Evaluation of Hierarchical Clustering Algorithms for Document Datasets

Paper by Ying Zhao and George Karypis
University of Minnesota (2002)

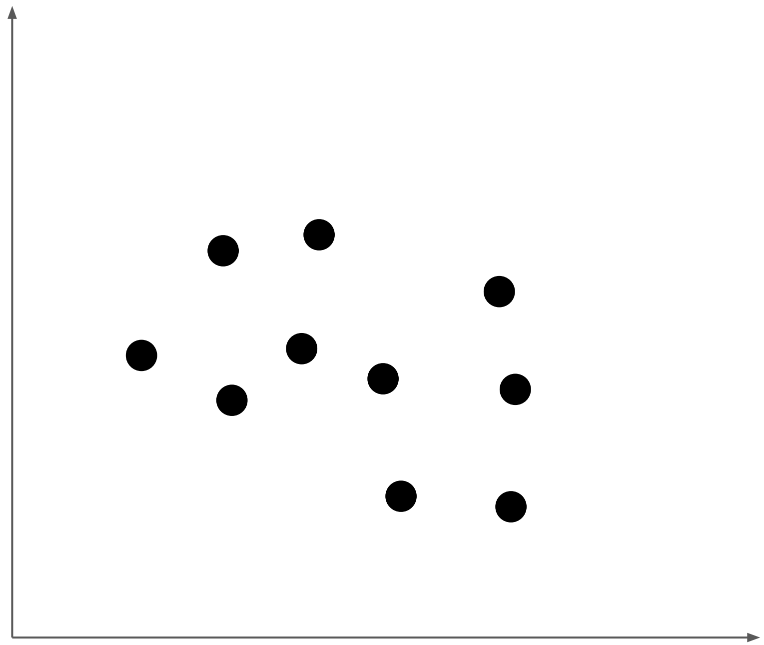
CS 6501 Paper Presentation - April 6, 2016

Matthew Hawthorn, Nikhil Mascarenhas, Shannon Mitchell

Motivation

- Hierarchical clustering of documents
  - Intuitive, clustering of different levels of granularity.
- Two major approaches
  - Partitional
  - Agglomerative
- General view was that partitional algorithms are inferior
- Authors ran an experiment to compare these approaches.
- Defined a new algorithm, a hybrid “constrained agglomerative algorithm”

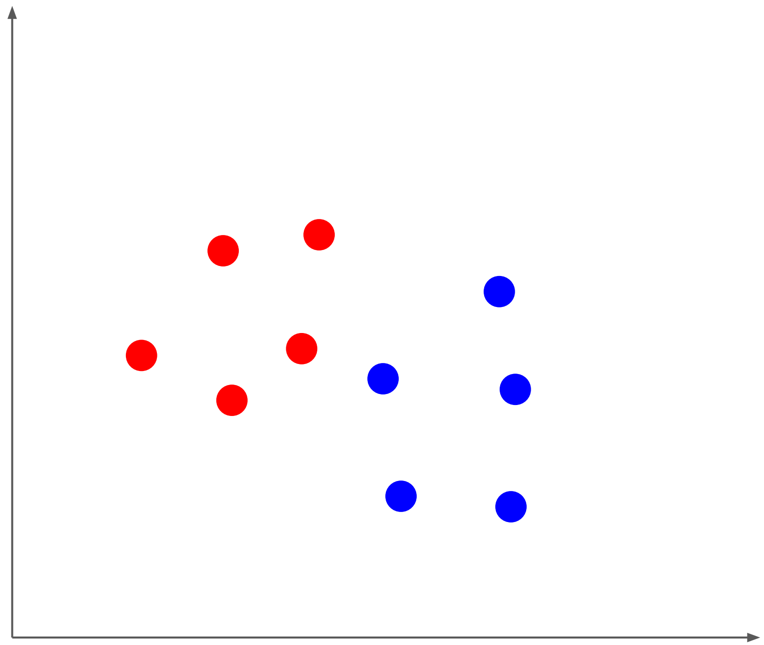
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves

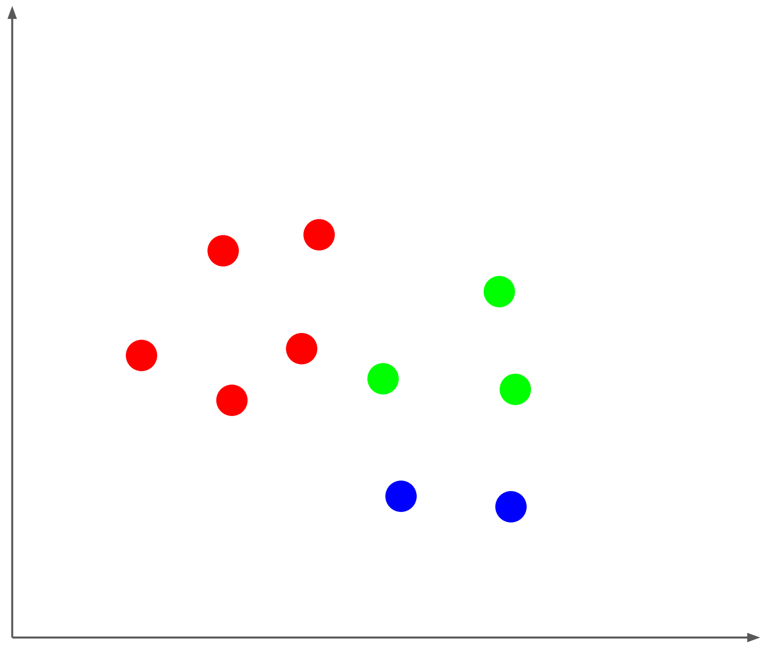
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

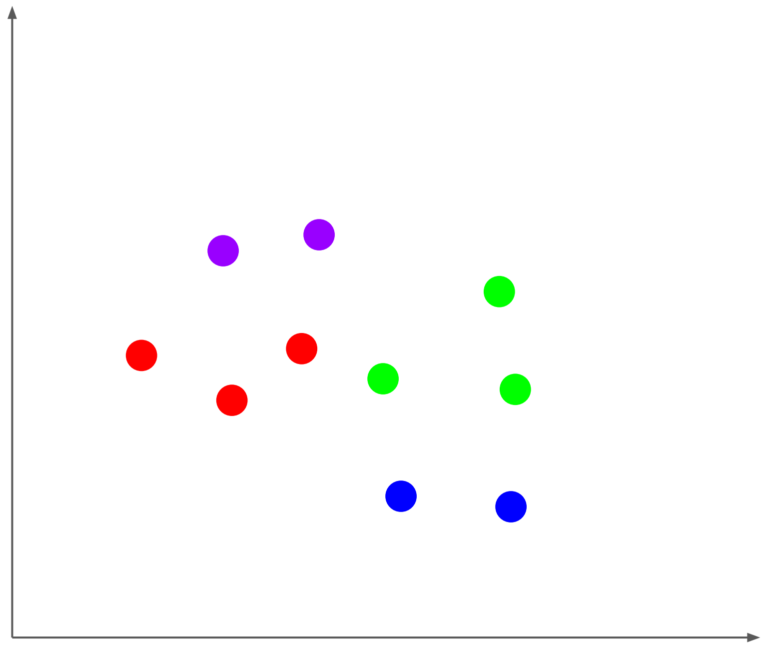
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

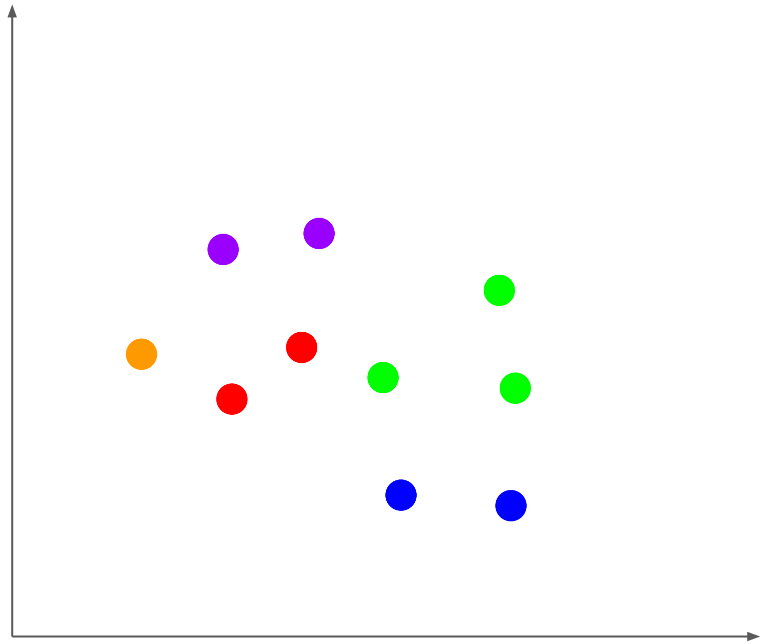
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

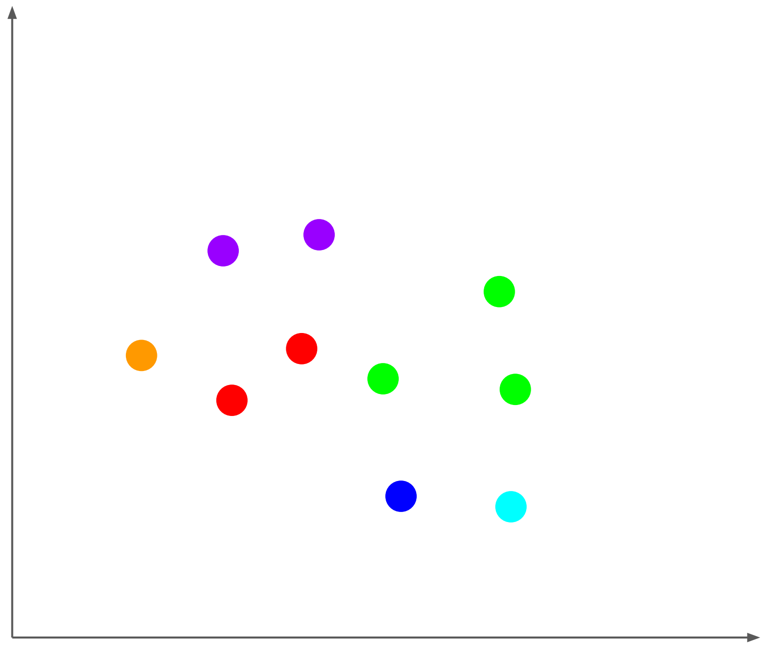
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

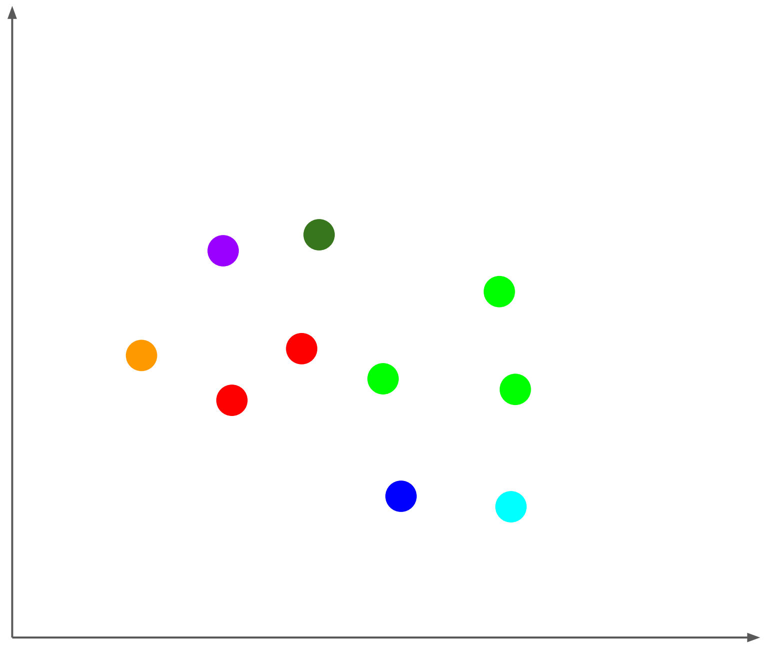
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

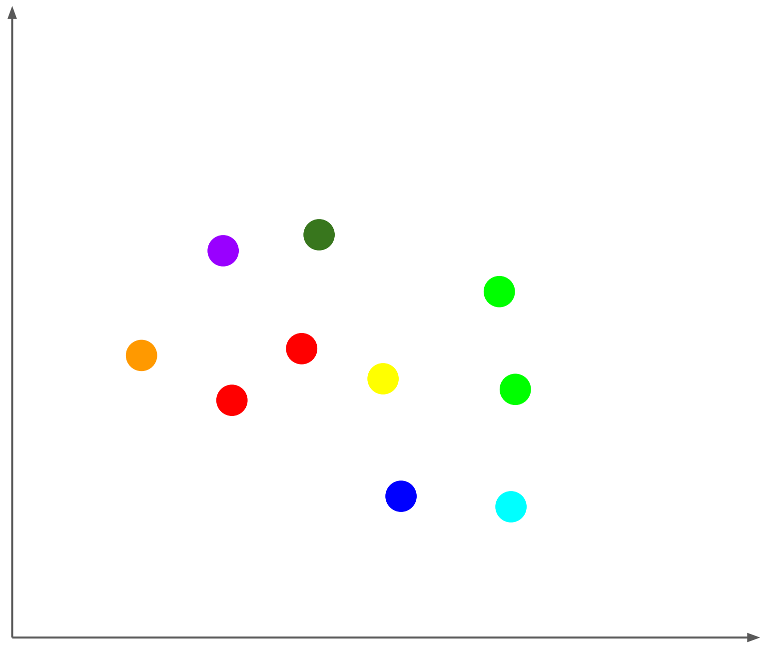
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

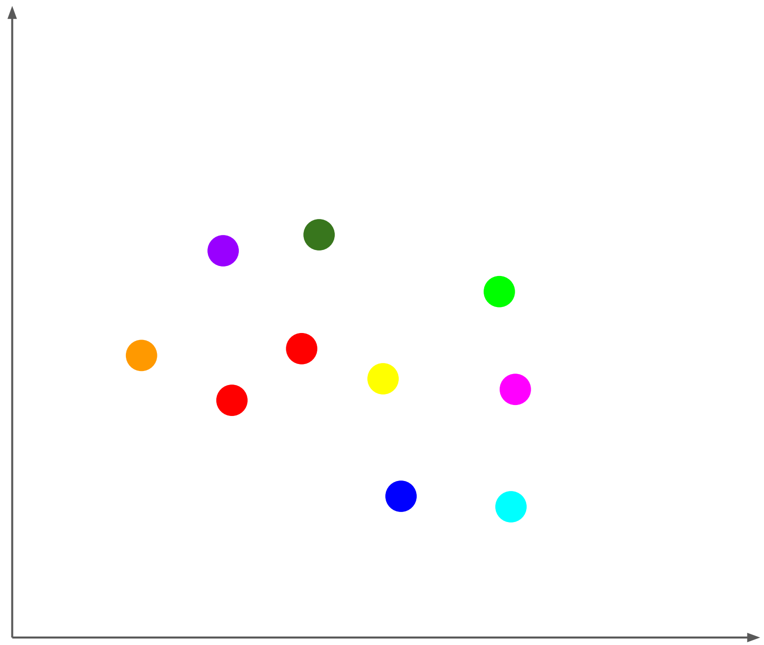
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

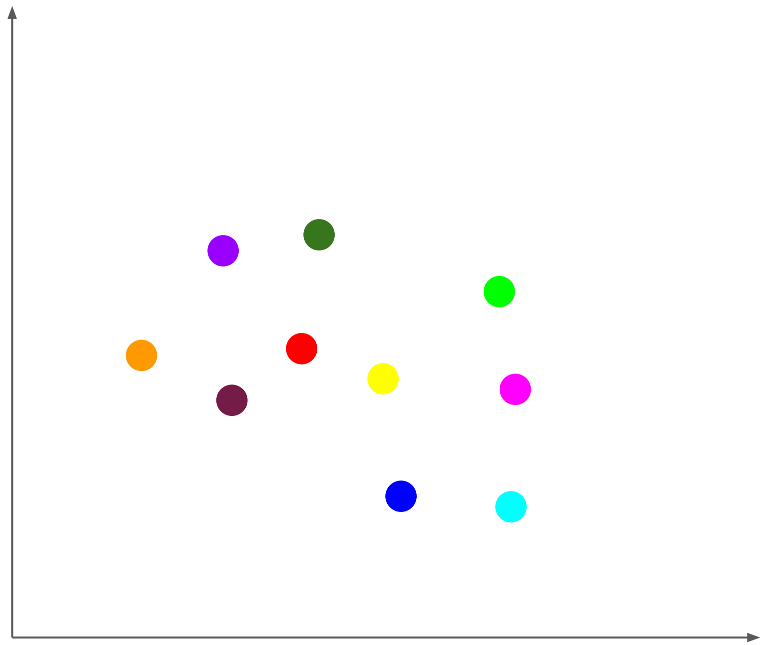
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

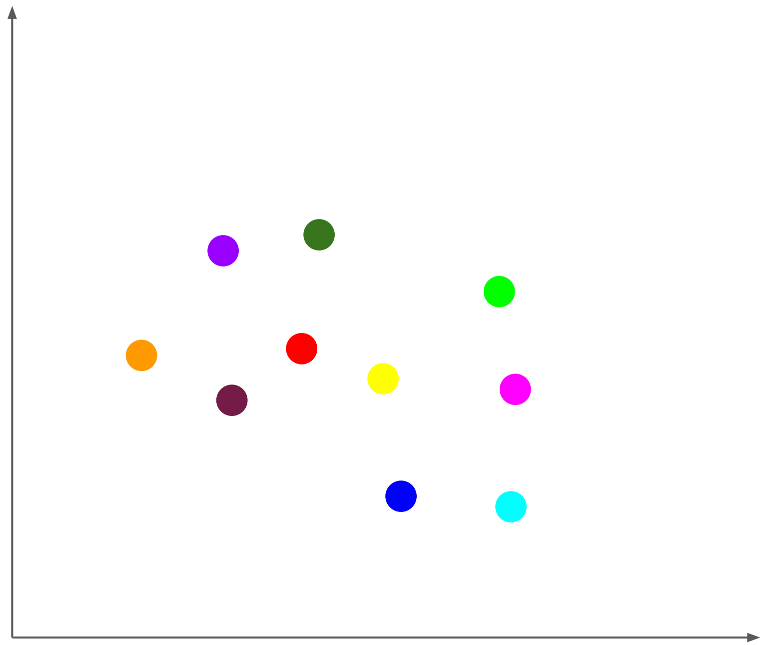
Hierarchical Clustering: Partitional Algorithms



Top-down

- Start with one cluster with all documents
- Start at root, divide down to leaves
- Split the cluster which most improves the criterion function
- Complexity: O (n log n)

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

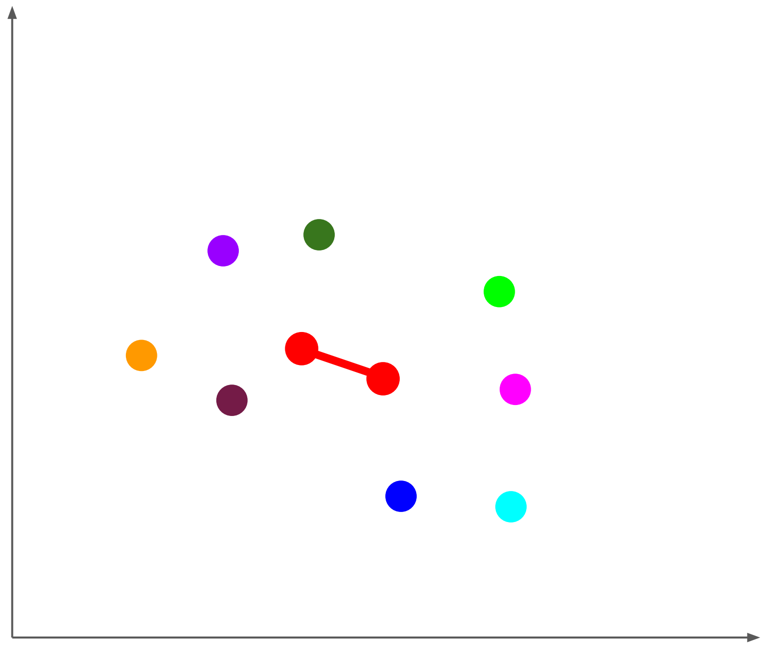
  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

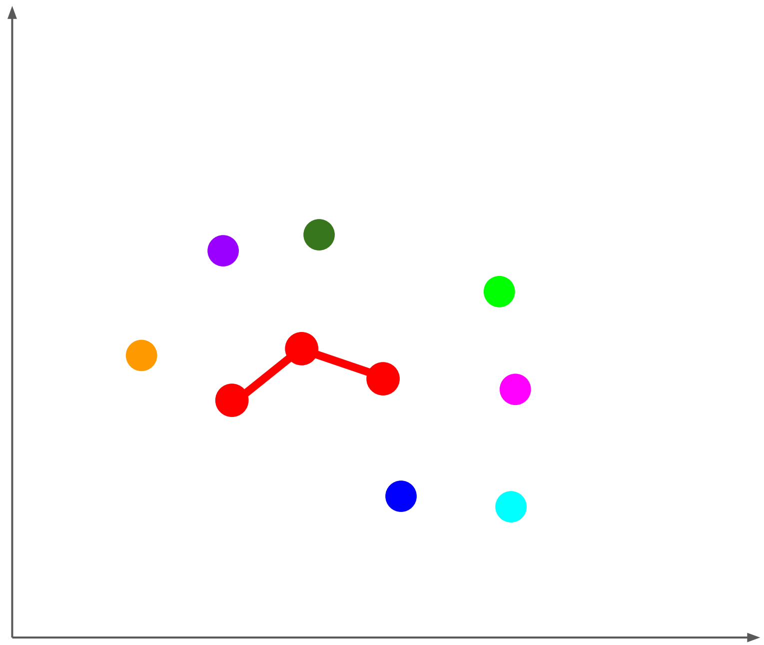
  $O(n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

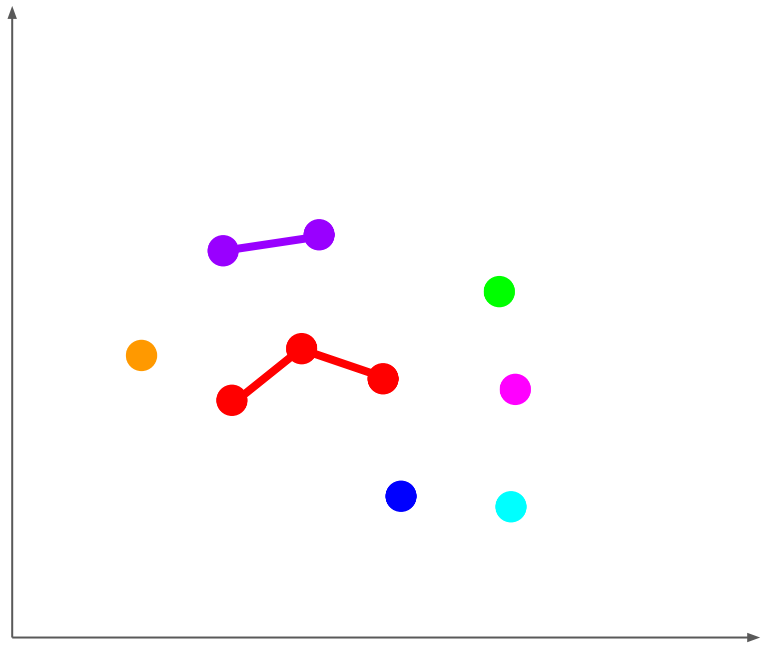
  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

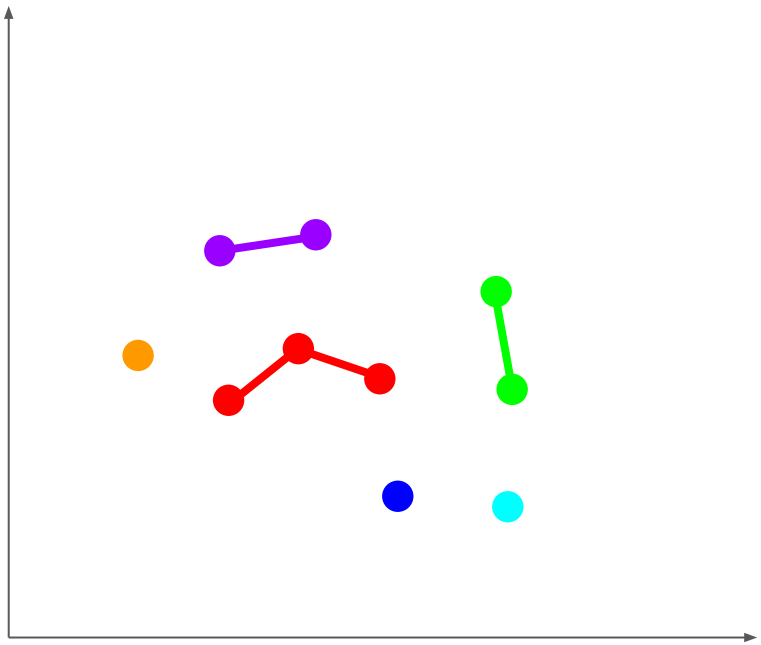
  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

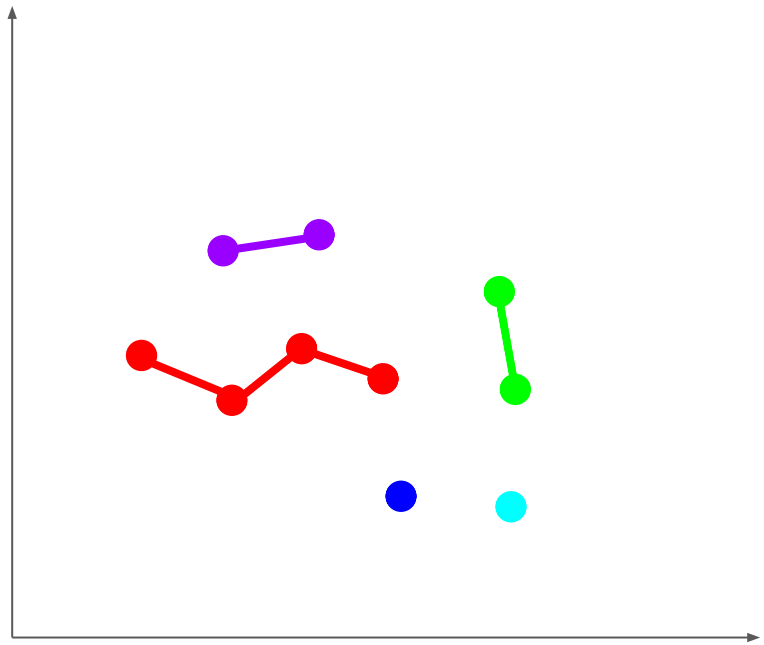
  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

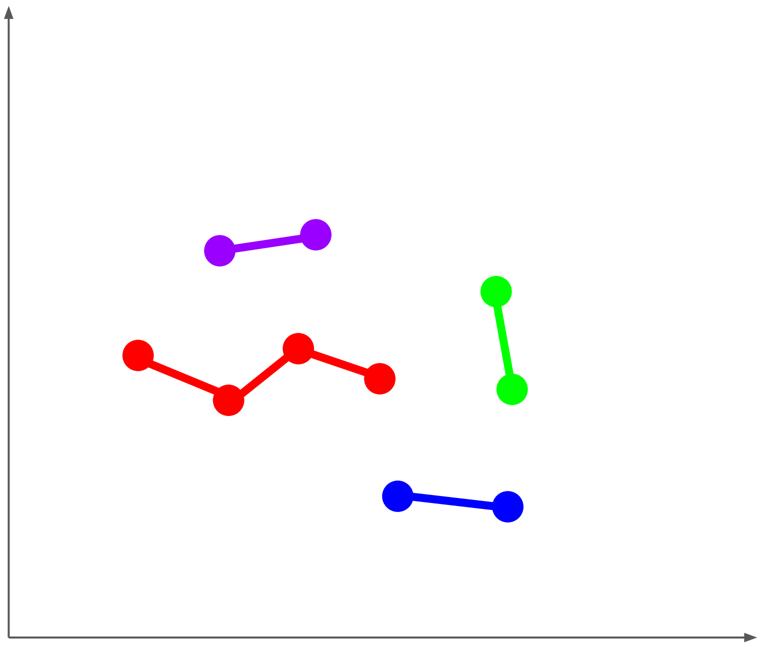
  $O(n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

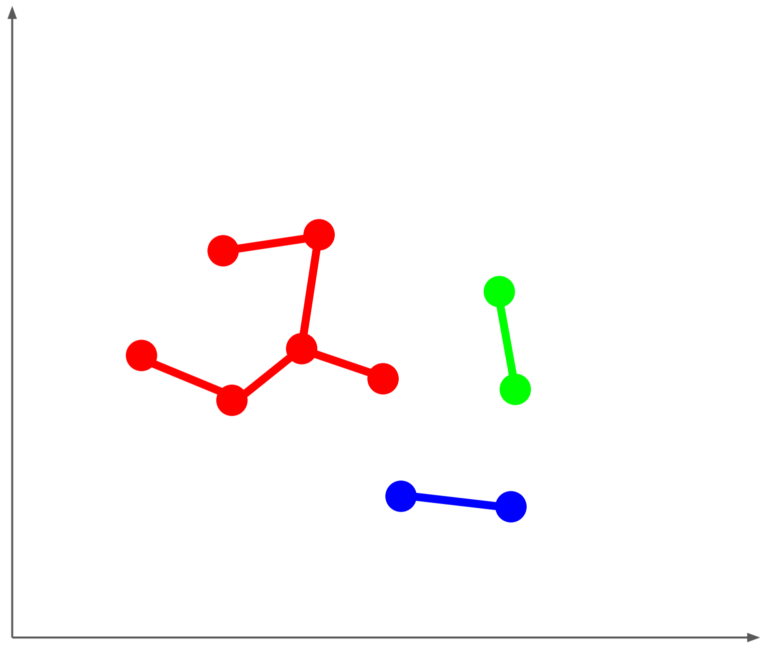
  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

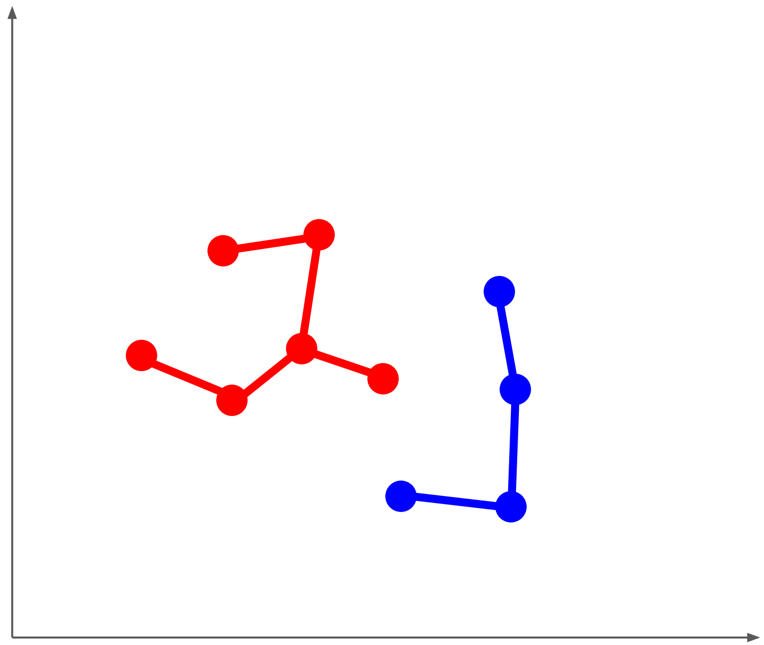
  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

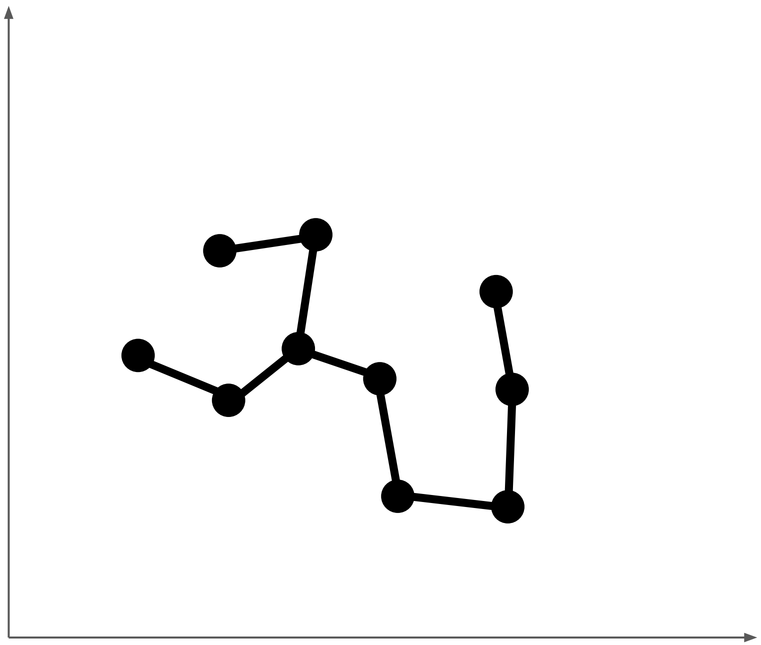
  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Hierarchical Clustering: Agglomerative Algorithms



Bottom-up

- Each document starts as its own cluster
- Start at leaves, merge to root
- Complexity:

  $O\ (n^2 \log n)$

  When caching of intermediate values of the objective is possible

  $O(n^3)$

  Otherwise

Criterion Functions

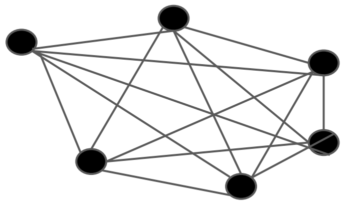
Global criterion functions drive the clustering process.

Internal Functions	External Functions	Graph Based Functions	Hybrid Functions
Considers only documents within a cluster	Considers how various clusters are different from each other.	Constructs a graph which represents the relationships between documents.	Simultaneously consider internal and external criterion functions

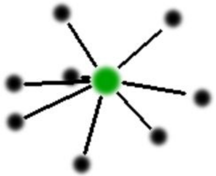
Internal Criterion Functions

m	Number of terms
n	Number of documents
k	Number of clusters
$S_1, S_2, \ldots S_k$	Each one of k clusters
$n_1, n_2, \ldots n_k$	Size of each cluster
$d_1, d_2, \ldots d_n$	Tf idf vector for a document
D_A	Sum of all vectors in cluster A
C_A	Centroid vector of cluster A

$$\mathcal{I}_1 = \sum_{r=1}^{k} n_r \left(\frac{1}{n_r^2} \sum_{d_i, d_j \in S_r} \cos(d_i, d_j) \right) = \sum_{r=1}^{k} \frac{\| D_r \|^2}{n_r}.$$



$$\mathcal{I}_2 = \sum_{r=1}^{k} \sum_{d_i \in S_r} \cos(d_i, C_r). = \sum_{r=1}^{k} \| D_r \|.$$

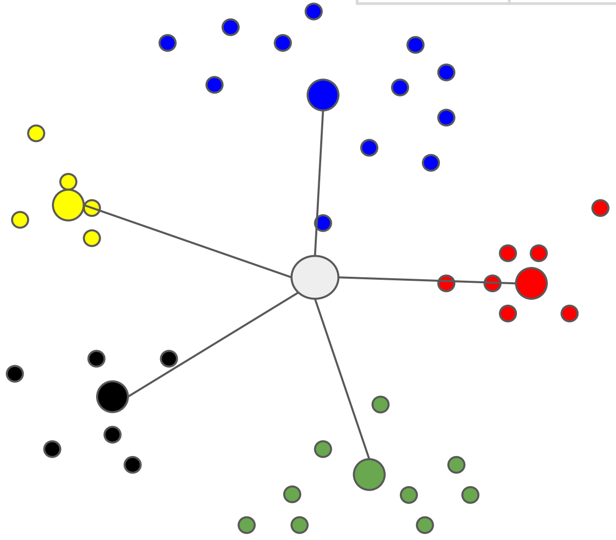


External Criterion Functions

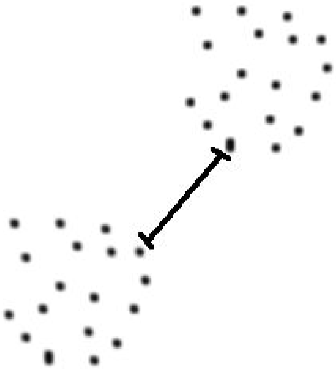
m	Number of terms
n	Number of documents
k	Number of clusters
$S_1, S_2, \ldots S_k$	Each one of k clusters
$n_1, n_2, \ldots n_k$	Size of each cluster
$d_1, d_2, \ldots d_n$	Tf idf vector for a document
D_A	Sum of all vectors in cluster A
C_A	Centroid vector of cluster A

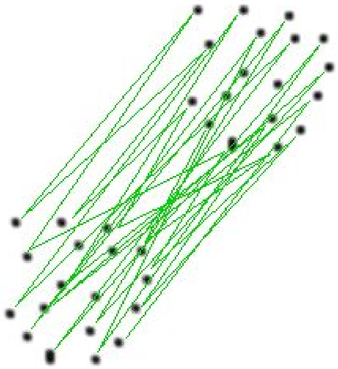
$$\sum_{r=1}^{k} n_r \cos(C_r, C) = \frac{1}{\|D\|}\left(\sum_{r=1}^{k} n_r \frac{{D_r}^t D}{\|D_r\|}\right)$$

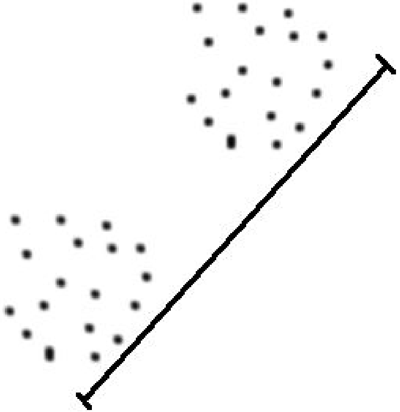
$$\mathcal{E}_1 = \sum_{r=1}^{k} n_r \frac{{D_r}^t D}{\|D_r\|}.$$



Traditional Agglomerative Clustering Criteria







	Single-linkage	Group average	Complete-linkage
	minimum distance	*average* of distances	*maximum* distance
Authors' abbreviation:	'slink'	'UPGMA'	'clink'

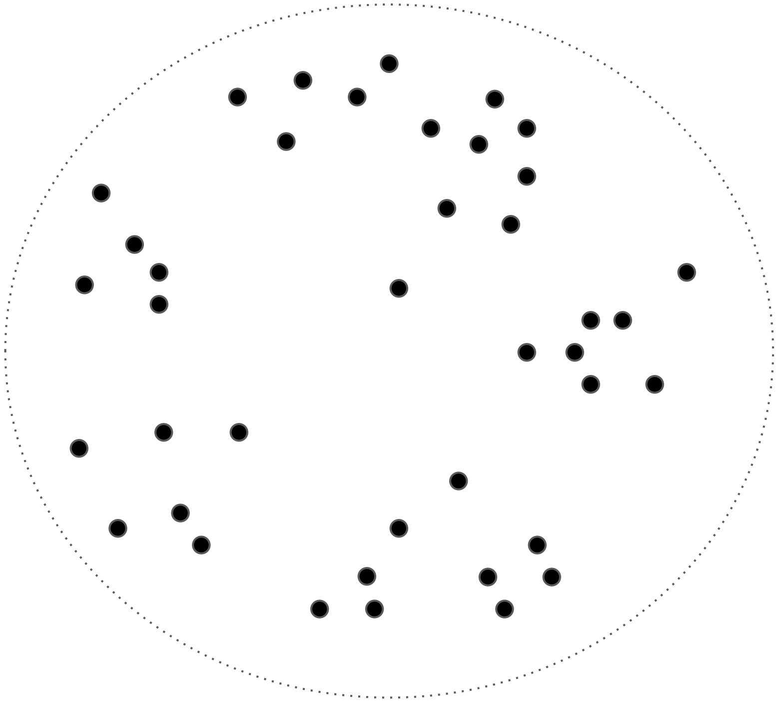
Hierarchical Clustering: Constrained Agglomerative

- Hybrid technique
- Constrains agglomerative clustering by initializing with intermediate hierarchical partitional clustering
- More likely to avoid early merge mistakes of agglomerative techniques

- But takes advantage of the ease with which agglomerative techniques find small and cohesive clusters

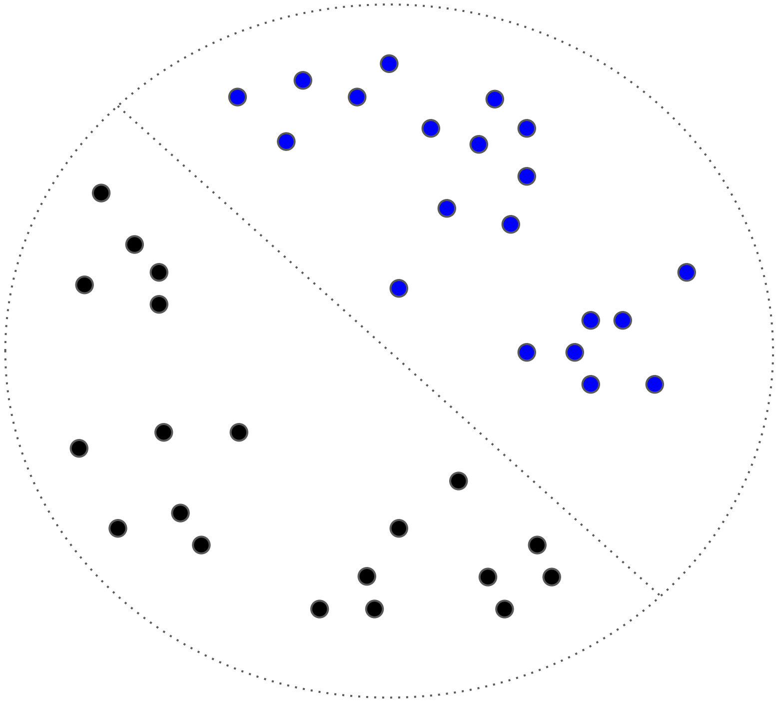
Hierarchical Clustering: Constrained Agglomerative

1. Find k clusters using partitional clustering



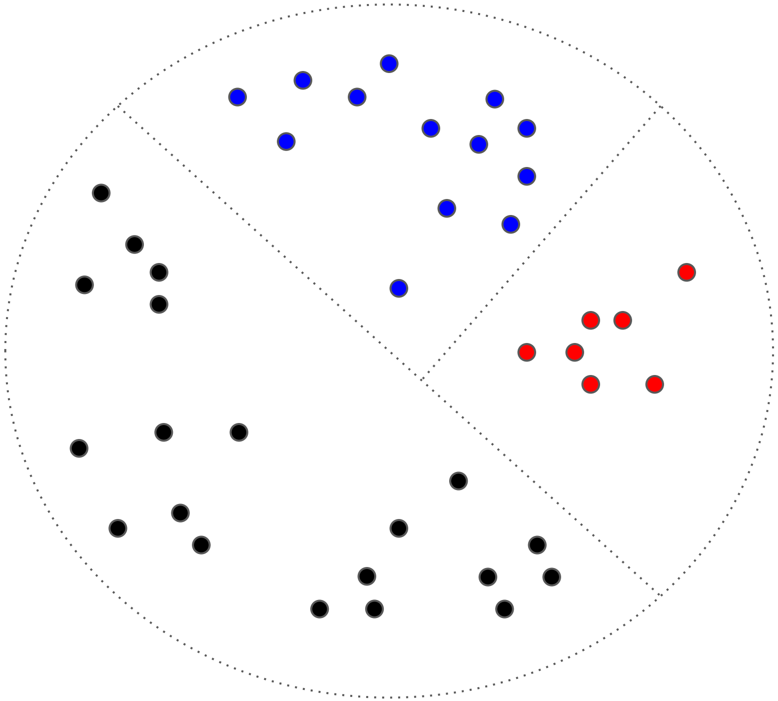
Hierarchical Clustering: Constrained Agglomerative

1. Find k clusters using partitional clustering



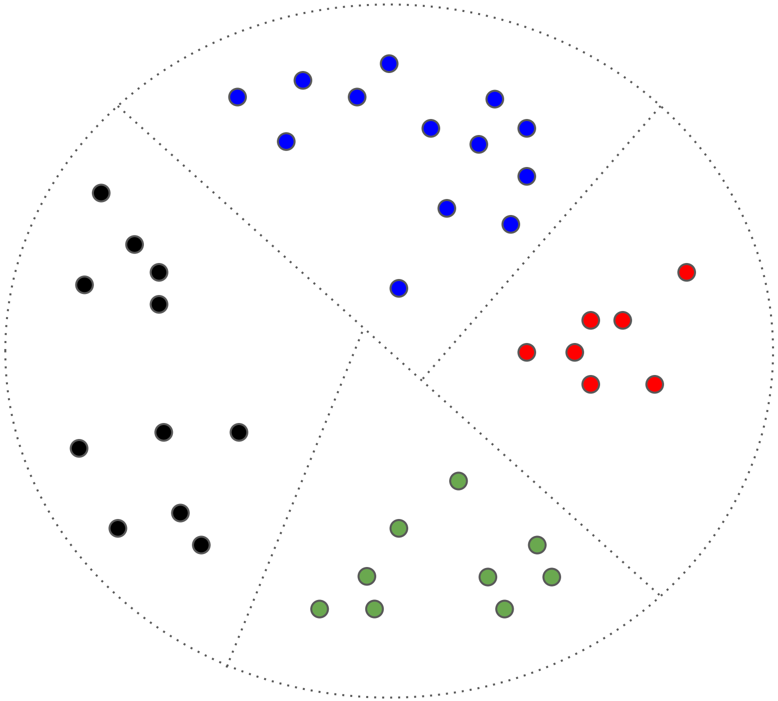
Hierarchical Clustering: Constrained Agglomerative

1. Find k clusters using partitional clustering



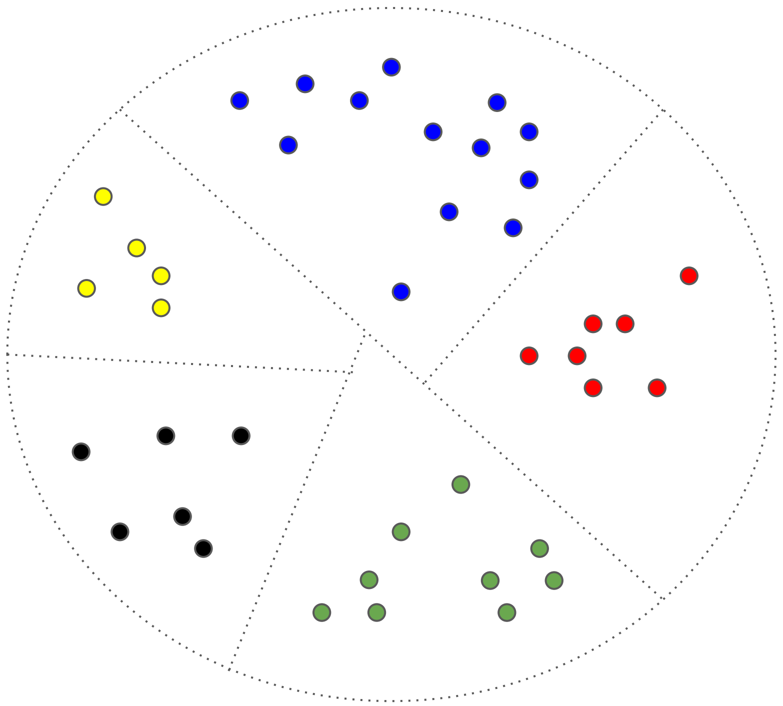
Hierarchical Clustering: Constrained Agglomerative

1. Find k clusters using partitional clustering



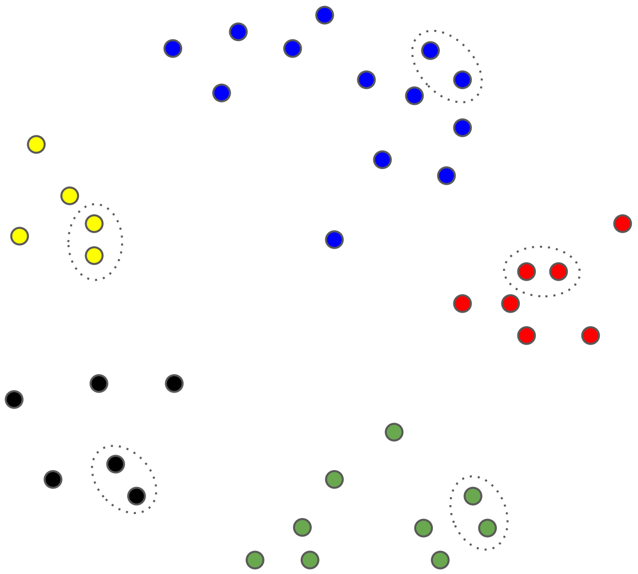
Hierarchical Clustering: Constrained Agglomerative

1. Find k clusters using partitional clustering



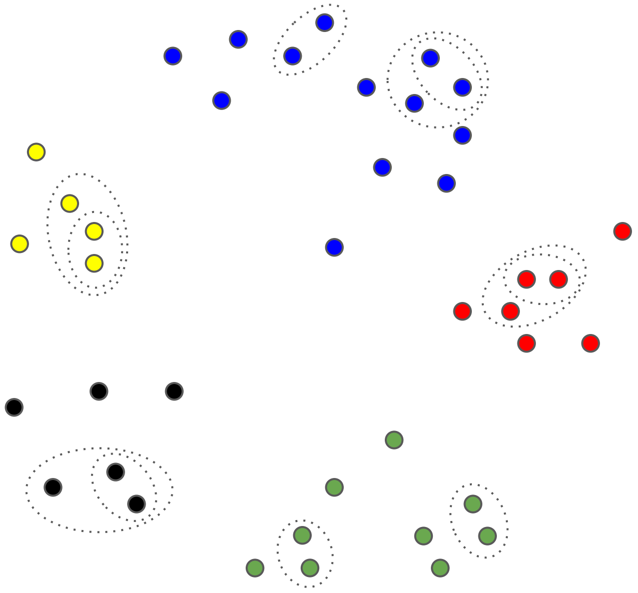
Hierarchical Clustering: Constrained Agglomerative

2. Cluster the documents in these clusters using agglomerative clustering



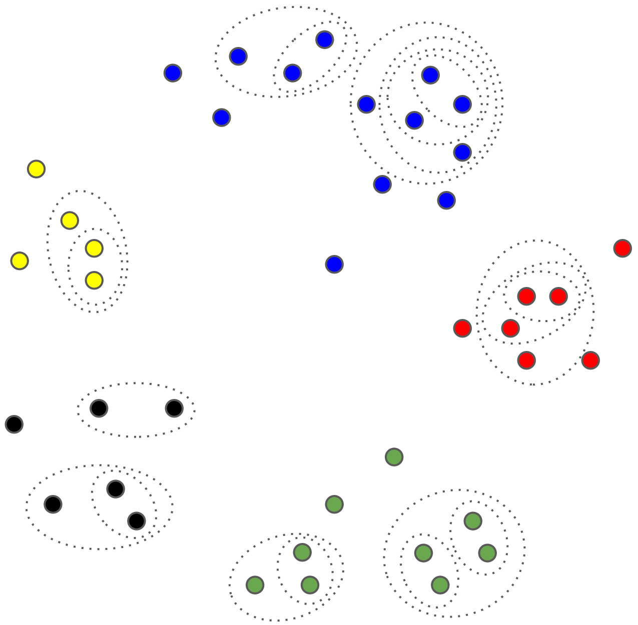
Hierarchical Clustering: Constrained Agglomerative

2. Cluster the documents in these clusters using agglomerative clustering



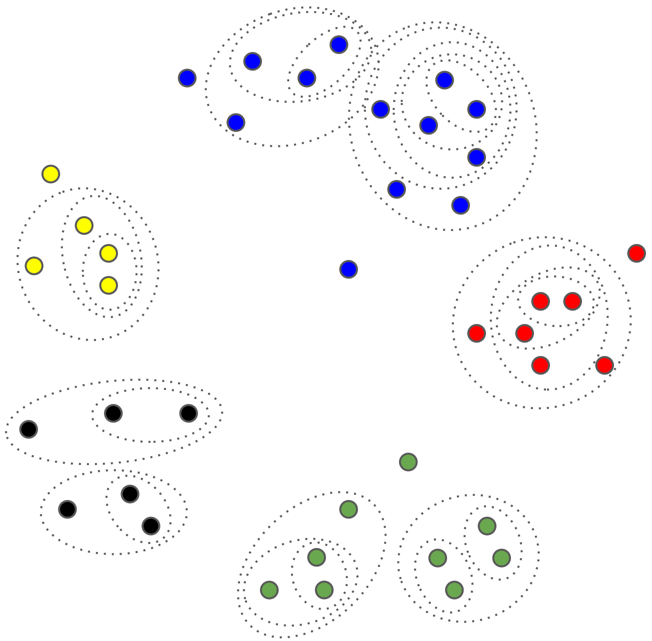
Hierarchical Clustering: Constrained Agglomerative

2. Cluster the documents in these clusters using agglomerative clustering



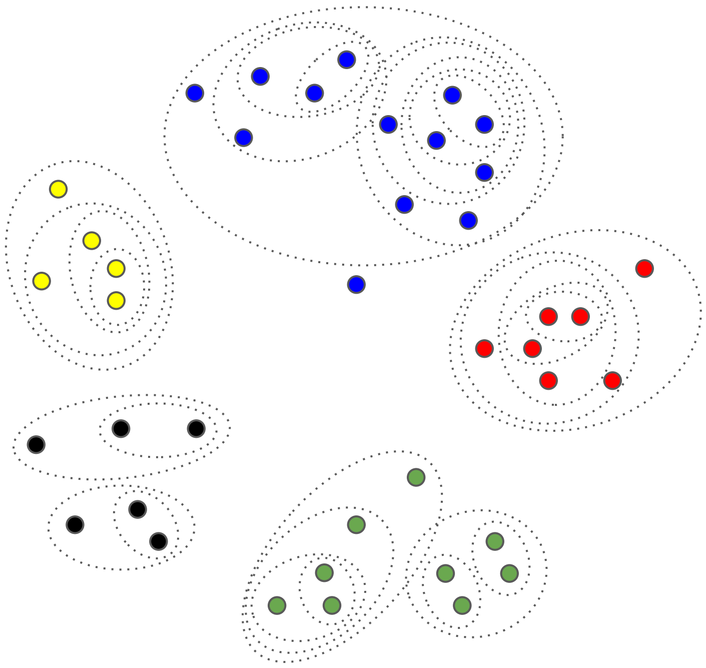
Hierarchical Clustering: Constrained Agglomerative

2. Cluster the documents in these clusters using agglomerative clustering



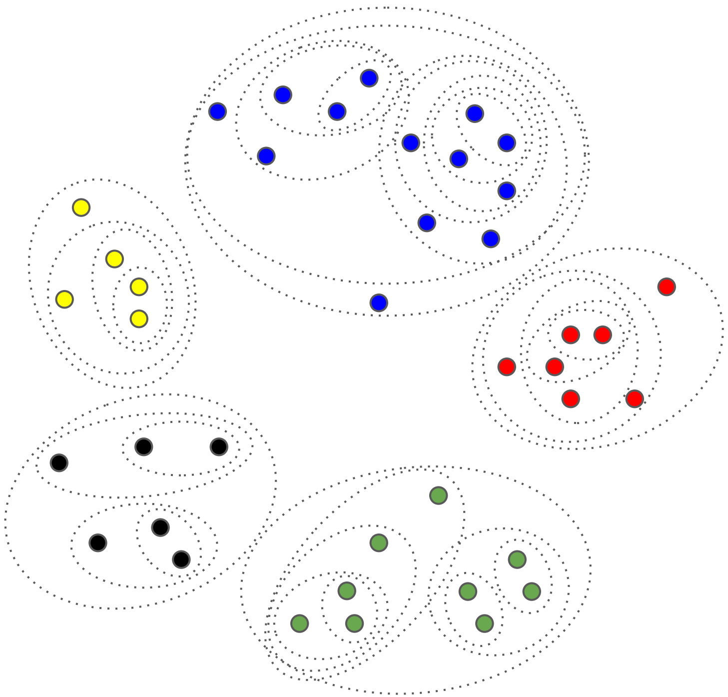
Hierarchical Clustering: Constrained Agglomerative

2. Cluster the documents in these clusters using agglomerative clustering



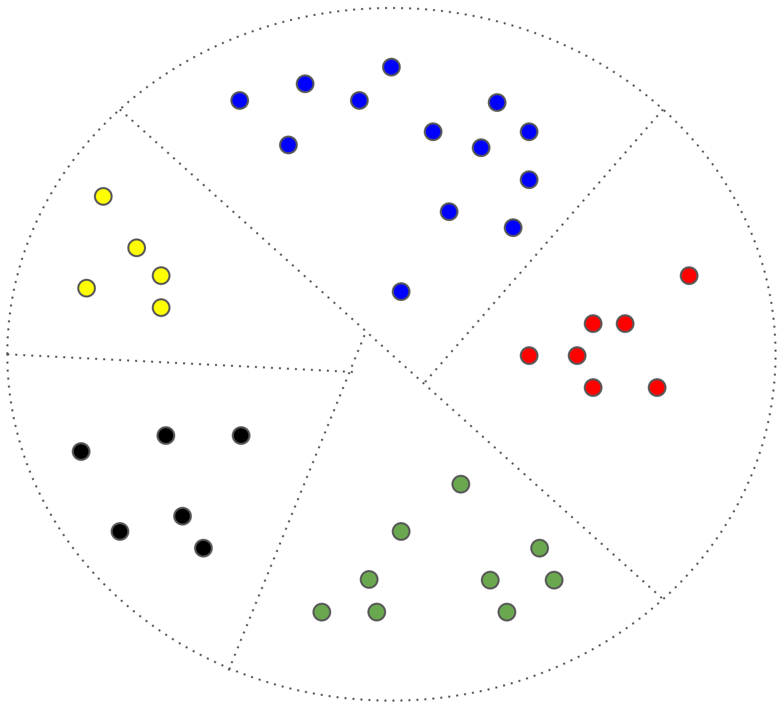
Hierarchical Clustering: Constrained Agglomerative

2. Cluster the documents in these clusters using agglomerative clustering



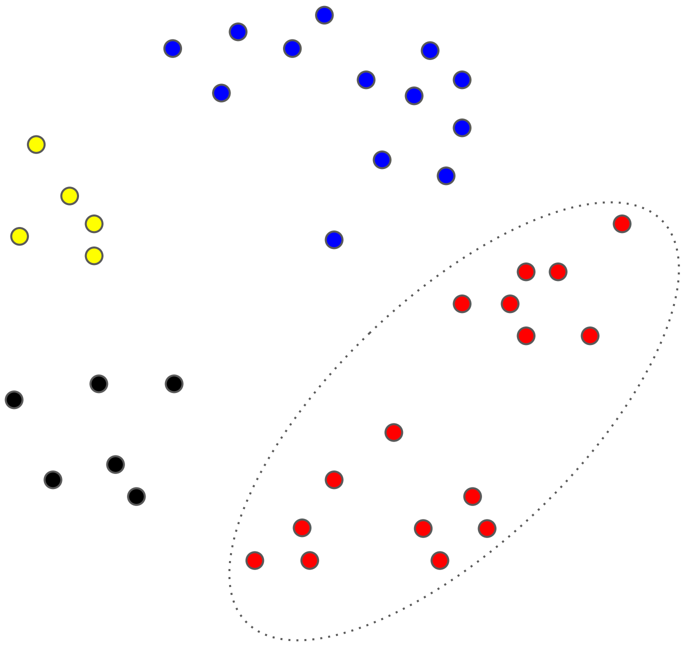
Hierarchical Clustering: Constrained Agglomerative

3. Cluster the k clusters using agglomerative clustering



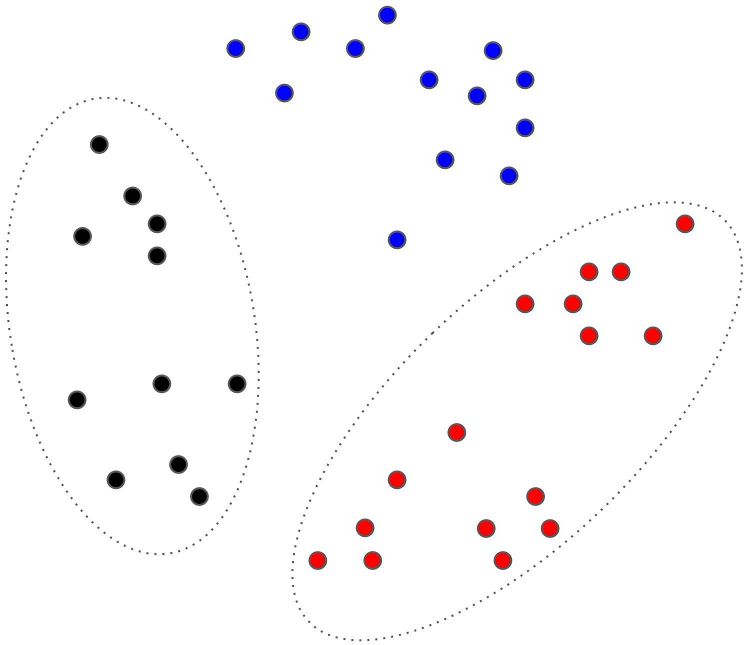
Hierarchical Clustering: Constrained Agglomerative

3. Cluster the k clusters using agglomerative clustering



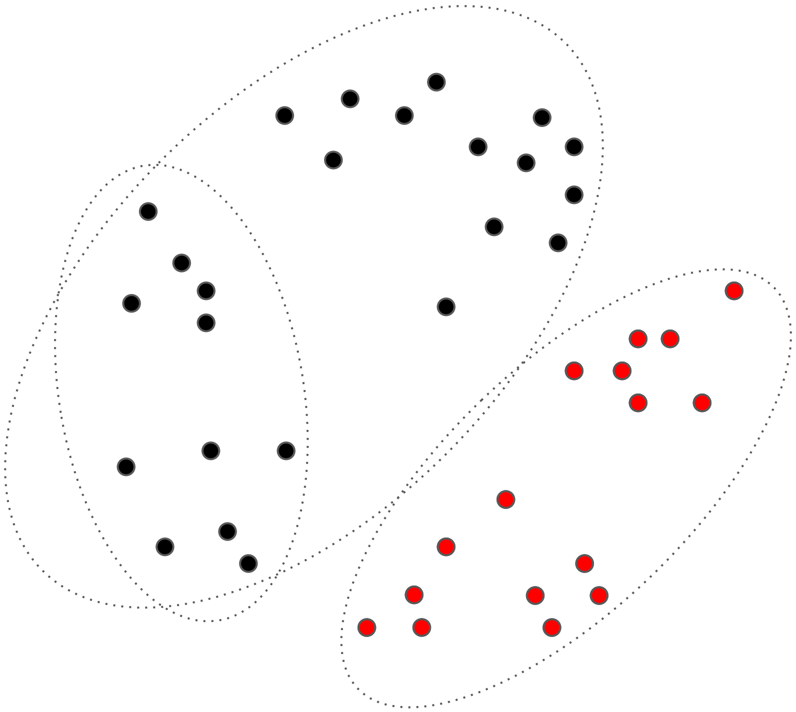
Hierarchical Clustering: Constrained Agglomerative

3. Cluster the k clusters using agglomerative clustering



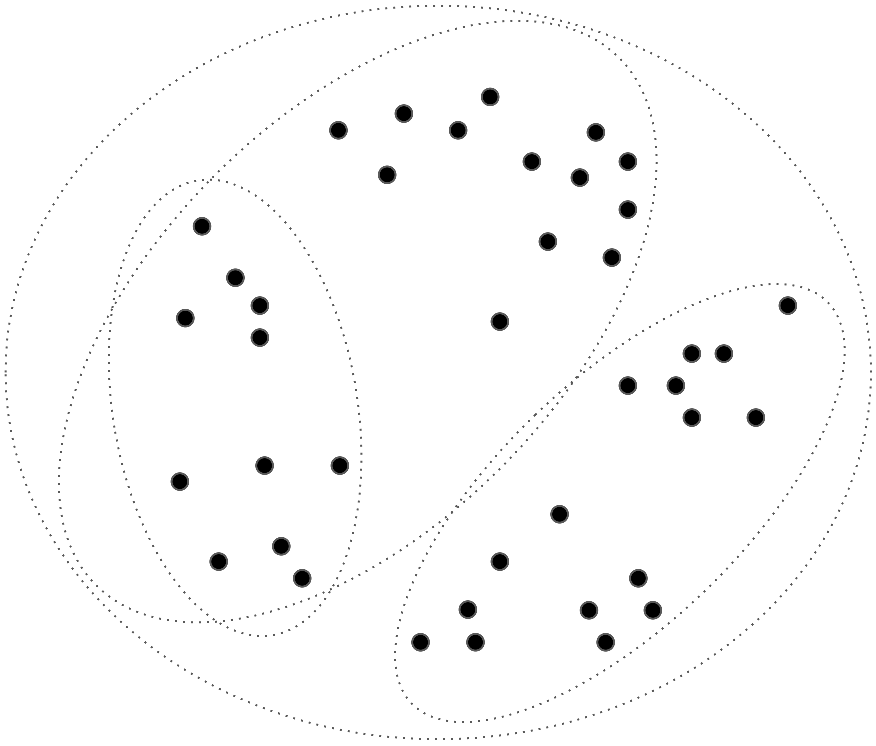
Hierarchical Clustering: Constrained Agglomerative

3. Cluster the k clusters using agglomerative clustering



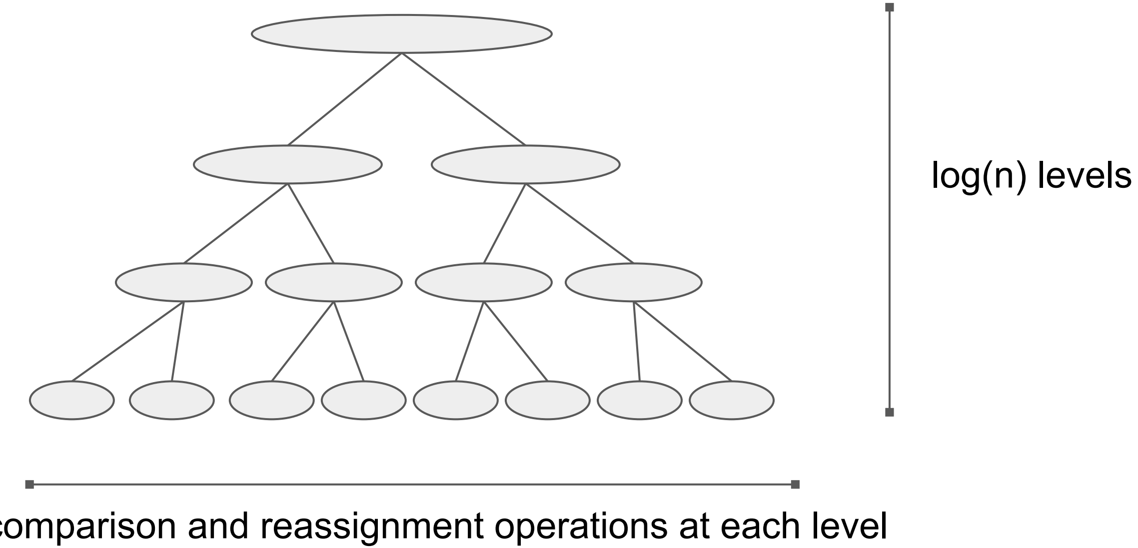
Hierarchical Clustering: Constrained Agglomerative

3. Cluster the k clusters using agglomerative clustering



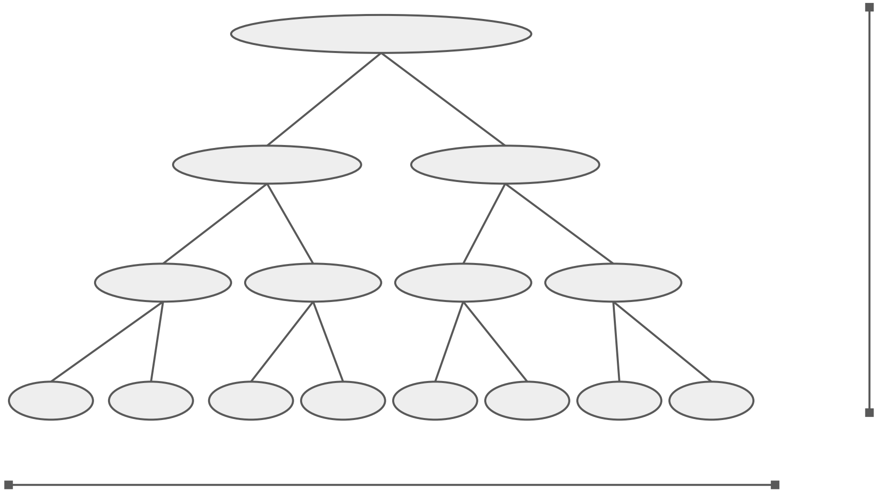
Computational Complexity

- Partitional clustering of data into k clusters:
  < O(n log(n)) (the cost of an entire partitional clustering)



Computational Complexity

- Partitional clustering of data into k clusters:
  < O(n log(n)) (the cost of an entire partitional clustering)



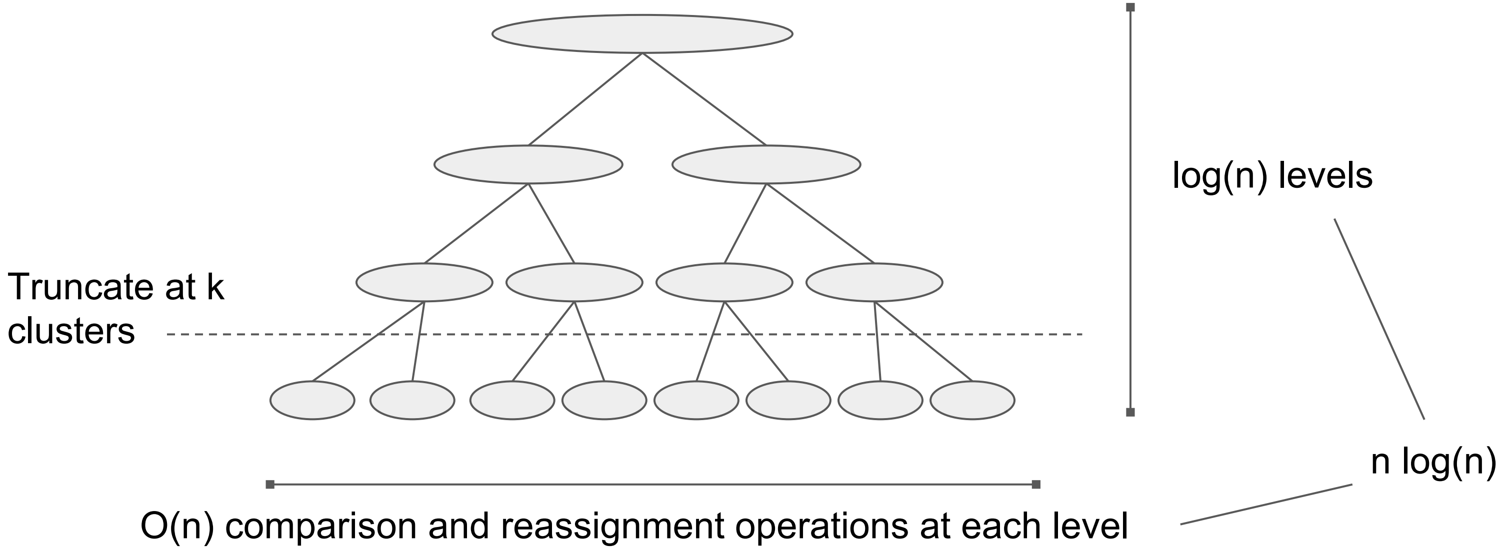
log(n) levels

n log(n)

O(n) comparison and reassignment operations at each level

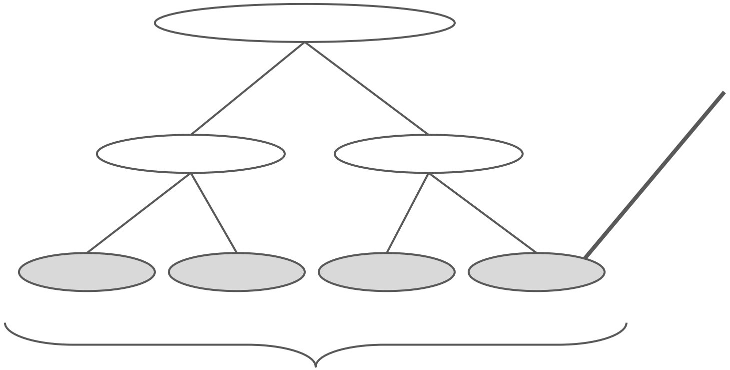
Computational Complexity

- Partitional clustering of data into k clusters:
  < O(n log(n)) (the cost of an entire partitional clustering)



Computational Complexity

- Agglomerative clustering of docs in the k clusters:
  $O(k\ (n/k)^2 \log(n/k))$

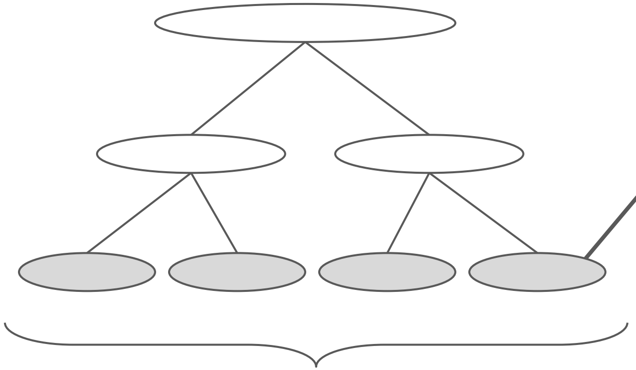


Cost to cluster one cluster:
$O(size^2 \log(size))$
$\approx (n/k)^2 \log(n/k)$

k clusters, size $\approx$ n/k

Computational Complexity

- Agglomerative clustering of docs in the k clusters:
  $O(k\ (n/k)^2 \log(n/k))$



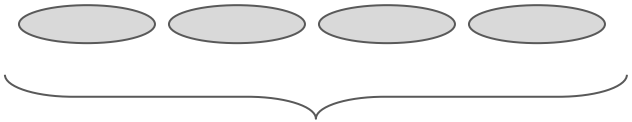
Cost to cluster one cluster:
$O(\text{size}^2 \log(\text{size}))$
$\approx (n/k)^2 \log(n/k)$

k clusters, size $\approx n/k$

$k\ (n/k)^2 \log(n/k)$

Computational Complexity

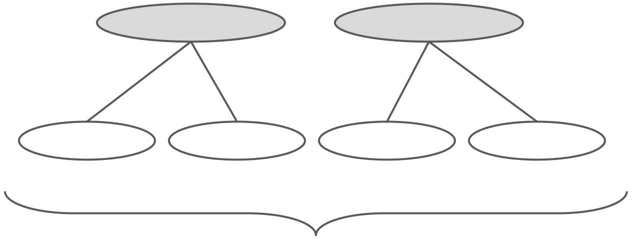
- Agglomerative clustering of docs in the k clusters:
  $O(k^2 \log(k))$



k clusters
cost to cluster agglomeratively = $O(k^2 \log(k))$

Computational Complexity

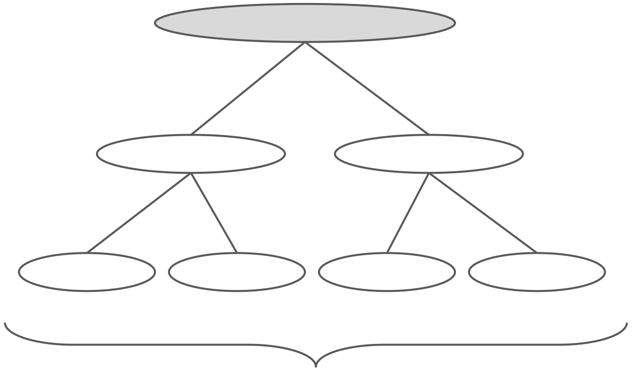
- Agglomerative clustering of docs in the k clusters:
  $O(k^2 \log(k))$



k clusters
cost to cluster agglomeratively = $O(k^2 \log(k))$

Computational Complexity

- Agglomerative clustering of docs in the k clusters:
  $O(k^2 \log(k))$



k clusters
cost to cluster agglomeratively = $O(k^2 \log(k))$

Computational Complexity

- Putting it all together:
  $O(n \log(n)) + O(k\ (n/k)^2 \log(n/k)) + O(k^2 \log(k))$

Initial partitional clustering

Agglomerative clustering **within** initial clusters

Agglomerative clustering **between** initial clusters

Computational Complexity

- Putting it all together:
  $O(n \log(n)) + O(k\,(n/k)^2 \log(n/k)) + O(k^2 \log(k))$

Dominant term for reasonable choices of k

Computational Complexity

- Putting it all together:

  $O(k\ (n/k)^2 \log(n/k))$
- If we let $k \approx \sqrt{n}$, this reduces to:

  $O(n^{3/2} \log(n))$

Computational Complexity

- Putting it all together:
  $$O(k\,(n/k)^2 \log(n/k))$$
- If we let $k \approx \sqrt{n}$, this reduces to:
  $$O(n^{3/2} \log(n))$$
- Or in general, if $k \approx n^{\alpha}$ with $0<\alpha<1$, complexity is:
  $$O(n^{\alpha+2(1-\alpha)} \log(n))$$
- Better than Agglomerative: $O(n^2 \log(n))$
  Worse than Partitional: $O(n \log(n))$

- But a slightly better performer than either on average

Evaluation: Experimental Design

12 document collections were analyzed with each of the hierarchical methods

	Partitional	Agglomerative	Constrained Agglomerative
Criterion Functions	Internal-1 Internal-2 Exernal Hybrid (Internal-1) Hybrid (Internal-2) Graph-Based	Internal-1 Internal-2 Exernal Hybrid (Internal-1) Hybrid (Internal-2) Graph-Based Single Link (slink) Complete Link (clink) Group Average (UPGMA)	Internal-1 Internal-2 Exernal Hybrid (Internal-1) Hybrid (Internal-2) Graph-Based
Number of initial clusters			10 20 n/40 n/20

Document Collections (12)

Data	Source	# of Docs.	# of terms	# of classes
fbis	FBIS (TREC)	2463	12674	17
hitech	San Jose Mercury (TREC)	2301	13170	6
reviews	San Jose Mercury (TREC)	4069	23220	5
la1	LA Times (TREC)	3204	21604	6
la2	LA Times (TREC)	3075	21604	6
tr31	TREC	927	10128	7
tr41	TREC	878	7454	10
re0	Reuters-21578	1504	2886	13
re1	Reuters-21578	1657	3758	25
k1a	WebACE	2340	13879	20
k1b	WebACE	2340	13879	6
wap	WebACE	1560	8460	20

Vector Space Model

Model design:

- TF-IDF term weighting

$$\left\{tf_1 \log\left(\frac{n}{df_1}\right), tf_2 \log\left(\frac{n}{df_2}\right), \dots, tf_m \log\left(\frac{n}{df_m}\right)\right\}$$

- Normalized by document length

$$\left\|d_{tfidf}\right\| = 1$$

- Cosine similarity

$$\cos(d_i, d_j) = \frac{d_i^t d_j}{\|d_i\|\left\|d_j\right\|}$$

FScore Metric

FScore for a class L_r and a cluster S_i: how well does the cluster align with the class?

$$F(L_r, S_i) = \frac{2 \times recall(L_r, S_i) \times precision(L_r, S_i)}{recall(L_r, S_i) + precision(L_r, S_i)}$$

FScore Metric

FScore for a class L_r and a cluster S_i: how well does the cluster align with the class?

$$F(L_r, S_i) = \frac{2 \times recall(L_r, S_i) \times precision(L_r, S_i)}{recall(L_r, S_i) + precision(L_r, S_i)}$$

Define F for class L_r as maximum over all clusters S_i in the clustering tree:

$$F(L_r) = \max_{S_i \in T} F(L_r, S_i)$$

FScore Metric

FScore for a class L_r and a cluster S_i: how well does the cluster align with the class?

$$F(L_r, S_i) = \frac{2 \times recall(L_r, S_i) \times precision(L_r, S_i)}{recall(L_r, S_i) + precision(L_r, S_i)}$$

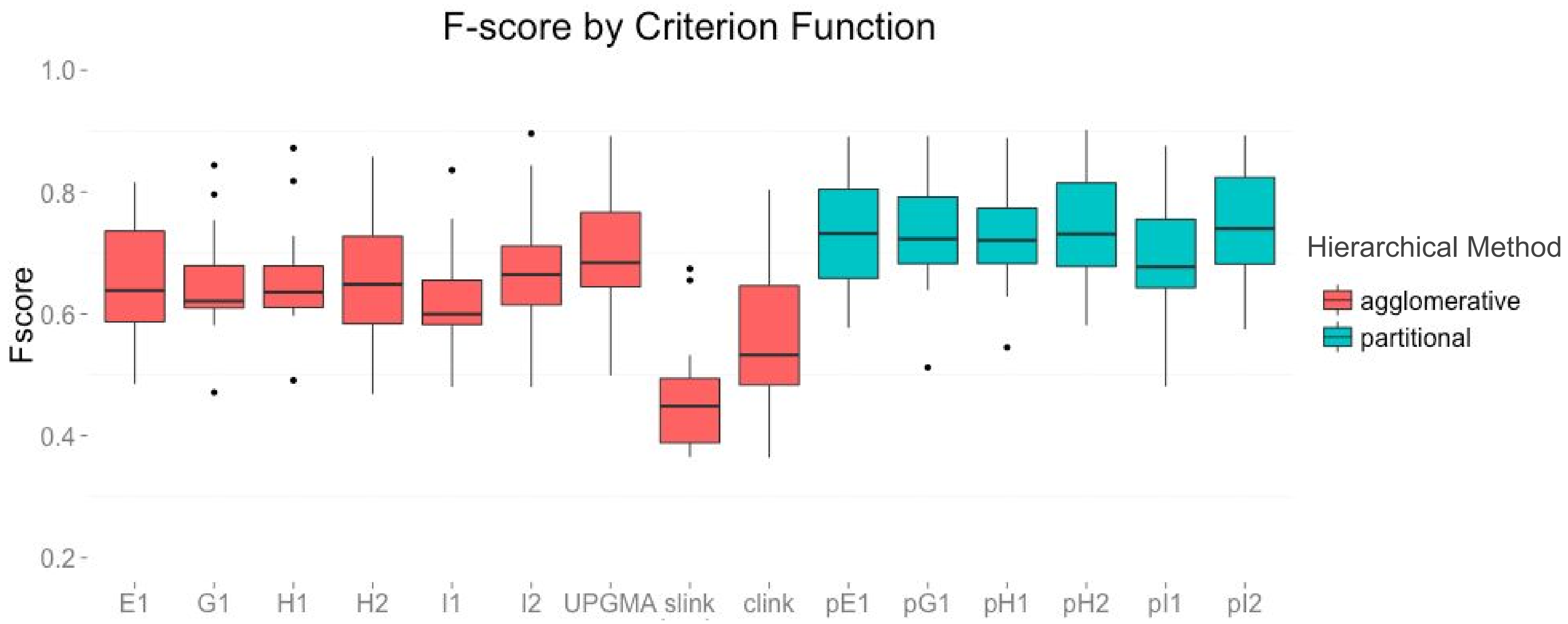
Define F for class L_r as maximum over all clusters S_i in the clustering tree:

$$F(L_r) = \max_{S_i \in T} F(L_r, S_i)$$

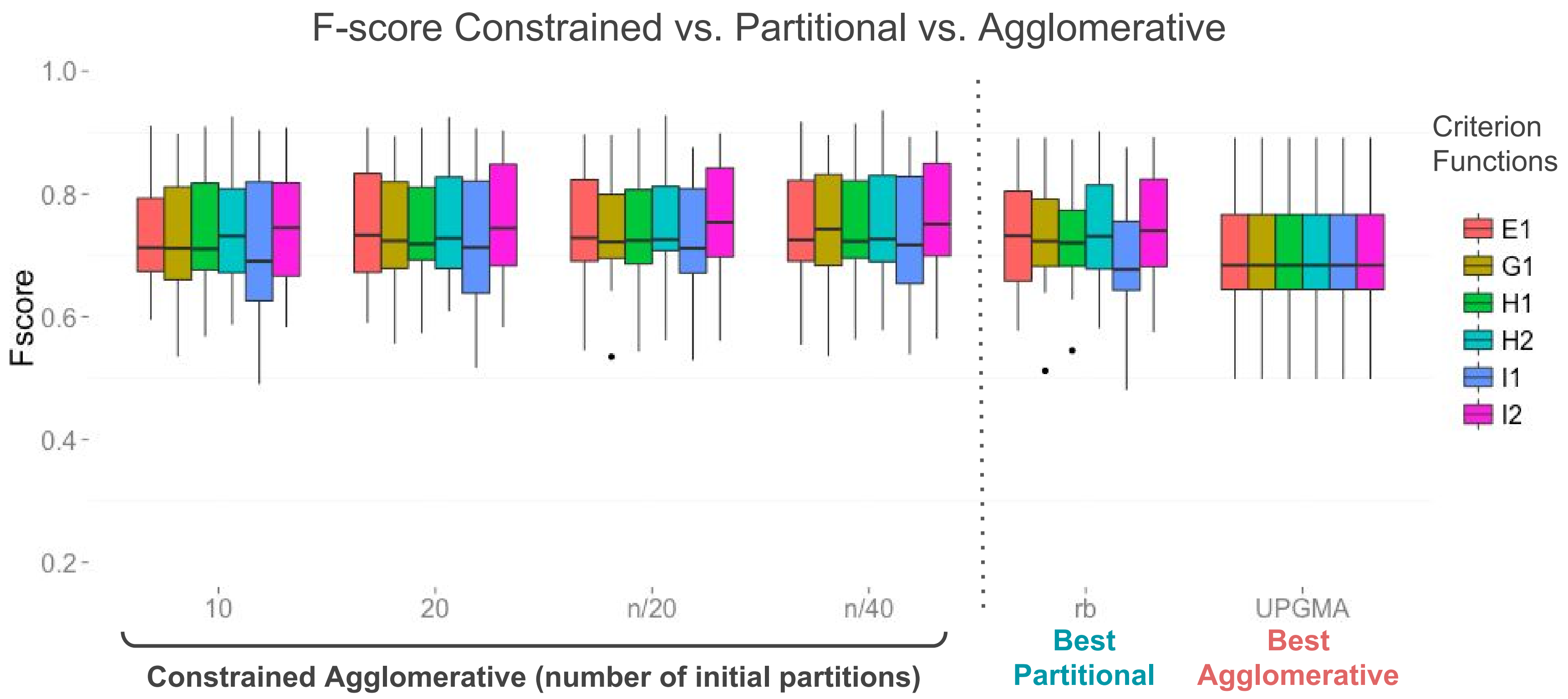
FScore for entire clustering: $F(L_r)$ summed across classes,weighted by class size

$$FScore = \sum_{r=1}^{c} \frac{n_r}{n} F(L_r)$$

Results: Agglomerative vs. Partitional



Results: Constrained Agglomerative



Conclusion

Zhao and Karypis did a thorough comparison of hierarchical clustering methods on large document collections

Partitional algorithms consistently outperformed agglomerative methods

Constrained agglomerative methods outperformed the partitional methods in many cases

Thank You

Clustering Method Comparisons

Partitional	**Agglomerative**	**Constrained Agglomerative**
Complexity: - $O(n \log n)$	Complexity: - $O(n^2 \log n)$ With caching in a binary heap - $O(n^3)$ If the similarity function is not cacheable	Complexity - $O(k((n/k)^2 \log (n/k)) + k^2 \log k)$ - $=O(n^{3/2} \log n)$ when number of partitional clusters ≈ $\sqrt{n}$
Limited studies show agglomerative methods outperform k-means with small datasets	Initial merging may contain errors, which can multiply during agglomeration	Partitional cluster constraint prevents initial merging errors (merging across cluster boundaries)
Suited for large datasets due to low computational requirements	Easy to group documents in small, cohesive clusters	
Common belief (2001) that k-means methods are inferior than agglomerative		