Hypergraph Modularities and Clustering

Bogumił Kamiński
Valérie Poulin
Paweł Prałat
Przemysław Szufel
François Théberge*
theberge@ieee.org

HyTAC, November 2019

Outline

Graph modularity

Let $G = (V, E)$.
We can write the modularity of a partition **A** of V as:

$$
\begin{aligned}
q_G(\mathbf{A}) &= \sum_{A_i \in \mathbf{A}} \left(\frac{e_G(A_i)}{|E|} - \frac{(vol(A_i))^2}{4|E|^2} \right) \\
&= \frac{1}{|E|} \sum_{A_i \in \mathbf{A}} \left(e_G(A_i) - \mathop{\mathbb{E}}_{G \in \mathcal{G}} (e_G(A_i)) \right)
\end{aligned}
$$

$e_G(A_i) = |\{e \in E : e \subseteq A_i\}|$ is the *edge contribution* and,
$\mathbb{E}_{G \in \mathcal{G}}(e_G(A_i))$ is the *degree tax*.

Chung-Lu Model

Select m edges $e = (u_1, u_2)$ where each u_i is independently sampled from V according to the multinomial distribution where $p(v_i) = deg_G(v_i)/vol(V)$.

Let $\mathcal{CL}_2(G)$, the distribution of graphs obtained. For $G' = (V, E') \sim \mathcal{CL}_2(G)$:

- $\mathbb{E}_{G' \sim \mathcal{CL}_2(G)}(deg_{G'}(v_i)) = deg_G(v_i), 1 \leq i \leq n$.
- we always have $|E'| = |E| = m$,
- there can be multi-edges,
- there can be self-edges,
- complexity is $O(m)$.

Chung-Lu Model

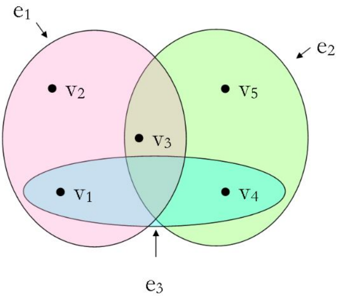
Lemma

The degree tax term in the modularity function for graph G is the expected value of the edge contribution term over the graphs $G' \sim \mathcal{CL}_2(G)$.

Can we generalize this model for hypergraphs?

Hypergraphs

- $H = (V, E)$ where $|V| = n$, $|E| = m$
- $e \in E$: (hyper)-edges where $e \subseteq V$, $|e| \geq 2$
- Edges can have weights
- We consider undirected hypergraphs



Hypergraphs

- Few hypergraph-based algorithms exist in data science
- They are typically slower
- Some have an equivalent graph representation

(q) Can we define a modularity function on hypergraphs?

Chung-Lu Model for Hypergraphs

Consider a hypergraph $H = (V, E)$ with $V = \{v_1, \ldots, v_n\}$.

Hyperedges $e \in E$ are subsets of V of cardinality greater than one where:

$$e = \{(v, m_e(v)) : v \in V\}$$

and $m_e(v) \in \mathbb{N} \cup \{0\}$ is the multiplicity of the vertex v in e

$|e| = \sum_v m_e(v)$ is the *size* of hyperedge e, and

$deg(v) = \sum_{e \in E} m_e(v)$, and

$vol(A) = \sum_{v \in A} deg(v)$ as for graphs.

Chung-Lu Model for Hypergraphs

Let F_d be the family of multisets of size d; that is,

$$F_d := \left\{ \{(v_i, m_i) : 1 \le i \le n\} : \sum_{i=1}^{n} m_i = d \right\}.$$

The hypergraphs in the random model are generated via independent random experiments. For each d such that $|E_d| > 0$, the probability of generating $e \in F_d$ is given by:

$$P_{\mathcal{H}}(e) = |E_d| \cdot \binom{d}{m_1, \ldots, m_n} \prod_{i=1}^{n} \left(\frac{deg(v_i)}{vol(V)} \right)^{m_i}.$$

where $m_i = m_e(v_i)$.

Chung-Lu Model for Hypergraphs

We can show that:

$$\mathbb{E}_{H' \sim \mathcal{H}}[deg_{H'}(v_i)] = \sum_{d \geq 2} \frac{d \cdot |E_d| \cdot deg(v_i)}{vol(V)} = deg(v_i),$$

with $vol(V) = \sum_{d \geq 2} d \cdot |E_d|$.

We use this generalization of the Chung-Lu model as our null model (*degree tax*) to define hypergraph modularity.

Hypergraph Modularities

Let $H = (V, E)$ and $\mathbf{A} = \{A_1, \ldots, A_k\}$, a partition of V. For edges of size greater than 2, several definitions can be used to quantify the *edge contribution* given $\mathbf{A}$, such as:

(a) all vertices of an edge have to belong to one of the parts to contribute; this is a *strict* definition;

(b) the *majority* of vertices of an edge belong to one of the parts;

(c) at least 2 vertices of an edge belong to the same part; this is implicitly used when we replace a hypergraph with its 2-section graph representation.

Strict Hypergraph Modularity

The edge contribution for $A_i \subseteq V$ is:

$$e(A_i) = |\{e \in E;\ e \subseteq A_i\}|.$$

The strict modularity of $\mathbf{A}$ on H is then defined as a natural extension of standard modularity in the following way:

$$q_H(\mathbf{A}) = \frac{1}{|E|} \sum_{A_i \in \mathbf{A}} \left(e(A_i) - \mathbb{E}_{H' \sim \mathcal{H}}[e_{H'}(A_i)] \right).$$

which can be written as:

$$q_H(\mathbf{A}) = \frac{1}{|E|} \left(\sum_{A_i \in \mathbf{A}} e(A_i) - \sum_{d \geq 2} |E_d| \sum_{A_i \in \mathbf{A}} \left(\frac{vol(A_i)}{vol(V)} \right)^d \right)$$

Link to Chung-Lu Model

For each d, sample $|E_d|$ edges $e = (u_1, .., u_d)$ where each u_i is independently sampled from V with $p(v_i) \propto deg(v_i)$.

Let $\mathcal{CL}_2(H)$, the distribution of hypergraphs obtained this way; for $H' = (V, E') \sim \mathcal{CL}_2(H)$:

- $\mathbb{E}_{H' \sim \mathcal{CL}_2(H)}(deg_{H'}(v_i)) = deg_H(v_i), 1 \leq i \leq n$.
- we always have $|E'_d| = |E_d|$,
- there can be multi-edges, and
- there can be repeated vertices within an edge.

Link to Chung-Lu Model

Lemma

The degree tax term in the modularity function for hypergraph $H = (G, V)$ and partition $\mathbf{A} = \{A_1, ..., A_k\}$ of V is the expected value of the edge contribution term over hypergraphs $H' \sim \mathcal{CL}_2(H)$.

Other Hypergraph Modularity

We can adjust the degree tax to many natural definitions of edge contribution, for example the majority definition.

In this case $(vol(A)/vol(V))^d$ (that is equivalent to $\mathbb{P}\left(\mathrm{Bin}(d, vol(A)/vol(V)) = d\right)$ becomes $\mathbb{P}\left(\mathrm{Bin}(d, vol(A)/vol(V)) > d/2\right)$.

The *majority* modularity function of a hypergraph partition is then:

$$\frac{1}{|E|}\left(\sum_{A_i \in \mathbf{A}} e(A_i) - \sum_{d \geq 2} |E_d| \sum_{A_i \in \mathbf{A}} \mathbb{P}\left(\mathrm{Bin}\left(d, \frac{vol(A_i)}{vol(V)}\right) > d/2\right)\right).$$

Other Hypergraph Modularity

Decomposing H into d-uniform hypergraphs H_d, we get the following degree-independent modularity function:

$$q_H^{DI}(\mathbf{A}) = \sum_{d \geq 2} \frac{|E_d|}{|E|} q_{H_d}(\mathbf{A}).$$

This is as before, but replacing the volumes computed over H with volumes computed over H_d for each d where $|E_d| > 0$.

Finally, we can generalize the modularity function to allow for weighted hyperedges.

Hypergraph Clustering

We seek $\mathbf{A} = \{A_1, ..., A_k\} \in \mathcal{P}(V)$, which maximize the **strict** hypergraph modularity $q_H()$.

Set $\mathcal{P}(V)$ of all partitions of V is huge.

Let: $\mathcal{S}(H) = \{H' = (V, E') \mid E' \subseteq E\}$ and define:

$$p : \mathcal{S}(H) \to \mathcal{P}(V)$$

the function that sends a sub-hypergraph of H to the partition its connected components induce on V.

We define an equivalence relation:

$$H_1 \equiv_p H_2 \iff p(H_1) = p(H_2)$$

and the quotient set $\mathcal{S}(H)/_{\equiv_p}$.

Hypergraph Clustering

Define the *canonical representative mapping*

$$f : \mathcal{S}(H)/_{\equiv_p} \to \mathcal{S}(H)$$

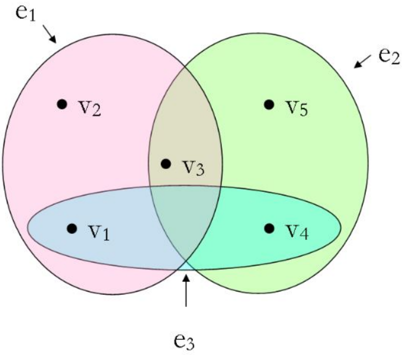
which maps an equivalence class to the largest member of the class: $f([H']) = H^*$.

Let $\mathcal{P}^*(V)$ be the image of p applied to the canonical representatives H^*.

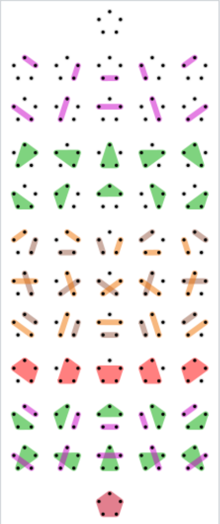
We'll show the optimal solution lies in $\mathcal{P}^*(V)$, a subset of size at most $2^{|E|}$.

Hypergraph Clustering

Consider the toy graph:

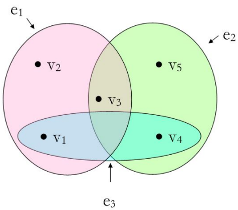


Here, $|\mathcal{P}(V)| = B_5 = 52$.



The 52 partitions of a set with 5 elements

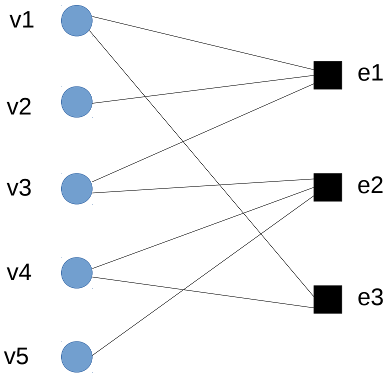
Hypergraph Clustering



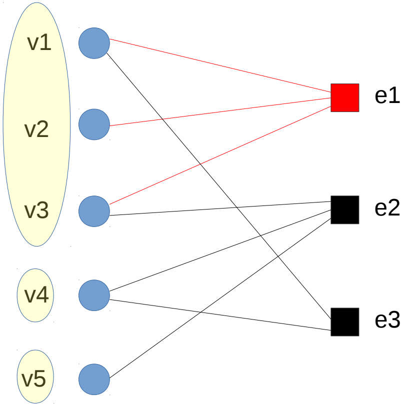
$|\mathcal{P}^*(V)| = 7$
$\leq 2^3 << B_5.$

i	$E_i \subseteq E$	$p(H_i),\ H_i = (V, E_i)$
0	$\emptyset$	$\{\{v_1\},\{v_2\},\{v_3\},\{v_4\},\{v_5\}\}$
1	$\{e_1\}$	$\{\{v_1,v_2,v_3\},\{v_4\},\{v_5\}\}$
2	$\{e_2\}$	$\{\{v_1\},\{v_2\},\{v_3,v_4,v_5\}\}$
3	$\{e_3\}$	$\{\{v_1,v_4\},\{v_2\},\{v_3\},\{v_5\}\}$
4	$\{\mathbf{e_1},\mathbf{e_2}\}$	$\{\{\mathbf{v_1},\mathbf{v_2},\mathbf{v_3},\mathbf{v_4},\mathbf{v_5}\}\}$
5	$\{e_1,e_3\}$	$\{\{v_1,v_2,v_3,v_4\},\{v_5\}\}$
6	$\{e_2,e_3\}$	$\{\{v_1,v_3,v_4,v_5\},\{v_2\}\}$
7	$\{\mathbf{e_1},\mathbf{e_2},\mathbf{e_3}\}$	$\{\{\mathbf{v_1},\mathbf{v_2},\mathbf{v_3},\mathbf{v_4},\mathbf{v_5}\}\}$

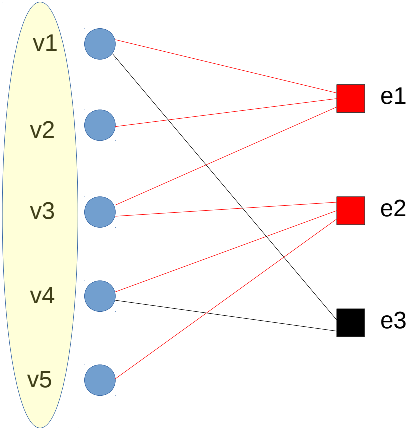
Bipartite graph view



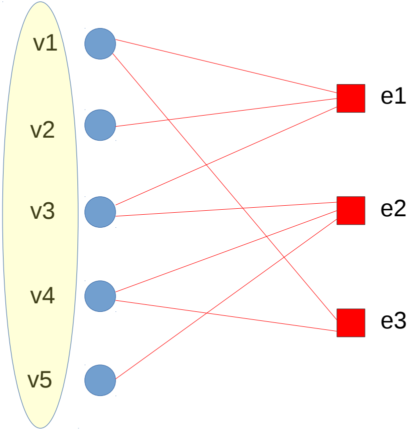
$E_1 = \{e_1\}$



$E_4 = \{e_1, e_2\}$



$E_7 = \{e_1, e_2, e_3\}$



Hypergraph Clustering

Lemma

Let $H = (V, E)$ be a hypergraph and $\mathbf{A} = \{A_1, ..., A_k\}$ a partition of V. If there exists $H' \in \mathcal{S}(H)$ such that $\mathbf{A} = p(H')$, then the edge contribution of $q_H(\mathbf{A})$ is $\frac{|E^*|}{m}$, where E^* is the edge set of the canonical representative H^* of $[H']$.

i.e. the proportion of hyperedges that are subsets of a part.

Hypergraph Clustering

Lemma

Let $H = (V, E)$ be a hypergraph and $\mathbf{A}$ *be any partition of V. If* $\mathbf{B}$ *is a refinement of* $\mathbf{A}$*, then the degree tax of* $\mathbf{B}$ *is smaller than or equal to the degree tax of* $\mathbf{A}$ *with equality if and only if* $\mathbf{A} = \mathbf{B}$.

Hypergraph Clustering

Lemma

Let $H = (V, E)$ be a hypergraph and $\mathbf{A}$ be any partition of V. If $\mathbf{B}$ is a refinement of $\mathbf{A}$, then the degree tax of $\mathbf{B}$ is smaller than or equal to the degree tax of $\mathbf{A}$ with equality if and only if $\mathbf{A} = \mathbf{B}$.

- We prove the following by showing that for any partition, there exists some $H^* \in \mathcal{P}^*(V)$ such that $p(H^*)$ is a refinement of that partition, with the same edge contribution.

Theorem

Let $H = (V, E)$ be a hypergraph. If $\mathbf{A} \in \mathcal{P}(V)$ maximizes the modularity function $q_H(\cdot)$, then $\mathbf{A} \in \mathcal{P}^*(V)$.

Hypergraph Clustering

Previous results give the steps to define heuristic algorithms:

- for $E' \subseteq E$, let $H' = (V, E')$
- find $H^* = [H'] = (V, E^*)$ and compute *edge contribution* part of $q_H()$
- find $\mathbf{A} = p(H^*)$ and compute *degree tax* part of $q_H()$

Simple ways to search for good candidates $E' \subseteq E$:

1. **Greedy random:** shuffle the edges and add edge to E' in turn if $q_H()$ improves; repeat;
2. **CNM-like:** look for best edge to add to E' at each step;

Hypergraph-CNM

Data: hypergraph $H = (V, E)$
Result: $\mathbf{A}_{opt}$, a partition of V with modularity q_{opt}

Initialize $E^* = E$ and $\mathbf{A}_{opt}$ the partition with all $v \in V$ in its own part with q_{opt} the corresponding modularity;
repeat
  if *(random version)* **then**
    $e^* = rand(E^*)$ #randomly select an edge;
    compute the partition A_{e^*} obtained when merging all parts in $\mathbf{A}_{opt}$ touched by e^*, and compute its modularity q_{e^*};
  end
  else
    foreach $e \in E^*$ **do**
      compute the partition A_e obtained when merging all parts in $\mathbf{A}_{opt}$ touched by e, and compute its modularity q_e;
    end
    select edge $e^* \in E^*$ with highest q_{e^*};
  end
  if $q_{e^*} \geq q_{opt}$ **then**
    $A_{opt} = A_{e^*}$ and $q_{opt} = q_{e^*}$;
    update E^*, the set of edges touching two or more parts in $\mathbf{A}_{opt}$;
  end
until *($q_{e^*} < q_{opt}$)* ***or*** *$E^* = \emptyset$* ***or*** *computational time budget exceeded*;
output: $\mathbf{A}_{opt}$ and q_{opt}

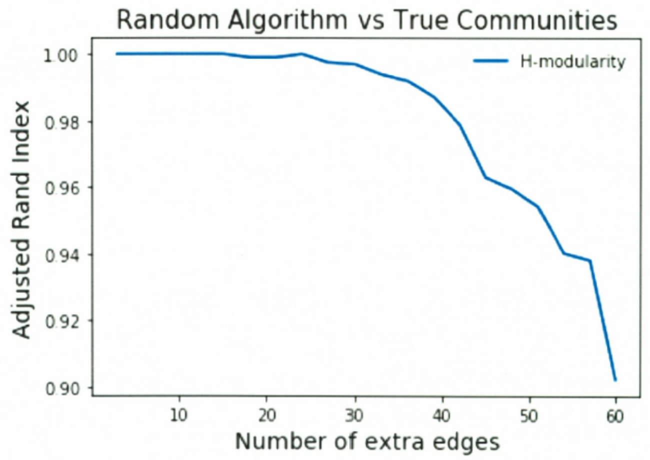
Hypergraph Clustering

Is it working?
Is $q_H()$ a "good" objective function?

Consider the following experiment:

- build hypergraphs with 3 communities of 20 vertices and 50 edges of size $2 \leq d \leq 5$ each;
- add $3 \leq k \leq 60$ random edges of same size(s);
- run random algorithm (with 25 repeats) several times over range of k values;
- for each k, compute mean adjusted RAND index;

Hypergraph Clustering



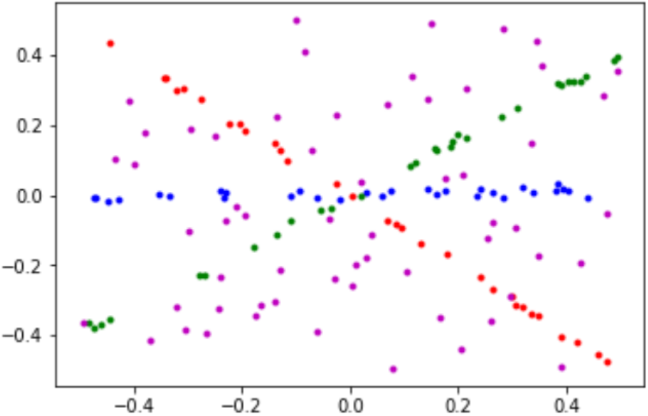
Synthetic Hypergraphs

[REF: M. Leordeanu, C. Sminchisescu, Efficient Hypergraph Clustering]

- Generate noisy points along 3 lines on the plane with different slopes
- add some random points
- select sets of 3 or 4 points (hyperedges)
  - all coming from the same line (“signal”)
  - or not (“noise”)
- Sample hyperedges for which the points are well aligned, and so that the expected proportion of signal vs. noise is 2:1.

We consider 3 different regimes: (i) mostly 3-edges, (ii) mostly 4-edges and (iii) balanced between 3 and 4-edges.

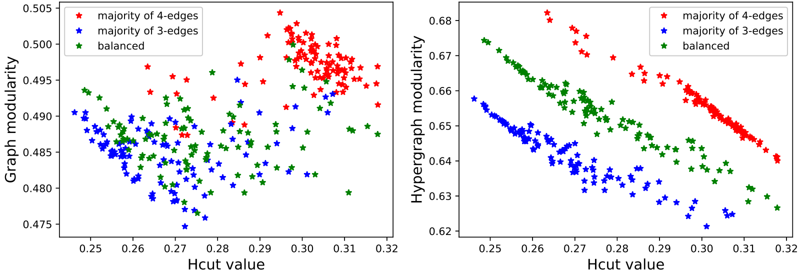
Synthetic Hypergraphs



Synthetic Hypergraphs

Cluster vertices via Louvain on (weighted) 2-section graph.

Modularity vs Hcut. We observe a higher correlation with Hcut (number of splitted hyperedges) with the H-modularity.



DBLP Hypergraph

Small co-author hypergraph with 1637 nodes and 865 hyperedges of sizes 2 to 7.

We compare Louvain (over 2-section) and hypergraph-CNM (with strict modularity).

Partitioning the DBLP dataset.

algorithm	$q_H()$	$q_G()$	Hcut	#parts
Louvain	0.8613	0.8805	0.1181	40
CNM	0.8671	0.8456	0.0945	92

DBLP Hypergraph

Algorithms based on $q_H()$ will tend to cut less of the larger edges, as compared to the Louvain algorithm, at expense of cutting more size-2 edges.

Proportion of edges of size 2, 3 or 4 cut by the algorithms.

Algorithm	2-edges	3-edges	4-edges
Louvain	0.0382	0.1815	0.3158
CNM	0.0590	0.1277	0.1842

Graph Laplacian

Let $G = (V, E)$ with W be the matrix of edge weights $w(u, v)$ for all $(u, v) \in E$.

Let D be the diagonal matrix of node degrees:
$d(v) = \sum_{u \sim v} w(u, v)$

We first review the (unsupervised) Ncut problem

Ref: Ulrike von Luxburg, *A Tutorial on Spectral Clustering*, Technical Report No. TR-149, Max Planck Inst., Germany, 2006.

Graph Laplacian

For a partition $V = S \cup S^c$:

$$Ncut(S, S^c) = \frac{Vol \partial S}{Vol S} + \frac{Vol \partial S}{Vol S^c}$$

where:

$\partial S = \{e \in E; |e \cap S| = |e \cap S^c| = 1\}$
$Vol(S) = \sum_{v \in S} d(v)$
$Vol(\partial S) = \sum_{(u,v) \in \partial S} w(u, v)$

This can be viewed as a random walk with transition probabilities $P = D^{-1} W$:

$$Ncut(S, S^c) = P(S|S^c) + P(S^c|S)$$

.

Graph Laplacian

The problem can be solved by relaxing over real values

$$f^* = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}}\, \Omega(f); \quad f \perp D^{1/2} \cdot 1, \ ||f||^2 = vol(V).$$

where $\Omega(f) = \langle f^t, \Delta f \rangle$ and

$$\Delta = I - D^{-1/2} W D^{-1/2}$$

is the (normalized) graph Laplacian.

Graph Laplacian

The Laplacian also appears in semi-supervised problems with some initial (seed) labels y on the vertices.

If nodes are close (large $w(u,v)$), we should have labels $f(u) \approx f(v)$ to keep $w(u,v)(f(u) - f(v))^2$ small.

Define a semi-supervised problem as a trade-off between "smoothness" with respect to the graph topology, and consistency with respect to y, such as:

$$f^* = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}} \left(\Omega(f) + \mu ||f - y||^2 \right)$$

Ref: D. Zhou and B. Schölkopf, *A Regularization Framework for Learning from Graph Data*, 2004.

Graph Laplacian

Let $\Omega(f) = \langle f^t, \Delta f \rangle$, then we can show that:

$$\Omega(f) = \frac{1}{2} \sum_{(u,v)\in E} w(u,v) \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2$$

and if $y \neq 0$, and $\mu > 0$, there exists a closed form solution:

$$f^* = \mu(\Delta + \mu I)^{-1} y = (1 - \alpha)(I - \alpha S)^{-1} y$$

where:

$$\alpha = (1 + \mu)^{-1}$$

$\Delta = I - D^{-1/2} W D^{-1/2}$, the normalized graph Laplacian

$S = I - \Delta$, the smoothness matrix

Hypergraph Laplacian

Ref: D. Zhou, J. Huang and B. Schölkopf, *Learning with Hypergraphs: Clustering, Classification and Embedding*, 2007.

For (undirected) hypergraphs, define:

E: set of subsets $e \subset V$
$w(e)$: hyperedge weight
$d(v) = \sum_{e; v \in e} w(e)$
$\delta(e) = |e| \geq 2$, the "hyperedge degree"
$H : |V| \times |E|$ s.t. $h(v, e) = 1$ iff $v \in e$
$W = diag(w(e))$,
$D_v = diag(d(v))$,
$D_e = diag(\delta(e))$.

Hypergraph Laplacian

The Ncut problem can be generalized to hypergraphs.

For a partition $V = S \cup S^c$, let:

$$\partial S = \{e \in E; e \cap S \neq \emptyset,\ e \cap S^c \neq \emptyset\}$$

$$VolS = \sum_{v \in S} d(v)$$

$$Vol\partial S = \sum_{e \in \partial S} w(e) \frac{|e \cap S| \cdot |e \cap S^c|}{|e|}$$

For the last expression, if e is mapped to its 2-section, the numerator is the number of 'edges' that would be cut.

Hypergraph Laplacian

The Ncut problem can again be illustrated via a random walk with:

$$p(u,v) = \sum_{e \in E} \frac{w(e)h(u,e)}{d(u)} \frac{h(v,e)}{|e|}$$

with stationary distribution $\pi(v) = \frac{d(v)}{VolV}$.

We get the following results:

$\frac{VolS}{VolV} = \sum_{v \in S} \pi(v)$

$\frac{Vol\partial S}{VolV} = \sum_{u \in S} \sum_{v \in S^c} \pi(u)p(u,v).$

Hypergraph Laplacian

Solving the relaxed problem yields the same form as with graph, but with:

$$\Delta = I - D_v^{-1/2} H^T W D_e^{-1} H D_v^{-1/2}$$

When all $|e| = 2$, we get:

$$\Delta = \frac{1}{2}(I - D_v^{-1/2} W D_v^{-1/2})$$

which is half the graph Lapacian, so Δ can be defined as the *Hypergraph Laplacian*.

Hypergraph Laplacian

We define the same semi-supervised problem as with graphs:

$$f^* = \underset{f \in \mathbb{R}^n}{\operatorname{argmin}}(\Omega(f) + \mu||f - y||^2)$$

where:

$$\Omega(f) = \langle f, \triangle f \rangle = \frac{1}{2} \sum_{e \in E} \frac{1}{\delta(e)} \sum_{(u,v) \in e} w(e) \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2$$

The solution to the above problem is given by

$$f^* = (1 - \alpha)(I - \alpha S)^{-1} y, \ \alpha = (1 + \mu)^{-1}, \ S = I - \Delta.$$

Random Walk

The random walk described previously amounts to:

- from vertex u, pick an hyperedge e at random for which $u \in e$
- pick a vertex $v \in e$ at random and jump to v.

If all $|e| = 2$, we get $a_{ii} = \sum_{e; v_i \in e} w(e)/2 = d_i/2$ and for $e = (v_i, v_j)$ we get $a_{ij} = w(e)/2$, therefore

$$\tilde{A} = \frac{1}{2}(D_v + A)$$

where A is the (weighted) adjacency matrix of the graph representation of this hypergraph.

Therefore, the solution will differ if G is seen as a graph or an hypergraph!

Random Walk

We define a new random walk as follows:

- from vertex u, pick an hyperedge e at random for which $u \in e$
- pick a vertex $v \in e,\ v \neq u$ at random and jump to v.

We can view the above as a graph with a weighted adjacency matrix $\tilde{A} = (a_{ij})$ where:

$a_{ij} = \sum_{e;(v_i,v_j)\in e} \frac{w(e)}{|e|-1},\ a_{ii} = 0$

with row sum

$a_{i.} = \sum_{e;v_i\in e} w(e) = d(v_i)$.

Hypergraph Laplacian

In matrix form: $\tilde{A} = H^T W \tilde{D}_e^{-1} H - D_v$

with $\tilde{D}_e$ the diagonal matrix with entries $\frac{1}{|e|-1}$.

In this case, the *adjusted hypergraph Laplacian* takes the following form:
$\Delta = I - S$ with $S = D_v^{-1/2} \tilde{A} D_v^{-1/2} - I$

If all $|e| = 2$ we get $\tilde{A} = A$ where A is the (weighted) adjacency matrix of the graph representation of this hypergraph.

Weighted Graph Modularity

In a recent paper (`arXiv:1812.10869`), “hypergraph modularity” is defined as follows:

- define the (corrected) adjacency $\tilde{A}$ as we saw
- apply graph modularity on this weighted graph

This is very simple, but the hypergraph structure is only weakly considered (as in a 2-section graph)

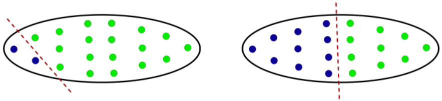
However, graph clustering such as Louvain or ECG can be used directly.

Iterative Reweighting

In the second part of the paper, the authors suggest an "Iterative Hyperedge Reweighting" framework.

In a nutshell:

- apply Louvain on the weighted graph described earlier
- for each hyperedge, re-weights such that
  - edges with unbalanced cuts get higher weight
  - edges with balanced cuts get lower weight



$$t = \left(\frac{1}{2} + \frac{1}{18}\right) \times 20 = 11.111 \qquad t = \left(\frac{1}{10} + \frac{1}{10}\right) \times 20 = 4$$

Iterative Reweighting

Algorithm 1: Iteratively Reweighted Modularity Maximization (IRMM)

input : Hypergraph incidence matrix H, vertex degree matrix D_v, hyperedge degree matrix D_e, hyperedge weights W

output : Cluster assignments *cluster_ids*, number of clusters c

```
// Initialize weights as W ← I if the hypergraph is unweighted
repeat
    // Compute reduced adjacency matrix
    A ← HW(D_e − I)^{-1} H^T
    // Zero out the diagonals of A
    A ← zero_diag(A)
    // Return number of clusters and cluster assignments
    cluster_ids, c = LOUVAIN_MOD_MAX(A)
    // Compute new weight for each hyperedge
```

Iterative Reweighting

// Compute new weight for each hyperedge
for $e \in E$ **do**
    // Compute the number of nodes in each cluster
    for $i \in [1, .., c]$ **do**
        // Set of nodes in cluster i
        $C_i \leftarrow cluster\_assignments[i]$
        $k_i = |e \cap C_i|$
    end
    // Compute new weight
    $w'(e) = \frac{1}{m} \sum_{i=1}^{c} \frac{1}{k_i+1}(\delta(e) + c)$
    // Take moving average with previous weight
    $W_{prev}(e) \leftarrow W(e)$
    $W(e) = \frac{1}{2}(w'(e) + W_{prev}(e))$
end
until $\|W - W_{prev}\| < threshold$

Questions and Ideas

- Gain better intuition behind various modularity functions
- Better, scalable clustering algorithm(s) with hypergraph modularities
- More experiments on real datasets
- Hybrid approach to hypergraph clustering:
  - use graph clustering algorithm(s)
  - use hypergraph-based objective function
  - use edge-reweighting or some other heuristic(s)
- Hypergraph benchmark with communities?