UNIVERSITY OF CALIFORNIA, BERKELEY

Lecture 22 - CS:189

Oscar Ortega

July 16, 2021

1 SPECTRAL GRAPH CLUSTERING

Input: Weighted, undirected graph $G = (V, E)$, no self-edges

- $w_{i,j}$ = weight of graph $(i, j) = (j, i)$, zero if the edge is not in the graph.
- large weight connecting two vertices means the two vertices really want to be connected.

Goal: Cut G into 2 or more pieces G_i, such that they have similar sizes, but don't cut too much edge weight. For example, we could minimize the **sparsity**

$$\frac{\text{cut}(G_1, G_2)}{\text{Mass}(G_1)\text{Mass}(G_2)}$$

This formula is also known as the **cut ratio** Where $\text{cut}(G_1, G_2)$ = total weight of cut edges and $\text{Mass}(G)$ = number of vertices in G_1, or we could alternatively assign masses to vertices.
It turns our a lot of these graph partitioning problems, such as sparsest cut, min-bisection, and max-cut are **NP-Hard**, so we seek to use approximation algorithms and heuristics to approximate the optimal solution.

1.1 TACKLING SPARSEST-CUT

Let n = $|v|$, and let $y \in R^n$ be an **indicator vector**

$$y_i = \{1 \text{ if vertex } i \in G_1, -1 \in G_2\}$$

Then $w_{i,j} = \frac{(y_i - y_j)^2}{4} = w_{i,j}$ if (i,j) is cut and is 0 otherwise.

$$\rightarrow \text{cut}(G_1, G_2) = \frac{1}{4} \sum_{i,j \in E} (w_{i,j} y_i^2 - 2 w_{i,j} y_i y_j + w_{i,j} y_j^2) \tag{1.1}$$

$$= \frac{1}{4} y^T L y \tag{1.2}$$

Where L is the **Laplacian** matrix of the graph G The Laplacian matrix has the diagonal terms hold the sum of the weights corresponding to the edge and the the other non-diagonal terms i, j correspond to the negative weight of the edge i, j. We assume weights are positive.

1.2 THINGS TO NOTICE

The vector of all ones is always an eigenvector of the matrix L, with eigenvector 0.
Bisection: exactly half the vertices in each edge of the cut $\rightarrow 1^T y = 0$. Therefore, we can solve the minimum bisection problem as follows.

$$\min_y y^T L y$$

$$\text{s.t } \forall i, y_i \in \{-1, 1\}$$

$$1^T y = 0$$

We call the constraints the **binary constraint** and the **balance constraint**. However, this problem is still NP-hard. We can relax the binary constraint in order to make this problem feasible. The practice of relaxing discrete constraints to continuous constraints is very common. In this case we often round after finding optimal constraints.
In this problem we know constrain y to be in the hypersphere of radius $\sqrt{n}$ and it turns out the relaxed problem is as follows:

$$\min y^T L y$$

$$\text{s.t } y^T y = n$$

$$1^T y = 0$$

In other words we are trying to minimize the **Rayleigh quotient** of L and y.
Let λ_2 = the second-smallest eigenvalue of L and let the corresponding v_2 be the **Fiedler Vector**. of the matrix L

def spectralPartition():

1. compute fiedler vector v_2 of L
2. round v_2 with a sweep cut
3. def SweepCut(v_2):
    - sort components of v_2
    - try the $n-1$ cuts between successive components and choose the min-sparsity cut.

1.3 Vertex Masses

Let M be diagonal matrix with vertex masses on diagonal. then our new balance constraint is as follows:

$$1^T M y = 0$$

. So we can go ahead and relax with a new ellipsoid constraint.

$$y^T M y = \text{mass}(G)$$

Now we want fiedler vector of this new generalized eigensystem $Lv = \lambda M v$. Fact: Sweep cut finds a cut w/sparsity $\leq \sqrt{2\lambda_2 \max_i \frac{L_{i,i}}{M_{i,i}}}$ This is **Cheeger's Inequality** in a vertex with uniform weights this corresponds to the degree of the cut.
The optimal cut as sparsity $\geq \frac{\lambda_2}{2}$
If we wanted multiple cuts, we could go ahead and perform this cutting procedure recursively:

1.4 Normalized Cut

Set vertex i's mass $M_{i,i} = L_{i,i}$, popular for **image segmentation**