairs of overlapping base clusters.
- Edge weights: measuring the similarity between the corresponding clusters using Jaccard similarity.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Constructing MCLA Meta-Graph



This is the important change of level:

Original hypergraph: objects are vertices → MCLA meta-graph: clusters/hyperedges are vertices

The input partitions may identify similar groups of objects but assign them different cluster labels. By representing clusters as vertices and their similarities as edges, MCLA can identify and group corresponding clusters across different partitions through graph partitioning.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA - Edge weights

How to compute the Jaccard similarity?
MCLA measures the similarity between every pair of hyperedges using the binary Jaccard measure $w_{ab} = \frac{|e_a \cap e_b|}{|e_a \cup e_b|}$.

Using the columns of H:

$$w_{ab} = \frac{h_a^\top h_b}{\|h_a\|_2^2 + \|h_b\|_2^2 - h_a^\top h_b}$$

The meta-graph can be represented by a weighted adjacency matrix $W \in [0,1]^{m \times m}$.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA - Edge weights

$$w_{ab} = \frac{h_a^\top h_b}{\|h_a\|_2^2 + \|h_b\|_2^2 - h_a^\top h_b}$$

Numerator

Columns of incidence matrix are binary vectors $h_a, h_b \in \{0,1\}^n$ where $h_{ia} = 1 \Leftrightarrow x_i \in e_a$.
Dot product:

$$h_a^\top h_b = \sum_{i=1}^{n} h_{ia}\, h_{ib}$$

Because both variables are binary:

$$h_{ia}h_{ib} = \begin{cases} 1 & x_i \in e_a \text{ and } x_i \in e_b \\ 0 & \text{otherwise} \end{cases}$$

Therefore, the sum counts the objects belonging to both hyperedges: $h_a^\top h_b = |e_a \cap e_b|$

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Edge weights

$$w_{ab} = \frac{h_a^\top h_b}{\|h_a\|_2^2 + \|h_b\|_2^2 - h_a^\top h_b}$$

Denominator

Squared norm is:

$$\|h_a\|_2^2 = \sum_{i=1}^{n} h_{ia}^2$$

But since $h_{ia} \in \{0,1\}$, we have that $h_{ia}^2 = h_{ia}$, hence:

$$\|h_a\|_2^2 = \sum_{i=1}^{n} h_{ia} = |e_a|$$

Similarly, $\|h_b\|_2^2 = |e_b|$

From basic set theory $|e_a \cup e_b| = |e_a| + |e_b| - |e_a \cap e_b|$.
Substituting the vector expressions:

$$|e_a \cup e_b| = \|h_a\|_2^2 + \|h_b\|_2^2 - h_a^\top h_b$$

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Edge weights (example)

$$w_{ab} = \frac{h_a^\top h_b}{\|h_a\|_2^2 + \|h_b\|_2^2 - h_a^\top h_b}$$

$h_a = [1,1,1,0,0]^\top$ and $h_b = [0,1,1,1,0]^\top \longrightarrow e_a = \{x_1, x_2, x_3\}, \quad e_b = \{x_2, x_3, x_4\}$

Intersection: $h_a^\top h_b = 2$

Cluster sizes: $\|h_a\|_2^2 = 3, \quad \|h_b\|_2^2 = 3$
Union: $3 + 3 - 2 = 4$
Weight: $w_{ab} = \frac{2}{4} = 0.5$

The two base clusters have a Jaccard similarity of 0.5, so the corresponding vertices in the MCLA meta-graph are connected with weight 0.5.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Partitioning the meta-graph

Once $G_M = (V_M, E_M, W)$ has been constructed, MCLA partitions its m vertices into k disjoint meta-clusters: $V_M = M_1 \cup \cdots \cup M_k$.

Recall that each vertex $u_a \in V_M$ represents one hyperedge, i.e. one cluster from an input partition. Therefore, we are now clustering the original clusters.

A large w_{ab} means that hyperedges e_a and e_b contain many of the same objects. Therefore, we want strongly connected vertices to remain in the same meta-cluster. Equivalently, we want to minimize the total weight of edges that are cut by the partition.

$$\text{cut}(M_1, \ldots, M_k) = \sum_{\substack{(u_a, u_b) \in E_M \\ u_a \in M_p, u_b \in M_q \\ p \neq q}} w_{ab}$$

$$\min_{M_1, \ldots, M_k} \text{cut}(M_1, \ldots, M_k)$$

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Partitioning the meta-graph

If we only minimize the cut, a trivial solution could put almost every vertex into one group and very few into the others. Therefore MCLA uses balanced graph partitioning.
Importantly, in the original formulation, meta-graph vertices are weighted according to the size of the corresponding hyperedges. That is, for vertex u_a:

$$\omega_a = |e_a|$$

where $|e_a|$ is the number of original objects belonging to that base cluster.
The weight of a meta-cluster is therefore:

$$\Omega(M_j) = \sum_{u_a \in M_j} \omega_a$$

The graph partitioner seeks k approximately balanced meta-clusters while minimizing the weighted edge cut.

Minimize: similarity cut between meta-clusters
Subject to: approximately balanced meta-clusters

METIS is the graph-partitioning algorithm used to approximately solve this constrained optimization problem.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Partitioning the meta-graph

Suppose two meta-clusters contain: $\mathcal{M}_1 = \{e_1, e_2\}$ and $\mathcal{M}_2 = \{e_3, e_4\}$.
Having two hyperedges in each does not necessarily mean they are balanced.

If $|e_1| = 100, \quad |e_2| = 12$ while: $|e_3| = 10, \quad |e_4| = 15,$
then: $\Omega(\mathcal{M}_1) = 112$ but: $\Omega(\mathcal{M}_2) = 25.$

Hence MCLA considers the cardinality of the original clusters, not simply the number of meta-graph vertices.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Collapsing meta-cluster into meta-hyperedges

We still do not yet have a partition of the original objects.

Given a meta-cluster containing several hyperedges, how strongly does each original object belong to it?

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Collapsing meta-cluster into meta-hyperedges

After partitioning the meta-graph, we obtain k meta-clusters: $\mathcal{M}_1, \ldots, \mathcal{M}_k$.
Each meta-cluster contains several hyperedges corresponding to similar base clusters.
For each meta-cluster $\mathcal{M}_j$, MCLA collapses its hyperedges into a single meta-hyperedge.

Let the hyperedges contained in meta-cluster $\mathcal{M}_j$ be represented by their binary indicator vectors: $\mathbf{h}_a \in \{0,1\}^n$
MCLA computes the association vector:

$$a_j = \frac{1}{\left|\mathcal{M}_j\right|} \sum_{h_a \in M_j} h_a$$

where:

- $\left|\mathcal{M}_j\right|$: number of hyperedges in meta-cluster j;
- $a_j \in [0,1]^n$: association vector of meta-cluster j;
- a_{ji}: association between object x_i and meta-cluster j.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Interpretation of association vector

For object x_i:

$$a_{ji} = \frac{\text{number of hyperedges in } \mathcal{M}_j \text{ containing } x_i}{|\mathcal{M}_j|}$$

- $a_{ji} = 1 : x_i$ belongs to all base clusters grouped in meta-cluster j;
- $a_{ji} = 0$: means that x_i belongs to none of them;
- Intermediate values represent partial support.

For example, $a_{ji} = 0.75$ means that x_i belongs to 75% of the hyperedges contained in $\mathcal{M}_j$.

Suppose $\mathcal{M}_1 = \{\mathbf{h}_1, \mathbf{h}_4, \mathbf{h}_7\}$:

- $\mathbf{h}_1 = [1, 1, 1, 0, 0]^\mathsf{T}$
- $\mathbf{h}_4 = [1, 1, 0, 0, 0]^\mathsf{T}$
- $\mathbf{h}_7 = [1, 1, 1, 0, 0]^\mathsf{T}$



Then $a_1 = \frac{h_1 + h_4 + h_7}{3}$ so:

$$\mathbf{a}_1 = \left[1, 1, \frac{2}{3}, 0, 0\right]^\mathsf{T}$$

- x_1, x_2: strongly associated with meta-cluster 1;
- x_3: partially associated;
- x_4, x_5: not associated.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

MCLA – Meta-hyperedges matrix

- n: number of original objects;
- k: number of meta-clusters, which corresponds to the desired number of final consensus clusters.

Collect the association vectors into: $A = [\mathbf{a}_1, \ldots, \mathbf{a}_k]$, with $A \in [0,1]^{n \times k}$ where A_{ij} measures the association of object x_i with meta-cluster j.

So after collapsing, we pass from many hyperedges to k meta-hyperedges.

The final step of MCLA is called Compete for Objects. Each object is assigned to the meta-cluster with maximum association (operates row by row):

$$\lambda^*(x_i) = \arg \max_{j=1,\ldots,k} A_{ij}$$

Thus, the final consensus partition is:

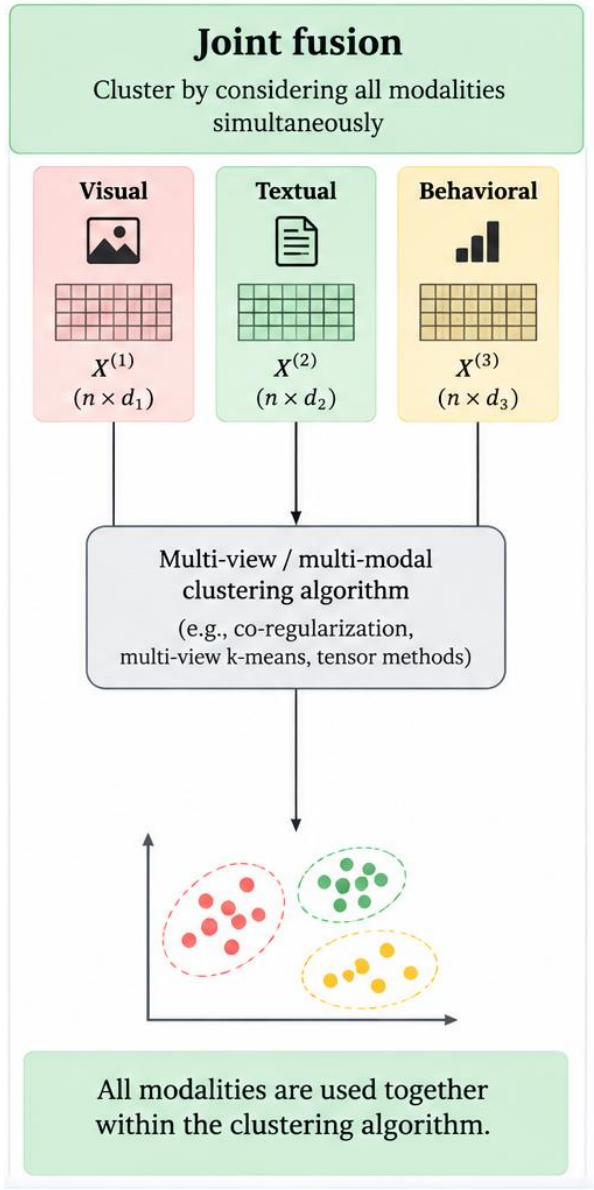
$$C^* = \{C_1^*, \ldots, C_k^*\}$$

with each object assigned according to its strongest meta-cluster association.

Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions**. *Journal of machine learning research*, 3(Dec), 583-617.

Joint Fusion

Joint fusion



Learning a common representation or model while explicitly exploiting dependencies and complementarities among views.

$$\{X^{(1)}, \dots, X^{(V)}\} \rightarrow Z \rightarrow C$$

where:

- $X(v)$: representation of view v;
- Z is a shared or consensus representation;
- C: final clustering.

This is only one possible joint-learning strategy.

Joint fusion

Advantages:

- information discovered in one view to influence the representation or clustering learned from another view
- exploit interactions between views

Limitations:

- more complex optimization
- higher computational cost.

Multi-view K-means

K-Means assumes a single representation $X^{(v)} \in \mathbb{R}^{n \times d_v}$.
In multi-view data, the same n objects are described by different feature sets $X^{(1)}, X^{(2)}, \ldots, X^{(V)}$.
Multi-View K-Means keeps the views separate, but makes them interact during clustering.
Standard K-Means alternates between assign objects to clusters and update centroids. Multi-view K-means does this for multiple views.

Each view has its own feature space $X^{(v)} \in \mathbb{R}^{n \times d_v}$ and its own centroids $\mu_1^{(v)}, \ldots, \mu_k^{(v)}$, but the clustering should be consistent across views.

Bickel, S., & Scheffer, T. (2004, November). **Multi-view clustering**. In *ICDM* (Vol. 4, No. 2004, pp. 19-26).

Multi-view K-means

For one view v, the usual K-means objective is:

$$J^{(v)} = \sum_{i=1}^{n} \sum_{c=1}^{k} w_{ic}^{(v)} \left\| x_i^{(v)} - \mu_c^{(v)} \right\|^2$$

where:

- $x_i^{(\mathrm{v})}$: object i in view v
- $\mu_c^{(v)}$: centroid of cluster c in view v
- $w_i c^{(v)} = 1$ if object i is assigned to cluster c, otherwise 0

If we clustered each view independently, we would minimize each $J^{(\mathrm{v})}$separately.

Bickel, S., & Scheffer, T. (2004, November). **Multi-view clustering**. In *ICDM* (Vol. 4, No. 2004, pp. 19-26).

Multi-view K-means

Multi-View K-Means follows a co-training style idea.
At each iteration:

1. Cluster one view using K-Means.
2. Transfer the cluster assignment information to the other view.
3. Re-estimate the centroids in the other view.
4. Repeat until the cluster assignments stabilize.

So the views are not fused at the feature level.

Bickel, S., & Scheffer, T. (2004, November). **Multi-view clustering**. In *ICDM* (Vol. 4, No. 2004, pp. 19-26).

Multi-view K-means

Strengths:

- easy to understand after standard K-Means
- keeps separate feature spaces
- introduces the idea of cross-view interaction
- useful bridge from K-Means to more advanced multi-view clustering

Limitations:

- still relies on K-Means assumptions
- sensitive to initialization
- mainly suitable when cluster geometry is centroid-based
- less expressive than spectral or graph-based multi-view methods
- order of the views may affect the results

MultiviewKMeans performs multiple initializations (*n_init*) and keeps the solution with the best total inertia across the views.

Bickel, S., & Scheffer, T. (2004, November). **Multi-view clustering**. In *ICDM* (Vol. 4, No. 2004, pp. 19-26).

Multi-view Spectral Clustering

K-Means clusters objects directly in their feature space. Spectral clustering first represents objects as a similarity graph, transforms that graph into a new representation where the cluster structure is easier to detect, and then applies a standard clustering algorithm such as K-Means.

For each view R, $X^{(v)} \rightarrow S^{(v)} \rightarrow U^{(v)}$ where:

- $S^{(v)}$: similarity graph of view R
- $U^{(v)}$: low-dimensional spectral representation

The views are then iteratively refined using information from the other views.
"Co–training = iterative exchange of spectral information"

Kumar, A., & Daumé, H. (2011). **A co-training approach for multi-view spectral clustering**.
In Proceedings of the 28th international conference on machine learning (*ICML-11*) (pp. 393-400).

Co-regularized Multi-view Spectral Clustering

Each view learns its own spectral representation $U^{(1)}, U^{(2)}, \ldots, U^{(V)}$
but all representations are optimized jointly.

$$\min_{U^{(1)},\ldots,U^{(V)}} \sum_{v=1}^{V} L_v\left(U^{(v)}\right) + \lambda \sum_{v \neq w} D\left(U^{(v)}, U^{(w)}\right)$$

view-specific spectral objective

cross-view agreement

where:

- $\mathcal{L}_v$: preserves the clustering structure of view R
- ϴ : penalizes disagreement between views
- λ: controls the strength of cross-view consistency

Kumar, A., Rai, P., & Daume, H. (2011). **Co-regularized multi-view spectral clustering**. *NeurIPS*, 24.

Multi-view Spectral Clustering

Strengths:

- exploits graph structure in each view
- can capture non-spherical clusters
- allows views to improve each other
- more expressive than Multi-View K-Means

Limitations:

- depends on graph construction
- requires parameters such as number of neighbours or kernel scale
- more computationally expensive
- still relies on iterative refinement, so convergence and initialization may affect results

Kumar, A., & Daumé, H. (2011). **A co-training approach for multi-view spectral clustering**.
In Proceedings of the 28th international conference on machine learning (*ICML-11*) (pp. 393-400).

References

1. Huang, Z. (1997). **A fast clustering algorithm to cluster very large categorical data sets in data mining**. *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*
2. Huang, Z. (1998). **Extensions to the k-means algorithm for clustering large data sets with categorical values**. *Data mining and knowledge discovery*, 2(3), 283-304.
3. Gower, J. C. (1971). **A general coefficient of similarity and some of its properties**. *Biometrics*, 857-871.
4. Strehl, A., & Ghosh, J. (2002). **Cluster ensembles---a knowledge reuse framework for combining multiple partitions.** Journal of machine learning research, 3(Dec), 583-617.
5. Bickel, S., & Scheffer, T. (2004, November). **Multi-view clustering**. In ICDM (Vol. 4, No. 2004, pp. 19-26).
6. Kumar, A., & Daumé, H. (2011). **A co-training approach for multi-view spectral clustering**. In Proceedings of the 28th international conference on machine learning (ICML-11) (pp. 393-400).
7. Kumar, A., Rai, P., & Daume, H. (2011). **Co-regularized multi-view spectral clustering**. NeurIPS, 24.
8. https://mvlearn.github.io/index.html