Statistical Data Analysis

Dr. Jana de Wiljes

12. Dezember 2022

Universität Potsdam

Graph Partitioning and Community Detection

Difference

Graph Partitioning (GP)

- partition vertices into given number of groups
- sizes of groups are (roughly) fixed
- many edges inside groups, few edges between groups
- goal: dividing network into smaller more manageable pieces
- example:
    - numerical solution of network processes on a parallel computer

Community Detection (CD)

- partition vertices into groups
- sizes of groups are not fixed
- many edges inside groups, few edges between groups
- goal: understanding structure of a network
- examples:
    - collaboration
    - related web pages

Why is partitioning hard?

Problem

Partition vertex set into two parts (*graph bisection*).

n vertices into parts of sizes n_1 and n_2 ($n_1 + n_2 = n$):

- $\frac{n!}{n_1!n_2!}$ possibilities (half of it if order is ignored and $n_1 = n_2$)
- using Stirling's formula $n! \approx \sqrt{2\pi n}(n/e)^n$ we get

$$\frac{n!}{n_1!n_2!} \approx \frac{n^{n+1/2}}{n_1^{n_1+1/2} n_2^{n_2+1/2}}$$

- for a balanced partition ($n_1 \approx n_2$):

$$\text{roughly } \frac{2^{n+1}}{\sqrt{n}} \text{ possibilities}$$

Therefore, exhaustive search is usually unfeasible.

Arbitrary number of classes

Methods for graph bisection can be generalised (other ways are possible):

- number of vertices: n
- number of classes: k
- define $l = k - 2^{\lfloor \log_2 k \rfloor}$
- for $r = 1, \dots, l$ (ascending order) apply graph bisection method with

$$n_1^{(r)} = n - \frac{r \cdot n}{k} \quad \text{and} \quad n_2^{(r)} = \frac{n}{k}$$

- apply (equally sized) graph bisection $\lfloor \log_2 k \rfloor$ times to the $n - \frac{l \cdot n}{k}$ vertices in the large class
- results in k classes of (almost) same size $\frac{n}{k}$

Graph cuts

Given a graph $G = (V, E, \omega)$.

- for disjoint $A, B \subset V$ define the *cut size*

$$\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} \omega_{ij}$$

- for a partition $\mathcal{A} = A_1, A_2, \ldots, A_k$ define

$$\mathrm{cut}(A_1, A_2, \ldots, A_k) = \sum_{i=1}^{k} \mathrm{cut}(A_i, \overline{A}_i)$$

- basic problem: find $\mathcal{A}$ minimizing $\mathrm{cut}(A_1, A_2, \ldots, A_k)$

Types of methods (usually heuristics)

Local

- Kernighan-Lin algorithm
- Fiduccia-Mattheyses algorithm
- ...

Global

- Spectral clustering
- ...

Kernighan-Lin algorithm

The algorithm (Kernighan and Lin in 1970) works as follows:

Algorithm: Partition vertex set into two parts (Kernighan & Lin)

Input: Graph $G = (V, E)$, positive integers n_1, n_2

Choose random partition V_1, V_2 of V with $|V_l| = n_l$;

repeat

    minPair $\leftarrow \sum \omega_{ij} + 1$;

    for (*$i \in V_1, j \in V_2$, neither i nor j has been swapped before* **)**

        if $\mathrm{cut}\left((V_1 \setminus \{i\}) \cup \{j\}, (V_2 \setminus \{j\}) \cup \{i\}\right) \leq \mathrm{minPair}$ **then**

            minPair $\leftarrow \mathrm{cut}\left((V_1 \setminus \{i\}) \cup \{j\}, (V_2 \setminus \{j\}) \cup \{i\}\right)$;

            $i_0 \leftarrow i, j_0 \leftarrow j$;

        end

    $V_1 = (V_1 \setminus \{i_0\}) \cup \{j_0\}, V_2 = (V_2 \setminus \{j_0\}) \cup \{i_0\}$ (swap i_0 and j_0)

until *no pair of unused vertices exists*;

Pick partition (from the different V_1, V_2) with smallest cut size;

Output: Partition V_1, V_2 of V

Runtime of Kernighan-Lin algorithm

At first glance we get:

- $0 \leq \min\{n_1, n_2\} \leq \frac{n}{2} \rightsquigarrow \mathcal{O}(n)$ swaps (worst case)
- $\frac{n}{2} \cdot \frac{n}{2}$ of considered pairs per swap (worst case)
- cut size change (swap i and j):

$$\sum_{v \in N(i) \cap V_1} \omega_{iv} + \sum_{v \in N(j) \cap V_2} \omega_{jv} - \sum_{v \in N(i) \cap V_2} \omega_{iv} - \sum_{v \in N(j) \cap V_1} \omega_{jv} + \omega_{ij}$$

- evaluating this expression costs $\mathcal{O}(m/n)$, where $m = |E|$, if stored in adjacency list
- $\rightsquigarrow$ runtime is $\mathcal{O}(m \cdot n^2)$ for one (!) round

Notes on the Kernighan-Lin algorithm

- swapping every round avoids local minima
- usually the algorithm is repeated several times with last output as input
- first partition is random $\rightsquigarrow$ sometimes algorithm is started more than once to get other (hopefully better) results
- convergence rate unknown
- for larger number of classes (i.e. $k \geq 3$) optimum might not be reached if applied
- Generalisation to hypergraphs: Fiduccia-Mattheyses algorithm

Spectral clustering

- mathematical foundation by Donath & Hoffman and Fiedler in 1973
- applications in various fields/for various problems
  - image segmentation
  - educational data mining
  - entity resolution
  - speech separation
  - ...

Laplacian matrix (and another graph definition)

The degree matrix $D(G)$ is given by

$$d_{ij} = \begin{cases} \sum\limits_{l \in N(i)} w_{il} & \text{if } i = j, \\ 0 & \text{else.} \end{cases}$$

Laplacian matrix:

$$L(G) = D(G) - W(G)$$

We also need:

$$\mathrm{vol}(A) = \sum_{ij \in E, i,j \in A} \omega_{ij} \text{ for } A \subset V \text{ (no double counting!)}$$

Spectral clustering and graph cuts

Clustering corresponds to finding cuts in graphs (see next slides; other interpretations possible), there are (usually) two types:

- RatioCut (balanced by number of vertices in each cluster)
- NCut (balanced by sum of edge weights in each cluster)

Formalization of RatioCut and NCut

- $\text{RatioCut}(A_1, A_2, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{|A_i|}$
    - $\sum_{i=1}^{k}(1/|A_i|)$ is minimized if all A_i have the same size
    - hence minimizing RadioCut balances clusters by their number of vertices (as desired)
- $\text{NCut}(A_1, A_2, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{\text{vol}(A_i)}$
    - $\sum_{i=1}^{k}(1/\text{vol}(A_i))$ is minimized if all $\text{vol}(A_i)$ coincide
    - hence minimizing NCut balances clusters by their edge weights (as desired)
- corresponding optimization problems are NP hard
- spectral clustering is a way to solve relaxed versions of those problems

Useful properties of the Laplacian

Proposition: The Laplacian matrix L $(= D - W)$ satisfies the following properties:

i For every vector $f \in \mathbb{R}^n$ we have

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2.$$

ii L is symmetric and positive semi-definite.

iii The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector.

iv L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Useful properties of the Laplacian

Proof:

i By the definition of d_i,

$$f^T L f = f^T D f - f^T W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$

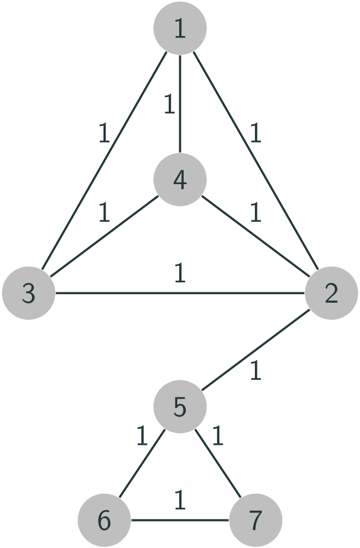
$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

ii The symmetry of L follows directly from the symmetry of W and D. The positive semi-definiteness is a direct consequence of Part (i), which shows that $f^T L f \geq 0$ for all $f \in \mathbb{R}^n$.

iii All eigenvalues are real (symmetric matrix). The rest follows easily from Part (ii) and the defining equation of eigenvalues.

iv This is a direct consequence of (i)-(iii).

Graph cuts and spectral clustering



$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

Connectivity

Proposition: The number of connected components of a graph is equal to the multiplicity of the first eigenvalue (which is 0) of the graph Laplacian matrix.

Relaxation of RatioCut for $k = 2$

Goal is to solve

$$\min_{A \subset V} \mathrm{RatioCut}(A, \overline{A})$$

Reformulation: for $A \subset V$ define $f = (f_1, \ldots, f_n) \in \mathbb{R}^n$ (some kind of indicator vector) via

$$f_i = \begin{cases} \sqrt{|\overline{A}|/|A|} & \text{if } v_i \in A, \\ -\sqrt{|A|/|\overline{A}|} & \text{if } v_i \in \overline{A}. \end{cases}$$

It can be shown that

$$f^T L f = |V| \cdot \mathrm{RatioCut}(A, \overline{A})$$

Relaxation of RatioCut for $k = 2$

Proof:

$$
\begin{aligned}
2f^T L f &= \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \\
&= \sum_{i \in A, j \in \overline{A}} w_{i,j} \left(\sqrt{\frac{|\overline{A}|}{|A|}} \right. \\
&\left. + \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 + \sum_{i \in \overline{A}, j \in A} w_{i,j} \left(-\sqrt{\frac{|\overline{A}|}{|A|}} - \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 \\
&= 2\mathrm{cut}(A, \overline{A}) \left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2 \right) \\
&= 2\mathrm{cut}(A, \overline{A}) \left(\frac{|A| + |\overline{A}|}{|A|} + \frac{|A| + |\overline{A}|}{|\overline{A}|} \right) \\
&= 2|V| \cdot \mathrm{RatioCut}(A, \overline{A})
\end{aligned}
$$

Relaxation of RatioCut for $k = 2$

Further, f is orthogonal to 1:

$$\sum_{i=1}^{n} f_i = \sum_{i \in A} \sqrt{\frac{|\overline{A}|}{|A|}} - \sum_{i \in \overline{A}} \sqrt{\frac{|A|}{|\overline{A}|}} = |A|\sqrt{\frac{|\overline{A}|}{|A|}} - |\overline{A}|\sqrt{\frac{|A|}{|\overline{A}|}} = 0.$$

Finally

$$\|f\|^2 = \sum_{i=1}^{n} f_i^2 = |A|\frac{|\overline{A}|}{|A|} + |\overline{A}|\frac{|A|}{|\overline{A}|} = |\overline{A}| + |A| = n.$$

Hence the minimization problem is equivalent to

$$\min_{A \subset V} f^T L f$$

subject to the given f.

Relaxation of RatioCut for $k = 2$

This problem is NP-hard since the solution vector only takes two particular values.

Relaxing this problem is possible:

$$\min_{f \in \mathbb{R}^n} f^T L f \text{ subject to } f \perp 1, \|f\| = \sqrt{n}$$

The Rayleigh-Ritz theorem (see e.g. Strang - Linear algebra and its applications) gives the solution of this problem via an eigenvector corresponding to the second smallest eigenvalue of L.

Remark: Solution has to be transformed!

Special case ($k = 2$) of unnormalized spectral clustering

Algorithm: Unnormalized spectral clustering ($k = 2$)

Input: Weight matrix (or any similarity matrix) $S \in \mathbb{R}^{n \times n}$

Construct similarity graph G;
Compute $L(G)$;
Compute Eigenvectors X_1 and X_2 of $L(G)$;
Build $U \in \mathbb{R}^{n \times 2}$ with X_1 and X_2 as columns;
Rows of U are $y_1, y_2, \ldots, y_n \in \mathbb{R}^2$;
Cluster $y_1, y_2, \ldots, y_n$ into Clusters C_1 and C_2 (k-means);

Output: Clusters $A_1 = \{j : y_j \in C_1\}$ and $A_2 = \{j : y_j \in C_2\}$

Relaxation of RatioCut for $k > 2$ (sketch)

- Define k indicator vectors $h_j = (h_{1,j}, h_{2,j}, \ldots, h_{n,j})^T$ via

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_j|} & \text{if } v_i \in A_j, \\ 0 & \text{else} \end{cases}$$

- Matrix $H \in \mathbb{R}^{n \times k}$ with columns $h_1, h_2, \ldots, h_k$ is orthogonal
- We have $h_i^T L h_i = \frac{\mathrm{cut}(A_i, \overline{A}_i)}{|A_i|}$ and $h_i^T L h_i = (H^T L H)_{ii}$
- Together this yields $\mathrm{RatioCut}(A_1, A_2, \ldots, A_k) = \mathrm{tr}(H^T L H)$
- Hence the minimum of $\mathrm{RatioCut}(A_1, A_2, \ldots, A_k)$ can be approximated by solving (the relaxed version)

$$\min_{H \in \mathbb{R}^{n \times k}} \mathrm{tr}(H^T L H) \text{ subject to } H^T H = I$$

General case of unnormalized spectral clustering

Algorithm: Unnormalized spectral clustering

Input: Weight matrix (or any similarity matrix) $S \in \mathbb{R}^{n \times n}$, number k of clusters

Construct similarity graph G;

Compute $L(G)$;

Compute Eigenvectors $X_1, X_2, \ldots, X_k$ of $L(G)$;

Build $U \in \mathbb{R}^{n \times k}$ with $X_1, X_2, \ldots, X_k$ as columns;

Rows of U are $y_1, y_2, \ldots, y_n \in \mathbb{R}^k$;

Cluster $y_1, y_2, \ldots, y_n$ into Clusters $C_1, C_2, \ldots, C_k$ (k-means);

Output: Clusters $A_1, A_2, \ldots, A_k$ with $A_i = \{j : y_j \in C_i\}$