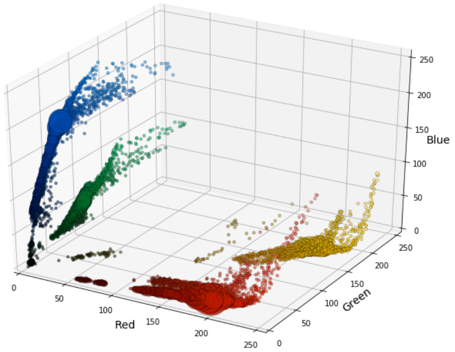
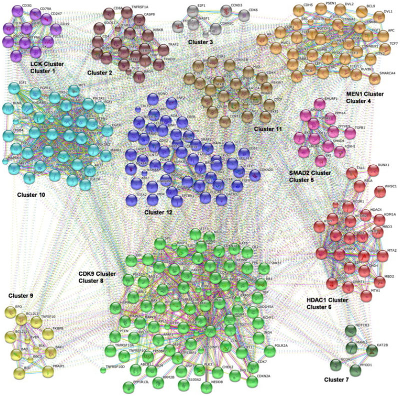
Clustering - Intro

Frederik Mallmann-Trenn
6CCS3AIN



- Goal: Points in the same cluster are ‘close’ and points of different clusters are ’far’

Application: Biological Networks



- Protein-Protein Interaction Networks (PPINs). See e.g., Rual et al Nature ’05 (+2500 citations)

Application: Recommendation Systems

- Similar users will buy similar items
- High dimensional data, how to find similar users? Clustering!

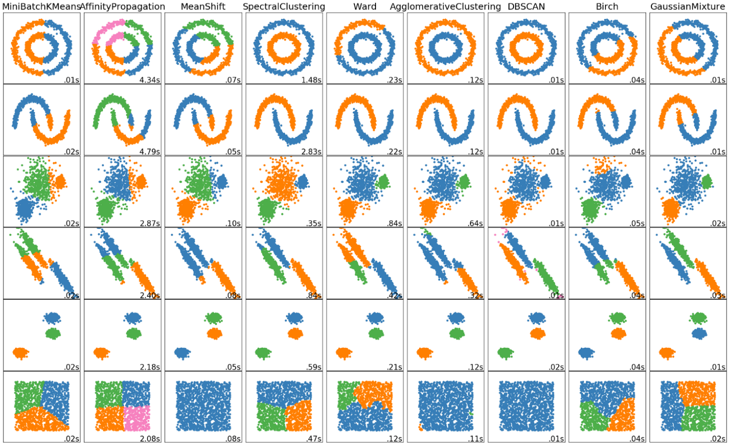
Other Applications

- Unsupervised learning (e.g., PCA)
- encoder-decoder neural network
- outlier detection applications such as detection of credit card fraud
- ....

Clustering Algorithm (incomplete)

- k-Means
- k-Median
- Agglomerative Clustering
  - Single-Linkage
  - Average-Linkage
  - Complete-Linkage
- Divisive Clustering
  - Sparsest-Cut

Clustering Algorithm Comparison



Source: https://towardsdatascience.com/

k-Means and k-Median

Frederik Mallmann-Trenn
6CCS3AIN

k-Means

- Say we have n points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.
- We want to partition them into k sets $S_1, S_2, \ldots, S_k$ such that the cost of the partition, $c(S_1, S_2, \ldots, S_k)$, is minimised:

$$c(S_1, S_2, \ldots, S_k) = \sum_{i=1}^{n} \left(\min_{j \in [k]} d(\mathbf{x}_i, \boldsymbol{\mu}_j) \right)^2,$$

where $\boldsymbol{\mu}_i$ is the mid-point of each cluster, i.e.,

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_j$$

k-Means

- Say we have n points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.
- We want to partition them into k sets $S_1, S_2, \ldots, S_k$ such that the cost of the partition, $c(S_1, S_2, \ldots, S_k)$, is minimised:

$$c(S_1, S_2, \ldots, S_k) = \sum_{i=1}^{n} \left(\min_{j \in [k]} d(\mathbf{x}_i, \boldsymbol{\mu}_j) \right)^2,$$

where $\boldsymbol{\mu}_i$ is the mid-point of each cluster, i.e.,

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_j$$

- Note that $\mathbf{x}_i$ and $\boldsymbol{\mu}_i$ are vectors for $i \in [n]$.

k-Means

- Say we have n points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.
- We want to partition them into k sets $S_1, S_2, \ldots, S_k$ such that the cost of the partition, $c(S_1, S_2, \ldots, S_k)$, is minimised:

$$c(S_1, S_2, \ldots, S_k) = \sum_{i=1}^{n} \left(\min_{j \in [k]} d(\mathbf{x}_i, \boldsymbol{\mu}_j) \right)^2,$$

where $\boldsymbol{\mu}_i$ is the mid-point of each cluster, i.e.,

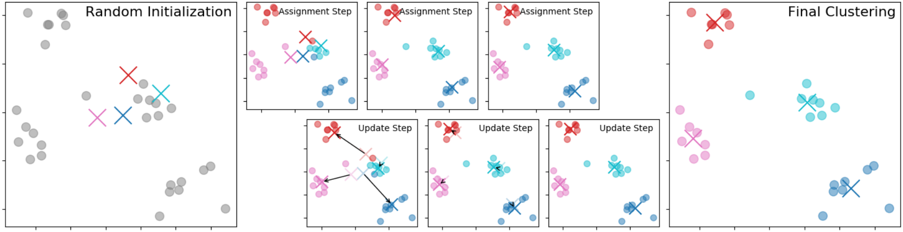
$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_j$$

- Note that $\mathbf{x}_i$ and $\boldsymbol{\mu}_i$ are vectors for $i \in [n]$.
- E.g., if $S_1 = \{\mathbf{x}_1, \mathbf{x}_2\}$ with $\mathbf{x}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\mathbf{x}_2 = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, then $\boldsymbol{\mu}_1 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$

k-Means

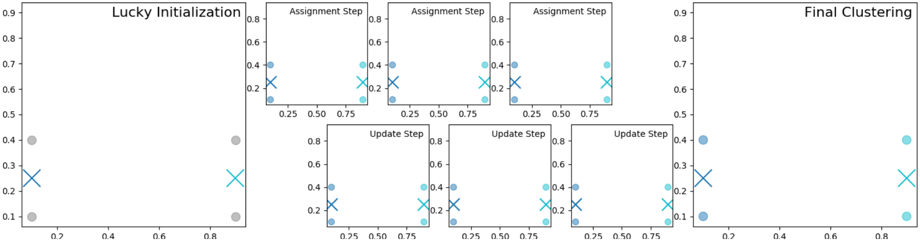
1. Select k cluster centres arbitrarily.
2. Repeat until convergence:
   2.1 Assignment Step:
      2.1.1 Assign each point to the cluster with the nearest mean
      2.1.2 $S_i = \{\mathbf{x}_j \mid d(\mathbf{x}_j, \boldsymbol{\mu}_i) \leqslant d(\mathbf{x}_i, \boldsymbol{\mu}_\ell)$ for all $\ell \in [k]\}$, where each point is assigned to exactly one cluster S_i.
   2.2 Update Step:
      2.2.1 Recalculate the mean point of the cluster
      2.2.2 $\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_j$

k-Means



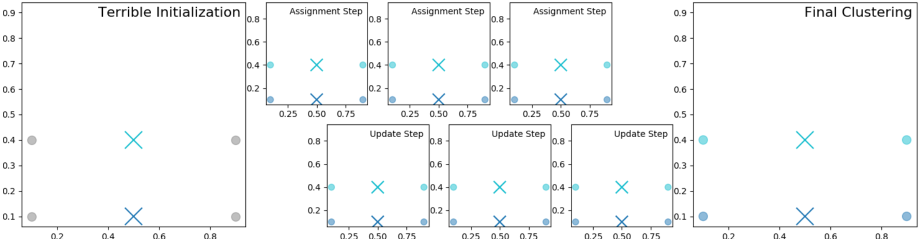
Random Initialization
Assignment Step
Assignment Step
Assignment Step
Update Step
Update Step
Update Step
Final Clustering

k-Means: Example with Optimal Instance



- Consider this example with four points.
- The optimal cluster is shown.

k-Means



- We can see that if we start with sub-optimal clusters, and we never change them!
- This can be made arbitrarily bad (by increasing the width of the rectangle).

k-Means++

- The way this can be solved is by using k-Means++
- It can be shown that the approximation factor is at most $O(\log k)$.

k-Means++

1. Set the first centre to be one of the input points chosen uniformly at random, i.e., $\boldsymbol{\mu}_1 = uniform(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$
2. For cluster $i = 2$ to k:
   2.1 For each point $\mathbf{x}_j$ compute the distance to the nearest centre, i.e., calculate $d_j = \min_\ell d(\mathbf{x}_j, \boldsymbol{\mu}_\ell)$
   2.2 Open a new centre at a point using the weighted probability distribution that is proportional to d_j^2. That is,

$$\Pr(\text{new centre in } \mathbf{x}_j) = \frac{d_j^2}{\sum_\ell d_\ell^2}$$

3. Continue with k-Means

k-Median

- Say we have n points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.
- We want to partition them into k sets $S_1, S_2, \ldots, S_k$ such that the cost of the partition, $c(S_1, S_2, \ldots, S_k)$, is minimised:

$$c(S_1, S_2, \ldots, S_k) = \sum_{i=1}^{n} \left(\min_{j \in [k]} d(\mathbf{x}_i, \boldsymbol{\mu}_j) \right),$$

where $\boldsymbol{\mu}_i$ is the mid-point of each cluster, i.e.,

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_j$$

k-Median

- Say we have n points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.
- We want to partition them into k sets $S_1, S_2, \ldots, S_k$ such that the cost of the partition, $c(S_1, S_2, \ldots, S_k)$, is minimised:

$$c(S_1, S_2, \ldots, S_k) = \sum_{i=1}^{n} \left(\min_{j \in [k]} d(\mathbf{x}_i, \boldsymbol{\mu}_j) \right),$$

where $\boldsymbol{\mu}_i$ is the mid-point of each cluster, i.e.,

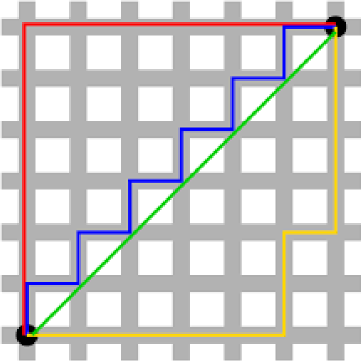
$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{j \in S_i} \mathbf{x}_j$$

- Recall that k-Means uses

$$\sum_{i=1}^{n} \left(\min_{j \in [k]} d(\mathbf{x}_i, \boldsymbol{\mu}_j) \right)^2$$

-

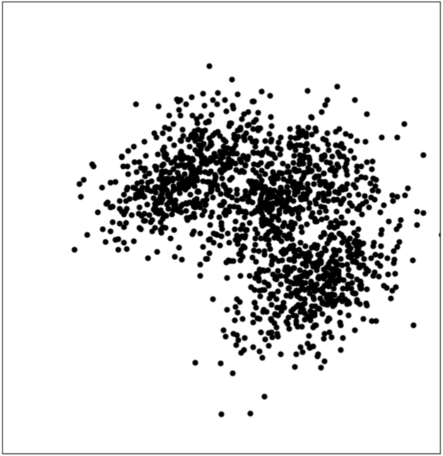
k-Median

- K-Means minimises the Euclidean/geometric distance
- K-Medians minimises the Manhattan distance
- 
- Source: Wikipedia

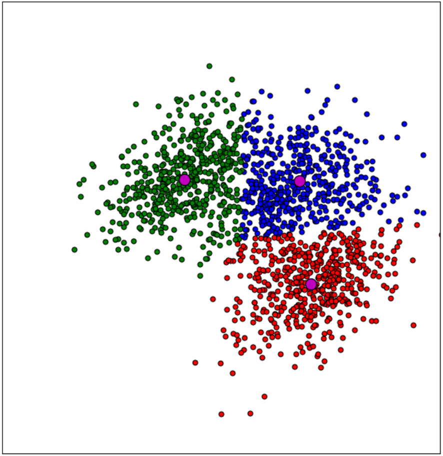
Hierarchical Clustering

Frederik Mallmann-Trenn
6CCS3AIN

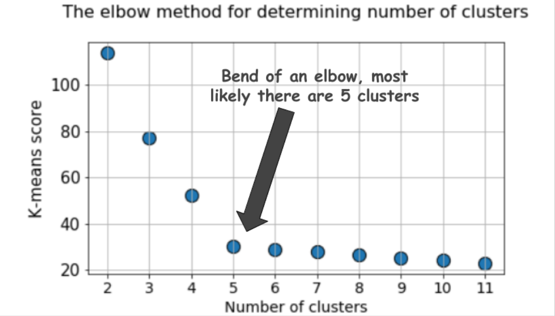
How should we choose k?



How should we choose k?



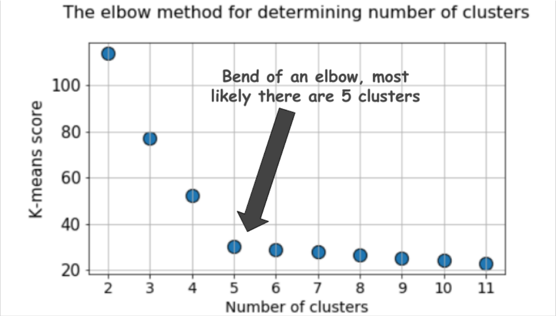
Elbow method



Source: https://towardsdatascience.com/

- The more clusters the better the lower the cost/score
- Of course we could have one cluster per datapoint, but that does not really help
- One way of strike a good tradeoff is to use the elbow method

Elbow method



Source: https://towardsdatascience.com/

- The more clusters the better the lower the cost/score
- Of course we could have one cluster per datapoint, but that does not really help
- One way of strike a good tradeoff is to use the elbow method
- A natural definition: Pick the $k \in \{2, 3, \ldots, n-1\}$ that maximises $cost_{k-1}/cost_k$, where $cost_j$ is the cost of the clustering with j clusters
- $k = 1$ and $k = n$ are ruled out as they result in trivial clusters

Flat Clustering

- So far we tried to assign the points to k clusters.
- We haven't assumed any structure among the clusters

Clustering

The problem with flat clustering is that it's flat

Example: Cluster the following news headlines in 3 categories

Clustering

The problem with flat clustering is that it's flat

Example: Cluster the following news headlines in 3 categories

- Deepmind's AlphaBingo wins world championship
- Black holes swallow stars whole according to new study
- Someone didn't dope
- Researcher finally figured out the rules of cricket

Clustering

The problem with flat clustering is that it's flat

Example: Cluster the following news headlines in 3 categories

CS	• Deepmind's AlphaBingo wins world championship
Physics	• Black holes swallow stars whole according to new study
Sports	• Someone didn't dope
Sports	• Researcher finally figured out the rules of cricket

Clustering

The problem with flat clustering is that it's flat

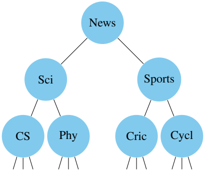
Example: Cluster the following news headlines in 3 categories

Science	• Deepmind's AlphaBingo wins world championship
Science	• Black holes swallow stars whole according to new study
Cycling	• Someone didn't dope
Cricket	• Researcher finally figured out the rules of cricket

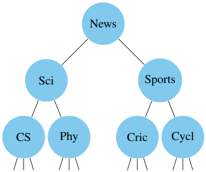
Structure is lost ...

Hierarchical Clustering

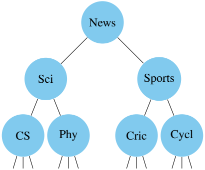
- Recursive partitioning of data at increasingly finer granularity represented as a tree
- The leaves of the hierarchical cluster tree represent data.



Hierarchical Clustering

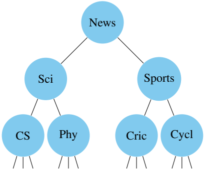


Hierarchical Clustering



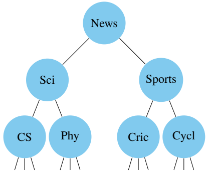
- Google's AlphaBingo wins world champinship
- Someone finally figured out why neural nets work
- Black holes swallow stars whole according to new study
- Someone didn't dope
- Researcher finally figured out the rules of cricket

Hierarchical Clustering



Science	• Google's AlphaBingo wins world champinship
Science	• Someone finally figured out why neural nets work
Science	• Black holes swallow stars whole according to new study
Sports	• Someone didn't dope
Sports	• Researcher finally figured out the rules of cricket

Hierarchical Clustering



CS	•Google's AlphaBingo wins world champinship
CS	•Someone finally figured out why neural nets work
Physics	•Black holes swallow stars whole according to new study
Cycling	•Someone didn't dope
Cricket	•Researcher finally figured out the rules of cricket

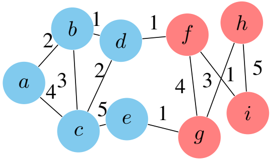
Hierarchical Clustering Algorithms

Frederik Mallmann-Trenn
6CCS3AIN

How are hierarchical clusters obtained in practice?

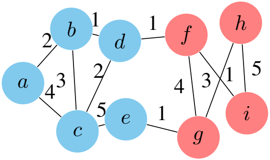
- Agglomerative clustering (bottom up)
  - Initially place each data point in its own clusters
  - Repeatedly merge most similar clusters
- Divisive clustering (top down)
  - Split using bisection k-means (or sparsest cut)
  - Recurse on each part

Agglomerative Clustering



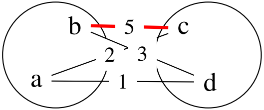
- An edge can mean similarity or dissimilarity
- In this lecture we'll only consider similarities.
- A high similarity value means that the corresponding two nodes are very similar and should be in the same cluster

Agglomerative Clustering



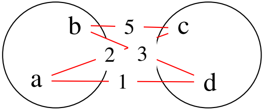
- Each node starts in a separate cluster
- The similarity between two clusters $C_1 = \{a, b\}$, $C_2 = \{c, d\}$ is

Single Linkage



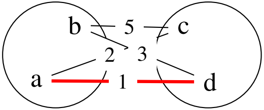
Similarity: 5

Average Linkage



Similiarity: 2.75

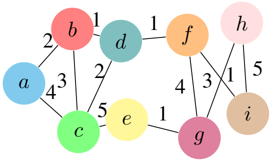
Complete Linkage



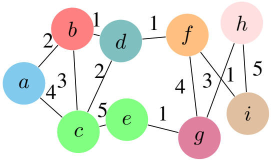
Similiarity: 1

Single-Linkage

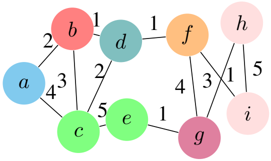
Round 0:



Round 1:

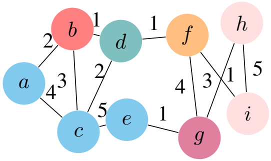


Round 2:

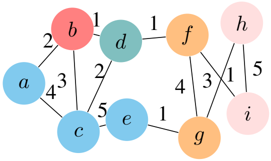


Single-Linkage

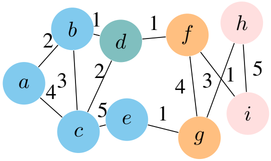
Round 3:



Round 4:

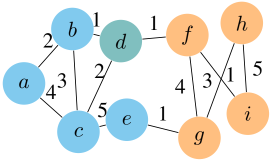


Round 5:

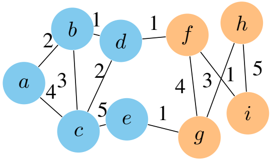


Single-Linkage

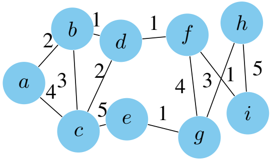
Round 6:



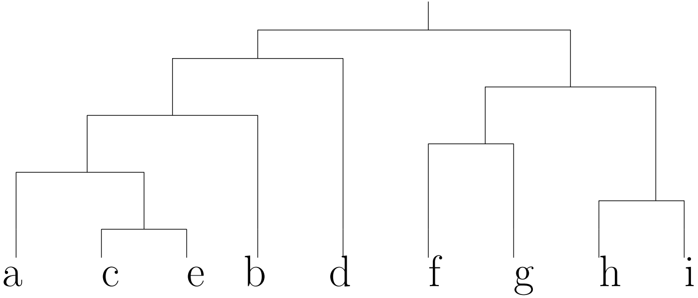
Round 7:



Round 8:

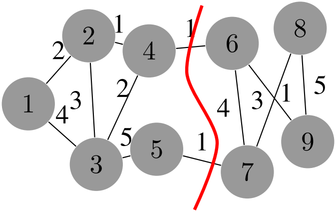


Hierarchical Clustering: Dendogram



- The obtained merges can be represented in a dendogram

Hierarchical Clustering: Divisive Heuristics



- Find a partition of the input similarity graph (or set of points)
  - Split using bisection k-means
  - Split using sparsest cut
- Recurse on each part
- Builds cluster-tree top-down

Sparsest cut

- Given a graph with nodes V and edges in $E \subset V \times V$
- The sparsity of a cut $\phi(S)$, is given by

$$\phi(S) = \frac{E(S, \bar{S})}{\min(|S|, |\bar{S}|)},$$

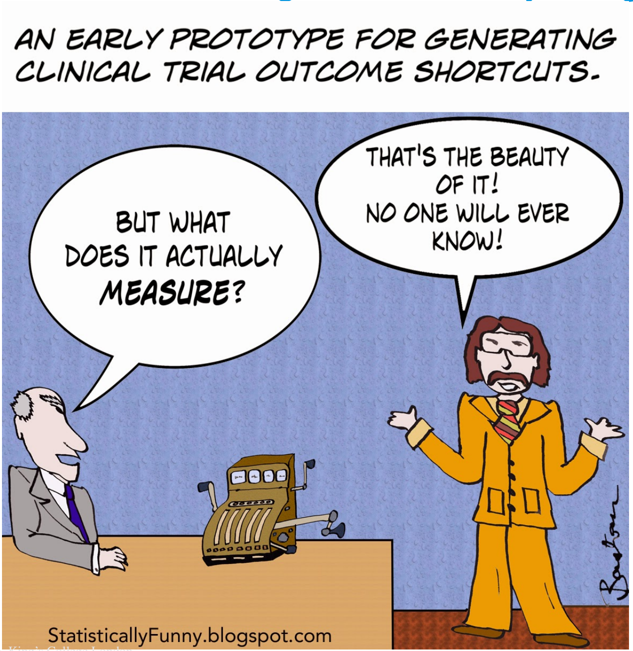
  where $E(S, \bar{S})$ is the sum of weights of edges crossing the cut. Here $\bar{S} = V \backslash S$.
- The sparsest cut of a graph is given by the S^* that minimises $\phi(S^*)$, i.e.,

$$\phi(S^*) = \min_S \phi(S)$$

Hierarchical Clustering Objective

Frederik Mallmann-Trenn
6CCS3AIN

What are the hierarchical algorithms actually doing?



What quantity are these algorithms optimizing?

- For flat clustering, algorithms designed to optimize some objective function
  - Remember:
    for **flat clustering** the goal was to find k points $\mu_1, \ldots, \mu_k$ that minimize, e.g.
    1. k-median objective $\sum_{i=1}^{N} \left(\min_{j \in [k]} d(\mathbf{x}_i, \mu_j) \right)$
    2. k-means objective $\sum_{i=1}^{N} \left(\min_{j \in [k]} d(\mathbf{x}_i, \mu_j) \right)^2$
- For hierarchical clustering, algorithms have been studied procedurally
  - Thus, comparisons between hierarchical clustering algorithms are only qualitative!

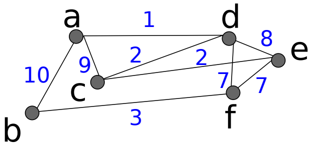
What quantity are these algorithms optimizing?

- For flat clustering, algorithms designed to optimize some objective function
  - Remember:
    for **flat clustering** the goal was to find k points $\mu_1, \ldots, \mu_k$ that minimize, e.g.
    1. k-median objective $$\sum_{i=1}^{N} \left(\min_{j \in [k]} d(\mathbf{x}_i, \mu_j) \right)$$
    2. k-means objective $$\sum_{i=1}^{N} \left(\min_{j \in [k]} d(\mathbf{x}_i, \mu_j) \right)^2$$
- For hierarchical clustering, algorithms have been studied procedurally
  - Thus, comparisons between hierarchical clustering algorithms are only qualitative!
- [Dasgupta '16]
  "The lack of an objective function has prevented a theoretical understanding"
- Dasgupta introduced an objective function to model the hierarchical clustering problem

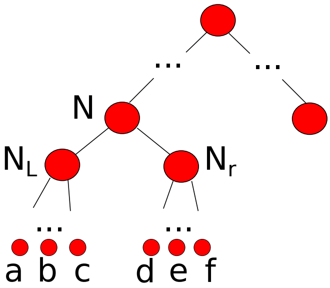
Dasgupta's Cost Function

Input: a weighted similarity graph G

- Edge weights represent similarities



Output: T a tree with leaves labelled by nodes of G



Cost of the output: Sum of the costs of the nodes of T

Cost of a node N of the tree:

$L = \{u \mid u$ is leaf of subtree rooted at $N_L\}$

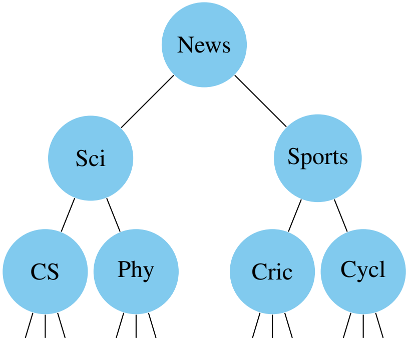
$R = \{v \mid v$ is leaf of subtree rooted at $N_R\}$

$$\boxed{\text{cost}(N) = (|L| + |R|) \cdot \sum_{\substack{u \in L \\ v \in R}} \text{similarity}(u, v)}$$

The total cost is the sum of costs of all subtrees.

Intuition: Better to cut a high similarity edge at a lower level

Results

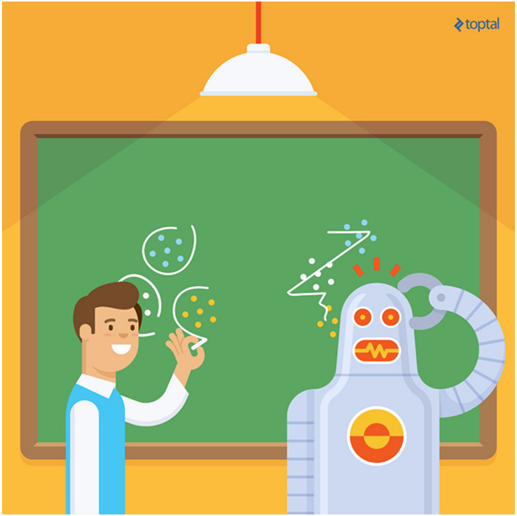


Hierarchical Clustering: Objective Functions and Algorithms
Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, Claire Mathieu
JACM 2019

Result 1: Is Dasgupta the only reasonable function?

- We characterize the set of ‘good’ objective functions based on axioms.
  - Disconnected components must be separated first
  - Symmetry
  - Only the ‘true’ hierarchical clustering should minimize the cost function (if there is one).
- In some sense Dasgupta is the most natural

Result 2: Algorithms



Hope: Recursive Sparsest Cut

- Dissimilarity graph:
  - We show Avg. Linkage has a $3/2$-approximation factor
  - We also show that other practical algorithms have a $\Omega(n^{1/4})$-approximation factor

Hope: Recursive Sparsest Cut

- Dissimilarity graph:
  - We show Avg. Linkage has a $3/2$-approximation factor
  - We also show that other practical algorithms have a $\Omega(n^{1/4})$-approximation factor
- Similarity graph:

Algorithm: Recursive Sparsest Cut

Input: Weighted graph $G = (V, E, w)$

$\{A, V\setminus A\} \leftarrow$ cut with sparsity $\leqslant \phi \cdot \min_{S\subseteq V} \frac{w(S, V\setminus S)}{|S| \cdot |V\setminus S|}$

Recurse on subgraphs $G[A], G[V\setminus A]$ to obtain trees $T_A, T_{V\setminus A}$

Output: Return tree whose root has subtrees $T_A, T_{V\setminus A}$

Hope: Recursive Sparsest Cut

- Dissimilarity graph:
  - We show Avg. Linkage has a $3/2$-approximation factor
  - We also show that other practical algorithms have a $\Omega(n^{1/4})$-approximation factor
- Similarity graph:

Algorithm: Recursive Sparsest Cut

Input: Weighted graph $G = (V, E, w)$

$\{A, V\backslash A\} \leftarrow$ cut with sparsity $\leqslant \phi \cdot \min_{S \subseteq V} \frac{w(S, V\backslash S)}{|S| \cdot |V\backslash S|}$

Recurse on subgraphs $G[A], G[V\backslash A]$ to obtain trees $T_A, T_{V\backslash A}$

Output: Return tree whose root has subtrees $T_A, T_{V\backslash A}$

  - We show $O(\phi)$-approximation
  - Improves the $O(\log n \cdot \phi)$-approximation [Dasgupta '16]
  - Current best known value for ϕ is $O(\sqrt{\log n})$ [ARV '09]

- For worst case inputs, Recursive Sparsest Cut gives $O(\phi)$-approximation
- Assuming the "Small Set Expansion Hypothesis", no polytime $O(1)$-approx.

Real-world graphs are often not worst-case