Master Degree in Computer Science
Master Degree in Data Science and Economics

Information Retrieval

Unsupervised Text Classification

Prof. Alfio Ferrara

Department of Computer Science, Università degli Studi di Milano
Room 7012 via Celoria 18, 20133 Milano, Italia alfio.ferrara@unimi.it

sed noli modo

Introduction to text classification

Text classification is the problem of **associating texts** (i.e., documents) **to classes** (or labels denoting classes).

Classes may be

- A partition of the document space (i.e., disjoint classes)
- Overlapping (i.e., a document may be classified in more than one class)

Methods may be

- Based on a training set of pre-classified documents: **supervised**
- Based on documents only: **unsupervised** (today's topic)

	Choice between 2 classes	**Many possible classes**
Disjoint classes	Binary classification	Multi-class classification
Overlapping classes	Soft binary classification	Multi-label classification

Introduction to text classification

Many problems may be modeled as a classification problem. Examples:

	Choice between 2 classes	Many possible classes
Disjoint classes	Binary classification	Multi-class classification
Overlapping classes	Soft binary classification	Multi-label classification

Document search
(Class of Relevant vs non relevant documents)

Product Reviews
(Classes for the quality of product, such as from 1 to 5 stars)

Ranked document search
(Class of Relevant vs non relevant documents with scores for documents)

Topic Modeling
Entity Recognition
(A document may contain more topics or entities)

Unsupervised classification: clustering

Clustering is the problem of to **group objects** (i.e., documents but also terms) in clusters that are coherent internally but different from each other

In particular, given a **distance criterion for items**, we aim at **minimizing the distance within a cluster** and **maximizing the distance among different clusters**

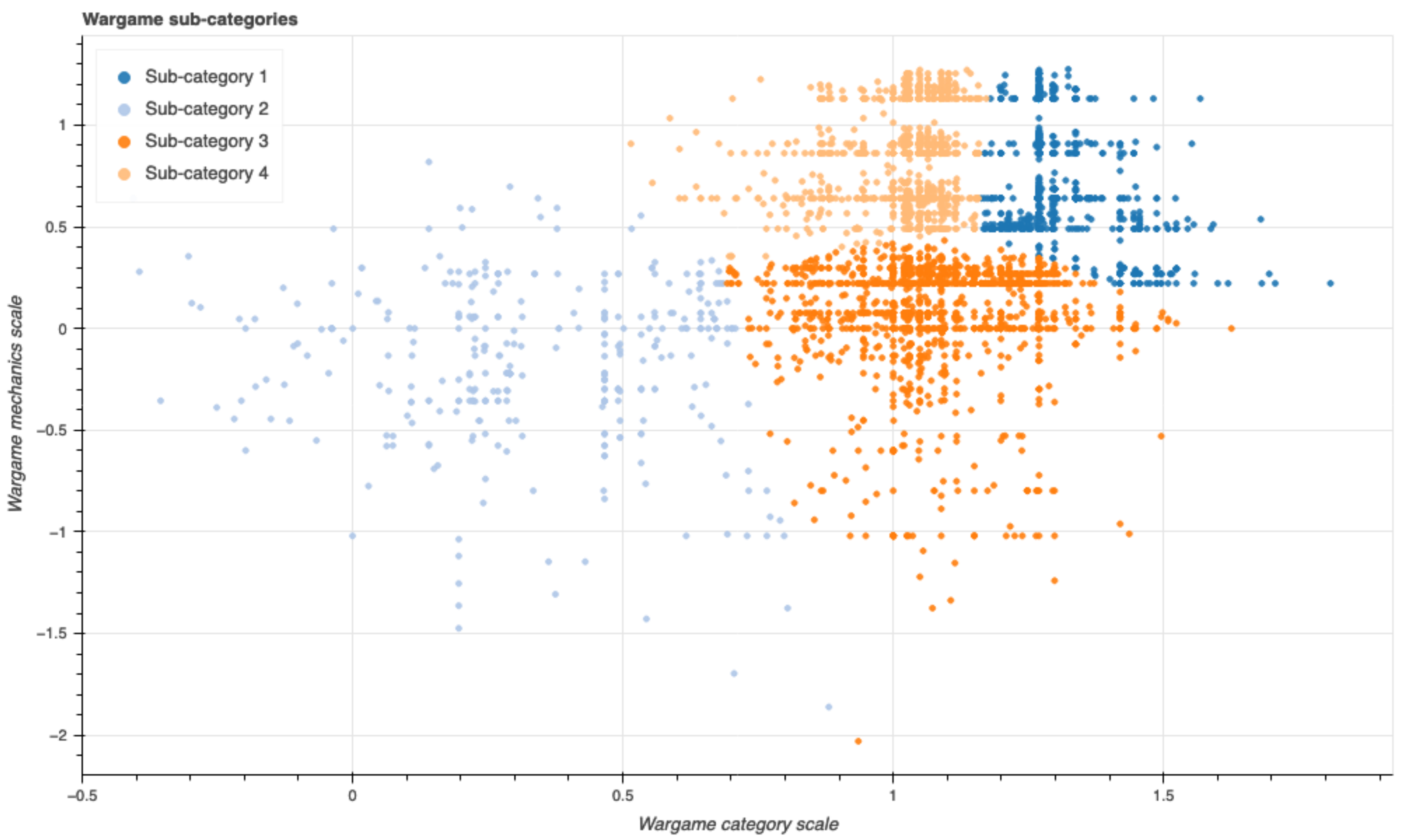
The function that measures the inter-cluster and intra-clusters distance is called **objective function**

Documents within a cluster should be as similar as possible and documents in a cluster should be as dissimilar as possible from documents in other clusters

Clustering

In clustering, a critical aspect is the **number of clusters**

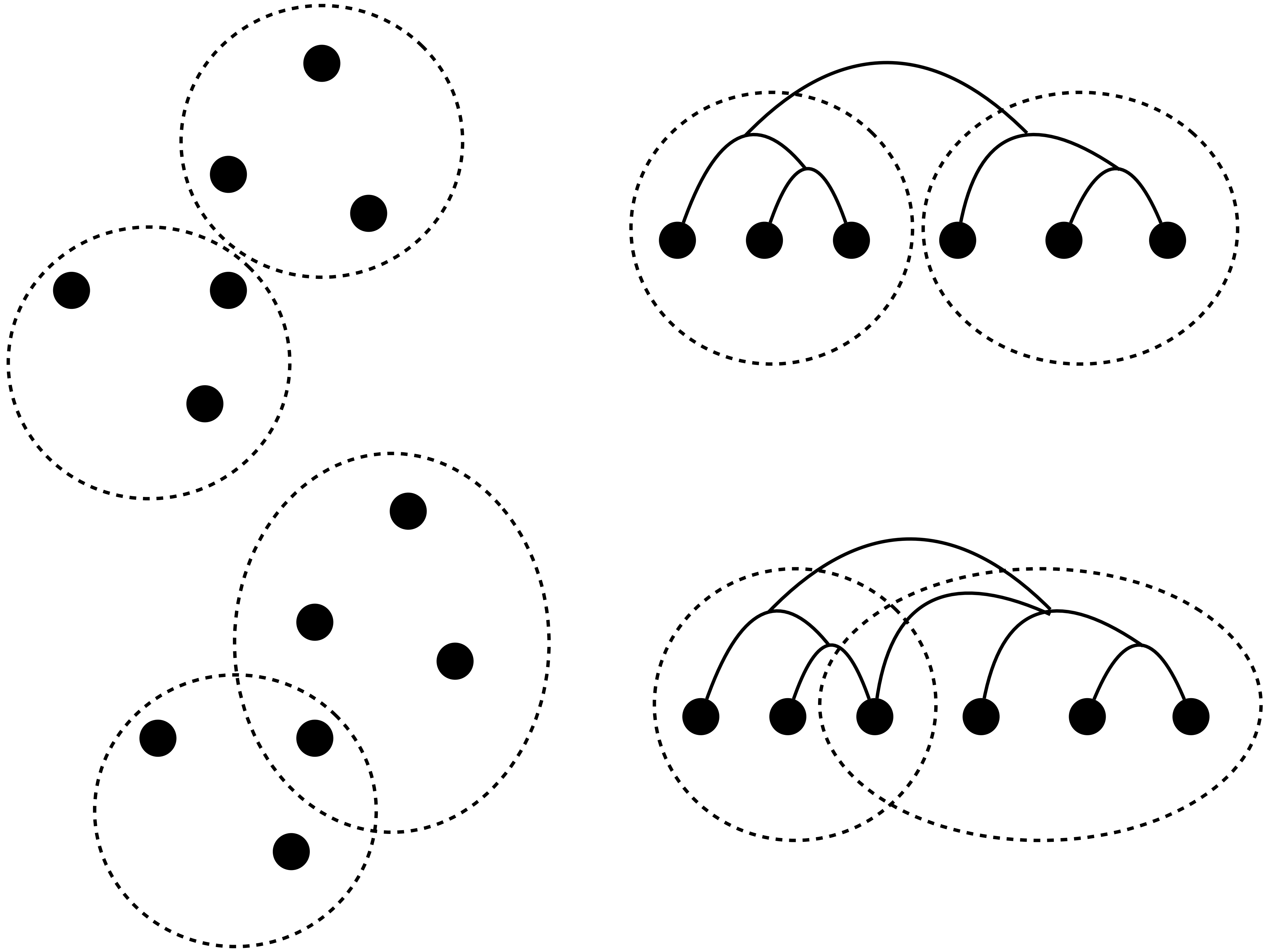
- A high number of clusters produces usually a set of highly consistent groups of items but it’s less effective in aggregating items
- A low number of clusters produces a useful result in terms of the capability of aggregating items, but also a set of less consistent groups of items



Clustering approaches

Clustering structure
- **Flat clustering:** no explicit structure that relates clusters one to each other
- **Hierarchical clustering:** clusters are organized in a hierarchy

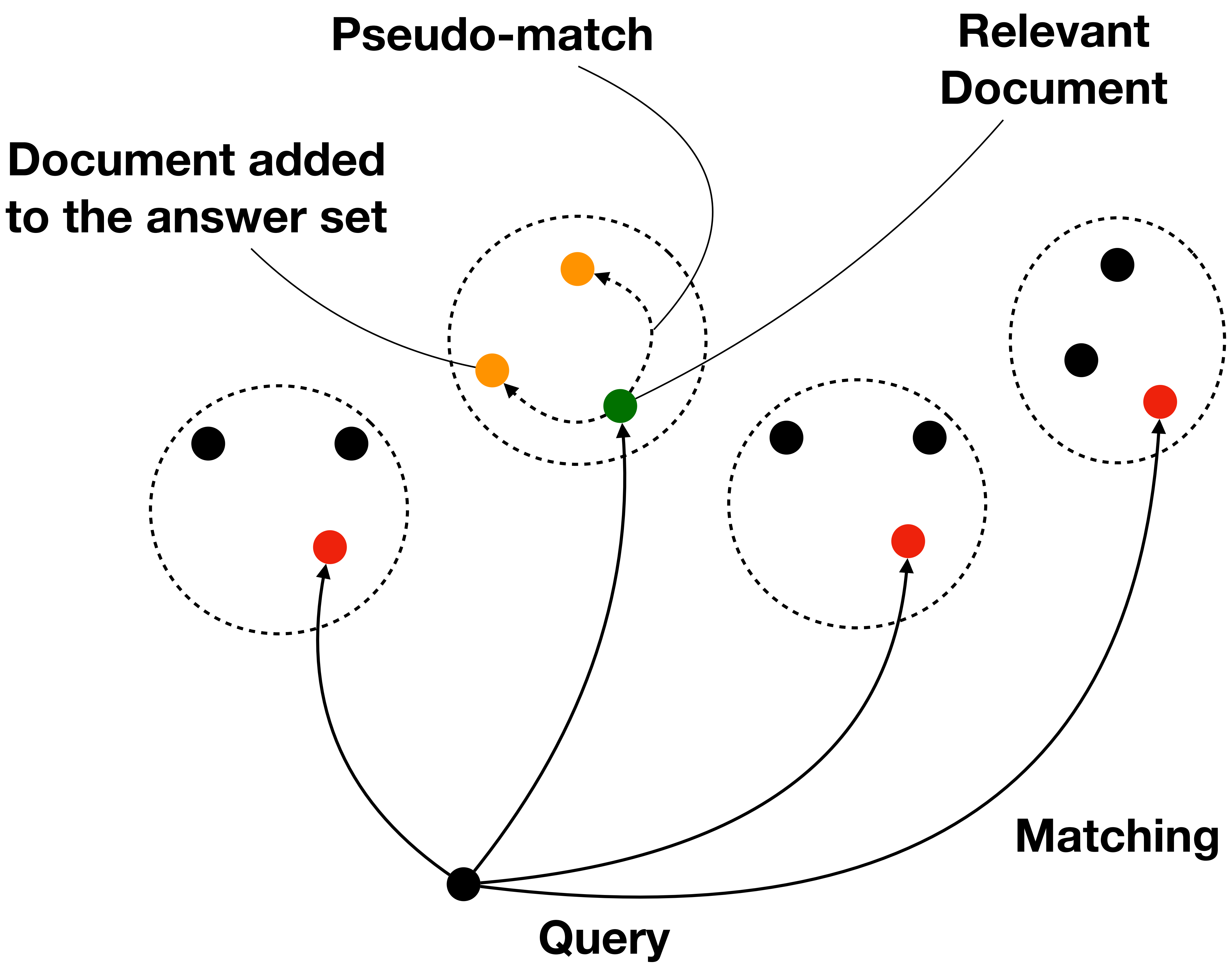
Cluster assignment
- **Hard clustering:** each object is part of exactly one cluster usually with a boolean measure of assignment
- **Soft clustering:** the assignment of an object is a distribution over the clusters and each object has a fractional membership in several clusters



Clustering as a support to document search

Terminological analysis
By analyzing the specific terminology of each cluster, we can discover groups of terms that are related one to the other and use this for query expansion

Cluster pruning
Instead of matching a query q against all the documents in the corpus, we can select a set of **representative documents for each cluster** and **match the query only against those documents**. Then, documents in the same cluster of a relevant representative document are returned in the results.



Evaluation of clustering

Given $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ as the set of clusters and $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$ as the set of (expected) classes, we can check the quality of clustering by focusing on the idea that pairs of items appear to be in the same cluster when they are in the same reference class and viceversa.

$$(x_i, x_j)$$

$$\exists \omega_k : (x_i, x_j) \in \omega_k$$

True → $\exists c_z : (x_i, x_j) \in c_z$
- **True** → **True Positive (TP)**
- **False** → **False Positive (FP)**

False → $\exists c_z : (x_i, x_j) \in c_z$
- **True** → **False Negative (FN)**
- **False** → **True Negative (TN)**

Evaluation of clustering

Given $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ as the set of clusters and $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$ as the set of (expected) classes:

Rand Coefficients

$$Rand = \frac{TP + TN}{TP + FP + FN + TN}$$

Purity

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j | w_k \cap c_j |$$

Normalized Mutual Information

Where H denotes the entropy

$$NMI(\Omega, \mathbb{C}) = \frac{I(\Omega; \mathbb{C})}{(H(\Omega) + H(\mathbb{C}))/2}$$

$$I(\Omega; \mathbb{C}) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)} =$$

$$= \sum_k \sum_j \frac{| \omega_k \cap c_j |}{N} \log \frac{N | \omega_k \cap c_j |}{| \omega_k | | c_j |}$$

$$H(\Omega) = - \sum_k P(\omega_k) \log P(\omega_k) = - \sum_k \frac{| \omega_k |}{N} \log \frac{| \omega_k |}{N}$$

K-means

The objective of K-means is to minimize the average squared Euclidean distance of documents from their cluster centers

Randomly select and initial set of K objects as centers, called seeds
Each seed represents a cluster
Assign documents to the cluster with the closest centroid
Recompute centroids on the basis of the current objects in clusters
Repeat the last two point until termination

Termination may be:
- Fixed number of iterations
- Assignment of documents to clusters does not change (i.e., centroids do not change)
- RSS falls below a threshold

Cluster center

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

Residual sum of squares

$$RSS = \sum_{k=1}^{K} \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

Bradley, P. S., & Fayyad, U. M. (1998, July). Refining initial points for k-means clustering. In ICML (Vol. 98, pp. 91-99).

K-means

Note that RSS monotonically decreases also as K increases (reaching the minimum with K equal to the number of points)

Estimating K

The best tradeoff between RSS and K can be estimated by

$$K = argmin_k(min(RSS_k) + \lambda K)$$

Model-based clustering

We can generalize k-means by interpreting the **centroids as a model that generates the data**

The idea is that we can pick a **centroid and add some noise to generate a document**

Cluster shape depends on the kind of distribution of noise

In model-base clustering, we **assume that data were generated by a model and we try to observe data in order to find the model**. **Clusters** and **assignment** are then **latent parameters that we want to estimate**.

McNicholas, P. D. (2016). Model-based clustering. Journal of Classification, 33(3), 331-373.

Model-based clustering

$\Theta = \{\overrightarrow{\mu_1}, \overrightarrow{\mu_2}, \ldots, \overrightarrow{\mu_K}\}$ is the set of model parameters (the centroids to be found for k-means)

$L(D \mid \Theta)$ is the log-likelihood of having the data *D* generated by Θ (this quantity has to be maximized: **objective function**)

$$\Theta = argmax_{\Theta} L(D \mid \Theta) = argmax_{\Theta} \log \prod_{n=1}^{N} P(d_n \mid \Theta) = argmax_{\Theta} \sum_{n=1}^{N} \log P(d_n \mid \Theta)$$

Note that the assignment probability $P(d_n \mid \omega_k; \Theta)$ that can be computed having Θ implies that a document can be assigned to different clusters with different probabilities: **soft clustering**

Expectation Maximization algorithm

The EM algorithm is an iterative algorithm that maximizes $L(D \mid \Theta)$. EM can be used to find latent models in a variety of applications, not only clustering.

In the example, we use multivariate Bernoulli distributions for data, corresponding to representing documents as binary vectors over the dictionary.

$$P(d \mid w_k; \Theta) = \left(\prod_{t_m \in d} q_{mk} \right) \left(\prod_{t_m \notin d} (1 - q_{mk}) \right) \qquad \Theta_k = (\alpha_k, q_{1k}, \ldots, q_{Mk})$$

$q_{mk} = P(U_m = 1 \mid \omega_k)$ is the probability that a document from cluster ω_k contains term t_m

α_k is the prior of cluster ω_k, i.e., the probability of a document to be in ω_k not having any information about the document

Ingole, M. N., Bewoor, M. S., & Patil, S. H. (2012). Text summarization using expectation maximization clustering algorithm. International Journal of Engineering Research and Applications, 2(4), 168-171.

Expectation Maximization algorithm

In our model, we generate a document by picking a cluster ω_k with probability α_k and than we generate the terms with probability q_{mk}

We aim at finding a_k and q_{mk}

Similarly to k-means, this is done by:
- **Maximization step:** re-computation of the parameters of the model
- **Expectation step:** reassignment

Expectation Maximization algorithm

Maximization step

$$q_{mk} = \frac{\sum_{n=1}^{N} r_{nk} I(t_m \in d_m)}{\sum_{n=1}^{N} r_{nk}} \quad ; \quad \alpha_k = \frac{\sum_{n=1}^{N} r_{nk}}{N}$$

where $I(t_m \in d_n) = 1$ if $t_m \in d_m$, 0 otherwise. r_{nk} is the soft assignment of d_n to ω_k

Expectation step

$$r_{nk} = \frac{\alpha_k (\prod_{t_m \in d_n} q_{mk})(\prod_{t_m \notin d_n} (1 - q_{mk}))}{\sum_{k=1}^{K} \alpha_k (\prod_{t_m \in d_n} q_{mk})(\prod_{t_m \notin d_n} (1 - q_{mk}))}$$

Not easy to find good initial seeds: a chance is to use another clustering algorithm like k-means for finding the seeds and then EM for soften up the assignment

Affinity propagation clustering

Input: a similarity matrix among documents e.g., $s(i,k) = -\,||\,\vec{i} - \vec{k}\,||^2$

Input: special values for $s(k,k)$, where larger values are more likely to be chosen as exemplars for clusters (preferences)

Documents exchange messages that can be combined to decide which items are exemplars and which items need to be associated with one of the exemplars

Affinity propagation clustering

Responsibility

$r(i, k)$ sent from *i* to *k* denotes how well *k* is an exemplar for *i*

Availability

$a(i, k)$ sent from candidate exemplar *k* to *i* denotes how appropriate is for *i* to choose *k* as exemplar

Procedure

Initially set availabilities to 0 and then recompute iteratively responsibilities and availabilities

Affinity propagation clustering

Responsibility

$$r(i,k) = s(i,k) - max_{k's.t.k' \neq k}\{a(i,k') + s(i,k')\}$$

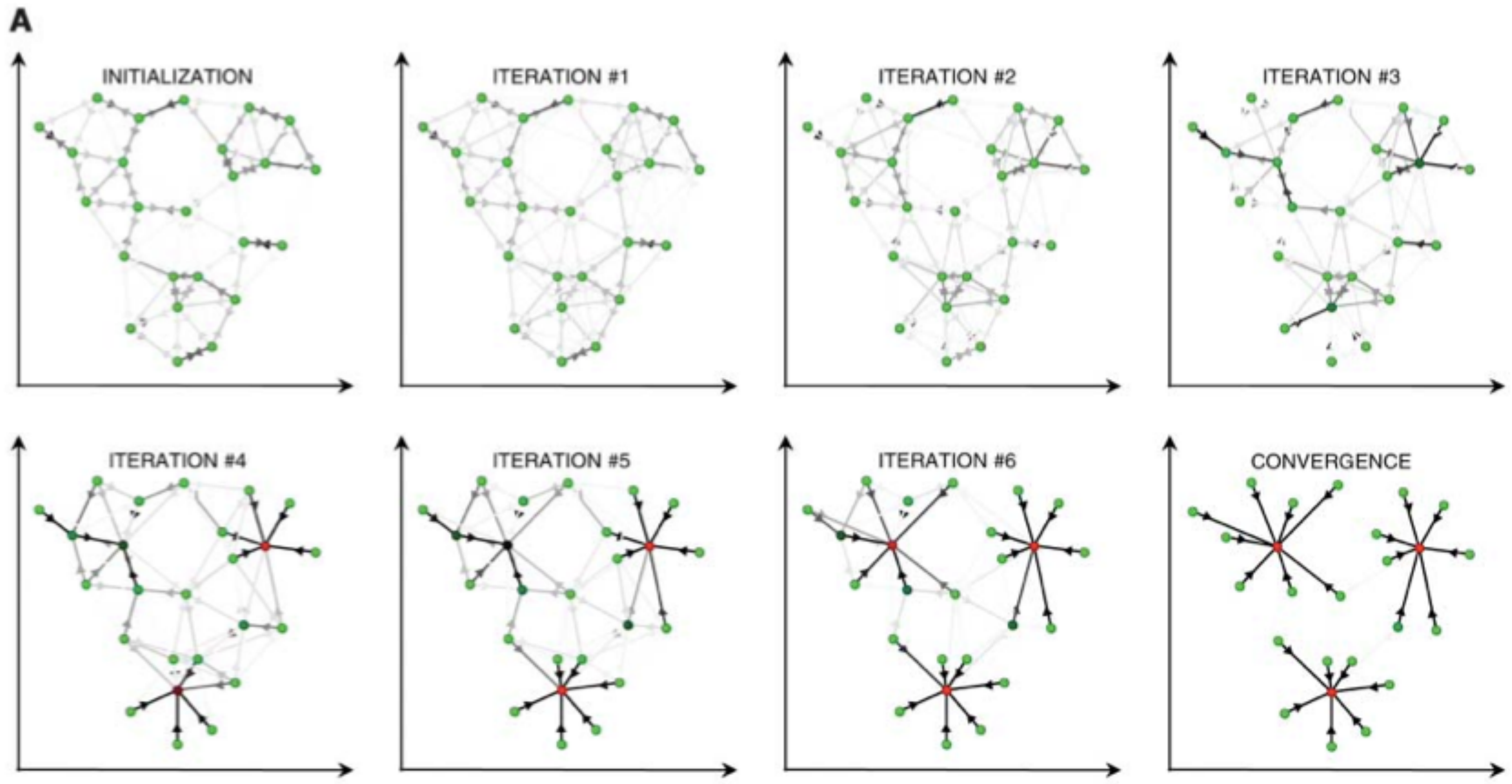
Availability

$$a(i,k) = min\left\{0, r(k,k) + \sum_{i's.t.i' \notin \{i,k\}} max\{0, r(i',k)\}\right\}$$

Self availability

$$a(k,k) = \sum_{i's.t.i' \neq k} max\{0, r(i',k)\}$$

At any point, for a point I the value of k that maximize a(i, k) + r(i, k) either identifies i as an exemplar if i = k or identifies the data point k that is the exemplar for i



Use a dumping factor λ. Each message is set to λ times its values from the previous iteration plus $1 - \lambda$ times its prescribed value

Frey, Brendan J., and Delbert Dueck. "Clustering by passing messages between data points." science 315.5814 (2007): 972-976.

Hierarchical clustering

Hierarchical clustering produces a hierarchy of clusters. Does not require to specify the number of clusters (but a criterion of optimal cluster selection).

Most hierarchical clustering algorithms are deterministic. Complexity is at least quadratic in the number of documents.

Bottom-up clustering or agglomerative
Clustering treat each document as a singleton cluster and successively merge clusters until a single cluster containing all the documents has created

Top-down clustering or divisive
Clustering starts from a single cluster containing all documents and recursively splits clusters until singleton clusters are formed

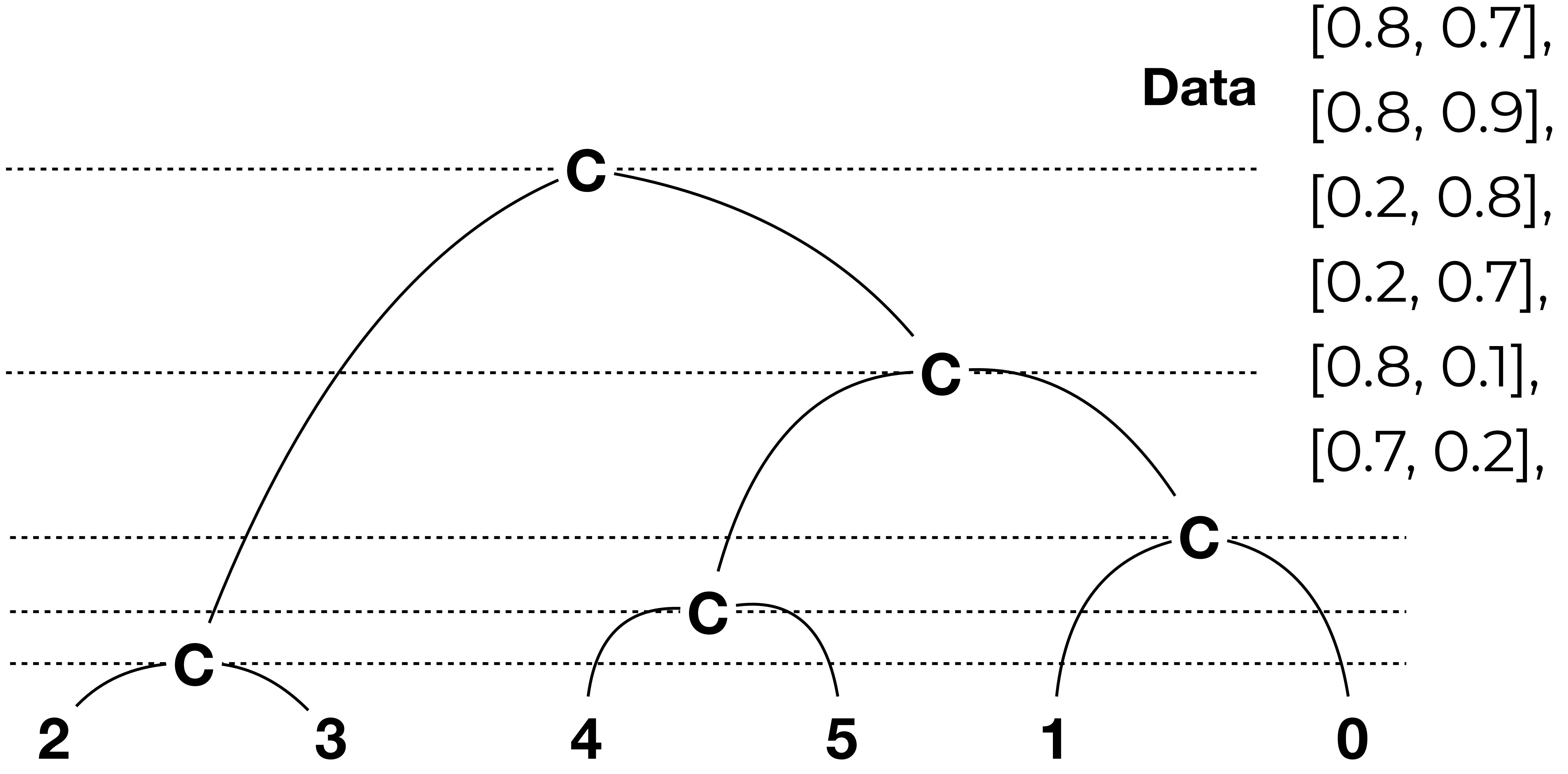
Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(1), 86-97.

Hierarchical clustering

The clustering procedure aggregates items starting with the most similar.

Each cluster that is defined this way, substitutes the corresponding items in the dataset.

Then, we need a method for comparing single items with clusters and clusters with other clusters in order to define their similarity for the subsequent steps



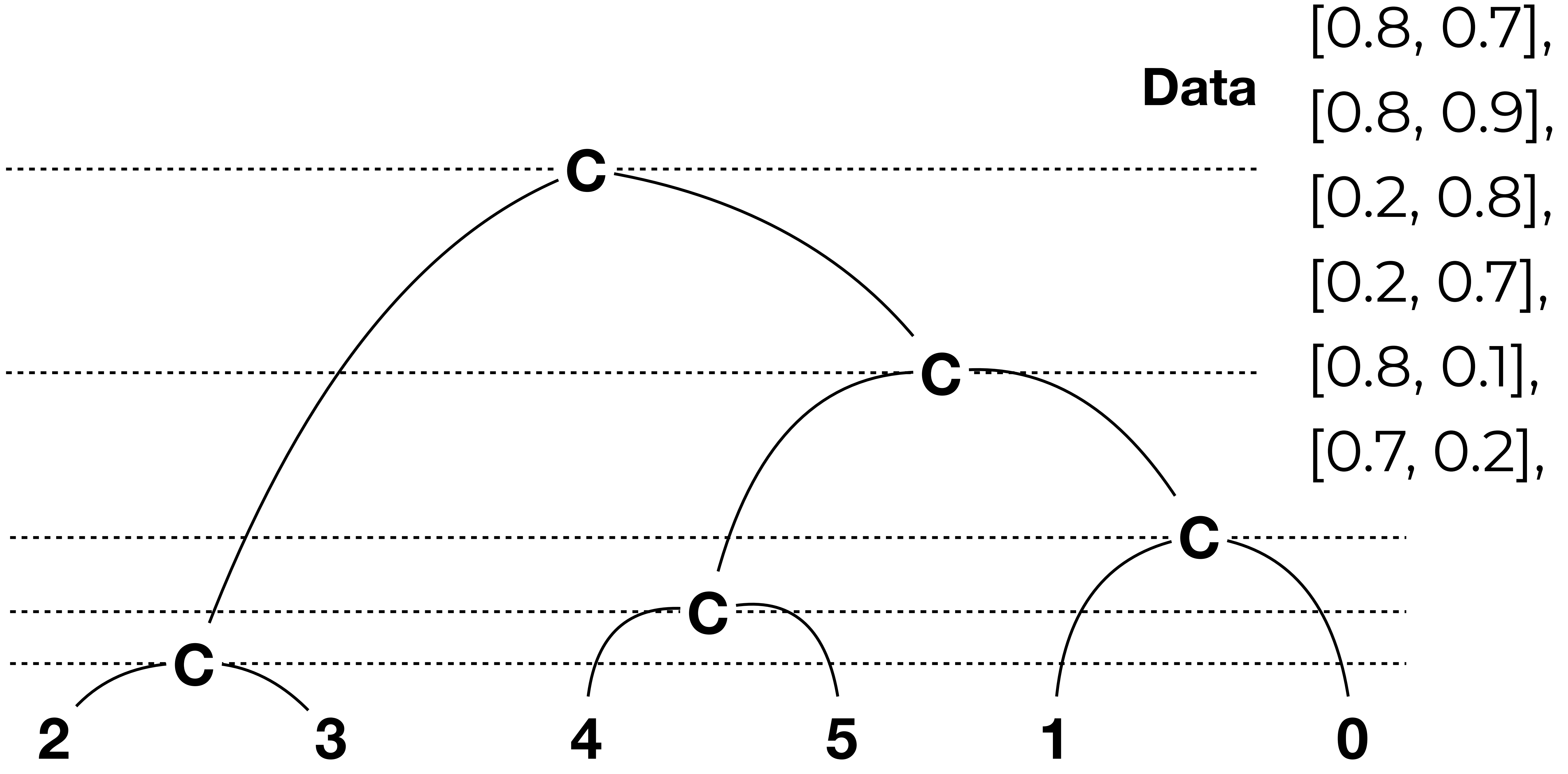
How to select clusters

Cut the dendogram at a specific level of similarity

Cut the dendogram where the gap between two successive combination similarities is largest

Apply $K = argmin_{K'}[RSS(K') + \lambda K']$ where K' is the cut threshold resulting in K clusters

Pre-define K and cut the dendogram accordingly



Computing clustering similarity

There are different stategies for computing the similarity between clusters for the merging step:

Single link: $d(u, v) = min(u[i], v[j])$

Complete link: $d(u, v) = max(u[i], v[j])$

Average link: $d(u, v) = \sum_{ij} \frac{d(u[i], v[j])}{|u||v|}$

Weighted link: $d(u, v) = (d(s, v) + d(t, v))/2$, where u was formed with cluster s and t and v is a remaining cluster in the forest.

Centroid link: $d(u, v) = || \vec{u} - \vec{v} ||^2$

Median link: same as centroid, but the new centroid of a cluster is the average of the two centroids

Ward link: $d(u, v) = \sqrt{\frac{|v| + |s|}{|v| + |s| + |t|} d(v, s)^2 + \frac{|v| + |t|}{|v| + |s| + |t|} d(v, t)^2 + \frac{|v|}{|v| + |s| + |t|} d(s, t)^2}$

Top-down clustering

Create a single cluster grouping all the documents

Apply a **flat hard clustering solution** (e.g., k-means) for **splitting the cluster in K partitions**

Recursively apply the partition step until all the clusters contain only one document

Keep track of the **parent of each cluster obtained by partition**

Other clustering algorithms

Mean-shift clustering

In Mean-shift we aim at moving the centroids representing clustering according to the distribution of points.

Given a candidate centroid x_i for iteration *t*, the candidate is updated according to the following equation:

$$x_i^{t+1} = x_i^t + m(x_i^t), \qquad m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

Where $N(x_i)$ is the neighborhood of samples within a given distance around x_i and *m* is the mean shift vector that is computed for each centroid that points towards a region of the maximum increase in the density of points.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. IEEE transactions on pattern analysis and machine intelligence, 17(8), 790-799.

Spectral clustering

Given a set of documents $D = d_1, \ldots, d_n$, define the affinity matrix *A* defined by

$A_{ij} = exp\left(\frac{-||d_i - d_j||^2}{2\sigma^2}\right)$ if $i \neq j$ and $A_{ij} = 0$ otherwise

Define D to be the **diagonal matrix** whose *(i,i)* element is the sum of A *i*th row and build $L = D^{-1/2}AD^{-1/2}$

Form the matrix $X = [x_1, x_2, \ldots, x_k]$ where $x_1, \ldots, x_k$ are the *k* largest eigenvectors of *L*. Take the unit length.

Execute KMeans over *X* and assign the original document d_i to cluster *j* if row *i* of *X* was assigned to *j*.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems, 2, 849-856.

DBSCAN

The DBSCAN algorithm views clusters as areas of high density separated by areas of low density.

A point p is a **core point** if at least m points are within distance ϵ from it. Those points are said to be **directly reachable** from p.

A point q is reachable from p if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. ACM Transactions on Database Systems (TODS), 42(3), 1-21.

DBSCAN

For each point *p*, we:

- we create the region of points within distance ϵ from *p*
- if the size of the region is less than *m* (i.e., *p* is not a core point) then *p* is discarded
- else, we create a cluster C for *p* and we examine all the points *p'* in its region
- we take the region of *p'* and, if it is larger than *m*, we merge it with the region of *p*
- we add *p'* to C if *p'* is not in a cluster yet

Issues:

- from which points to start with
- highly dependent from the parameters m and ϵ

Cluster labeling

A general problem for any clustering approach is to generate a good set of labels for cluster description

Differential labeling: select cluster labels by comparing the distribution of terms in one cluster with that of other clusters. To this end, we can use specificity measures like *mutual information*, χ^2 test, Kullback–Leibler divergence, TfIdf

Internal labeling: uses information internal to each cluster for labeling (e.g., title of document which is closest to the cluster centroid, list of terms with high weights in the centroid of cluster)