Learning from Networks

Graph Clustering

Fabio Vandin

December 4th, 2024

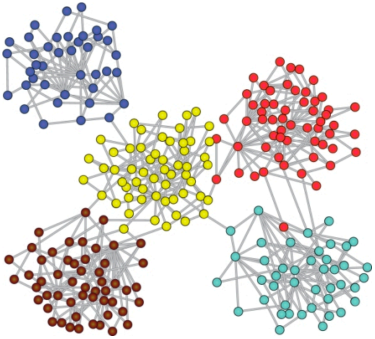
Graph Clustering: Definition

Given: graph $G = (V, E)$

Goal: partition V into clusters so that *similar vertices* are in the same cluster and *different vertices* are in different clusters.

Graph Clustering: Definition (continue)

Intuition: the similarity between vertices are represented by the edges



Given: *connected* graph $G = (V, E)$
Goal: partition V so that there are many edges *within* each cluster and few edges *between* clusters.

Many different formalizations based on this intuition.

Note: sometimes clusters in a graph are called *communities*

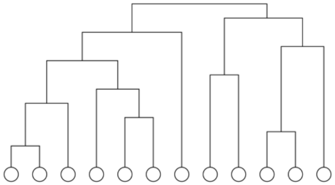
Graph Clustering: Approaches

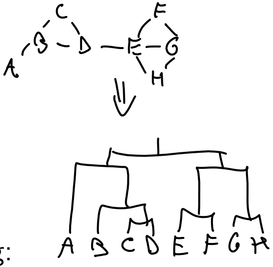
We will see different types of approaches for clustering:

- hierarchical clustering
- cost-based clustering
- (spectral clustering)

Hierarchical Clustering

The output is a **dendrogram**, representing the clustering structure of the whole graph G.





Two general approaches to hierarchical clustering:

- *agglomerative approach*: start with each node in a cluster, iteratively join clusters $\Rightarrow$ Ravasz algorithm
- *divisive approach*: start with all nodes in a cluster, iteratively split clusters $\Rightarrow$ Girvan-Newman algorithm

Ravasz Algorithm

Algorithm `AgglomerativeClustering(`G`)`
Input: connected graph $G = (V, E)$
Output: dendrogram whose leaves are the elements of V

```
                       1 assign each node $u$ to its own cluster $C_u$;
                       2 for all pairs $u, v \in V, u \neq v$: compute their similarity $sim(u, v)$
$\Theta(n) \Leftarrow$ 3 repeat until all nodes are in a single cluster:
$\leq n$ calcoli di        4 find the pair of clusters $C_1, C_2$ with highest similarity
$sim(C_1, C_2)$              $sim(C_1, C_2)$ (ties broken arbitrarily)
                           5 merge clusters $C_1, C_2$ in a single cluster $C'$
                           6 compute similarity between $C'$ and all other clusters
                       7 return the corresponding dendrogram
```

Different variants depending on the definition of $sim(u, v)$ and the definition of $sim(C_1, C_2)$.

Complexity? In general: $\Theta\left(|V|^2\right)$ computations of $sim(u, v)$ and of $sim(C_1, C_2)$

Ravasz Algorithm (continue)

Common choice for $sim(u,v)$:

$$sim(u,v) = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)| + A_{uv}}{\min\{deg(u), deg(v)\} + 1 - A_{uv}}$$

where A is the adjacency matrix of G

Ravasz Algorithm (continue)

Common choices for $sim(C_1, C_2)$ define different types of *linkage clustering*:

- *single linkage clustering*: $sim(C_1, C_2) = \min_{u \in C_1, v \in C_2} sim(u, v)$
- *average linkage clustering*:
  $$sim(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{u \in C_1, v \in C_2} sim(u, v)$$
- *complete linkage clustering*: $sim(C_1, C_2) = \max_{u \in C_1, v \in C_2} sim(u, v)$

Example

Girvan-Newman Algorithm

Based on the idea of iteratively removing the most *central* edge in the graph $G = (V, E)$.

Various definitions of *centrality* for edges, but the most common one is *link betweenness*.

Link betweenness

Let $\sigma_{s,t}$ be the number of shortest paths from node s to node t.
Let $\sigma_{s,t}(e)$ be the number of shortest paths from node s to node t that pass through *edge* e.

Definition

Given a connected graph $G = (V, E)$ and an edge $e \in E$, the link betweenness $b(e, G)$ of e in G:

$$b(e, G) = \sum_{s,t \in V : s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$$

Complexity of computing $b(e, G)$ for all edges $e \in E$?
$\Theta(|V| \cdot |E|)$

Example

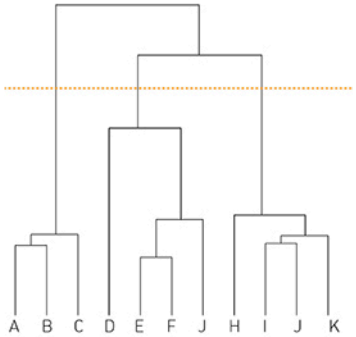
Girvan-Newman Algorithm (continue)

```
Algorithm GNClustering($G$)
Input: connected graph $G = (V, E)$
Output: dendrogram whose leaves are the elements of $V$
  1 assign all nodes $u$ to a single cluster $C$;
  2 repeat until all nodes are in different clusters:
      3 for each cluster $C$:
          4 for each edge $e \in C$: compute $b(e, C)$
      5 let $e_{\max}$ the edge of maximum betweenness, and let $C(e)$ its
        cluster;
      6 remove $e$ from $C(e)$;
  7 report the corresponding dendrogram
```

Complexity? In general: $\Theta\left(|E|^2|V|\right)$.

Hierarchical Clustering: Getting a Clustering

The output of hierarchical clustering is a dendrogram, not a clustering. How do we obtain a clustering?

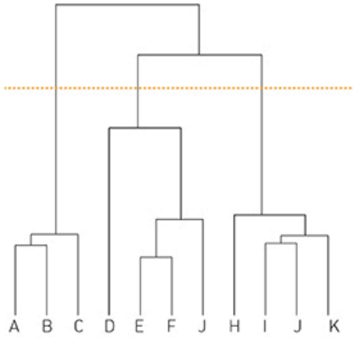


By cutting the dendrogram at a given level.

How do we select where to cut?

Hierarchical Clustering: Getting a Clustering (continue)

How do we select where to cut?



- if we know the number *k* of clusters we want: pick a level resulting in *k* clusters
- if we do not know *k*: define a score for clusterings, and pick the clustering from the dendrogram of maximum score

Cost-based Clustering

Common approach in clustering (not only for graphs):

- define a cost function over possible partitions of the objects
- find the partition (=clustering) of minimal cost

Modularity

Idea:a cluster should contain more edges than expected in a random graph.

Definition

Given a graph $G = (V, E)$ with $|V| = n$, $|E| = m$ the modularity $M(S)$ of a subset $S \subseteq V$ of the vertices of G is

$$M(S) = \frac{1}{2m} \sum_{u,v \in S} \left(A_{uv} - \frac{deg(u)deg(v)}{2m} \right)$$

Intuition: measures the difference between the number of edges within each cluster with the *expected* number of edges under the Chung-Lu model for random graphs.

Modularity (continue)

The modularity of a clustering of G is the sum of the modularity of each cluster.

Definition

Given a clustering $\mathcal{C} = C_1, C_2, \ldots$ of graph $G = (V, E)$ with $|V| = n$, $|E| = m$, the modularity $M(\mathcal{C})$ of $\mathcal{C}$ is:

$$\begin{aligned} M(\mathcal{C}) &= \sum_{C \in \mathcal{C}} M(C) \\ &= \frac{1}{2m} \sum_{C \in \mathcal{C}} \sum_{u,v \in C} \left(A_{uv} - \frac{deg(u)deg(v)}{2m} \right) \end{aligned}$$

Modularity (continue)

Proposition

Given a clustering $\mathcal{C} = C_1, C_2, \ldots$ of graph $G = (V, E)$ with $|V| = n$, $|E| = m$, the modularity $M(\mathcal{C})$ of $\mathcal{C}$ is equal to

$$M(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left(\frac{|E(C)|}{m} - \left(\frac{\sum_{u \in C} deg(u)}{2m} \right)^2 \right)$$

where $E(C)$ are the edges between nodes in cluster C:
$E(C) = \{(u, v) \in E : u \in C, v \in C\}$

Proof

Example

Modularity-Based Clustering

Input: graph $G = (V, E)$
Goal: find the clustering $\mathcal{C} = C_1, C_2, \ldots$ that maximizes the modularity

$$M(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left(\frac{|E(C)|}{m} - \left(\frac{\sum_{u \in C} deg(u)}{2m} \right)^2 \right)$$

Equivalent formulation: since the cost of clustering $\mathcal{C}$ is $-M(\mathcal{C})$, the following formulation is equivalent:

Input: graph $G = (V, E)$
Goal: find the clustering $\mathcal{C} = C_1, C_2, \ldots$ of minimum cost $-M(\mathcal{C})$.

Modularity-Based Clustering: Complexity

Informal: finding a clustering of maximum modularity is hard!

Problem (Modularity Clustering Problem)

Given a graph G and a value K, is there a clustering $\mathcal{C}$ of G such that $M(\mathcal{C}) \geq K$?

Proposition

The Modularity Clustering Problem is NP-complete.

So? (Greedy) agglomerative algorithm

Modularity-Based Clustering: Greedy Agglomerative Approach

```
Algorithm GreedyModularityClustering($G$)
Input: connected graph $G = (V, E)$
Output: clustering of the elements of $V$
  1 $\mathcal{C}_1 \leftarrow$ clustering where each node $u$ is assigned to its own cluster $C_u$; $i \leftarrow 1$;
  2 repeat until all nodes are in a single cluster:
      3 for each pair of clusters $C_1, C_2$ such that there exists one edge between $C_1$ and $C_2$: compute $\delta(\mathcal{C}_i, C_1, C_2) = M(\mathcal{C}_i - C_1 - C_2 + (C_1 \cup C_2)) - M(\mathcal{C}_i)$;
      4 find $C', C''$ that maximize $\delta(\mathcal{C}_i, C', C'')$
      5 $\mathcal{C}_{i+1} \leftarrow \mathcal{C}_i - C' - C'' + (C' \cup C'')$; $i \leftarrow i + 1$;
  6 return the clustering $\mathcal{C}^*$, across iterations, of maximum modularity: $\mathcal{C}^* = \arg\max_{\mathcal{C}_i, i=1,2,\ldots} M(\mathcal{C}_i)$
```

Complexity? In general: $O(|E| \cdot |V|)$ computations of $\delta(\mathcal{C}_i, C_1, C_2)$

Modularity-Based Clustering: Efficient Computation

Proposition

Let $E(C_1, C_2)$ be the edges between cluster C_1 and cluster C_2: $E(C_1, C_2) = \{(u, v) \in E : u \in C_1, v \in C_2\}$. Then

$$\delta(\mathcal{C}_i, C_1, C_2) = \frac{|E(C_1, C_2)|}{m} - \frac{\left(\sum_{u \in C_1} deg(u)\right)\left(\sum_{v \in C_2} deg(v)\right)}{2m^2}$$

Modularity-Based Clustering: Efficient Computation (continue)

Proposition

In every iteration of the repeat-until loop, the values $|E(C_1, C_2)|$ for all $C_1, C_2 \in \mathcal{C}$ and $\sum_{u \in C} deg(u)$ for all $C \in \mathcal{C}$ can be efficiently updated in total time $O(|E|)$.

Example