1 Overview

In this lecture, we will talk about the first part of graph sparsification, finding a sparse graph that preserves cuts. First we introduce definition of $\varepsilon$-cut sparsifier and Benczur-Karger Theorem. Then we talked about spectrum of graph and constructing sparsifier.

2 Introduction

Motivation: Sometimes graphs can be dense and take much more space and time to store or operate. For example, a graph can have $O(n^2)$ edges ($n$ is the number of vertices).

Hope: We want to find a sparse version of the graph, while maintaining useful properties like MIN-CUT, MAX-CUT, s-t CUT and connectivity (distance may not maintain).

Recall: Define $G = (V, E)$ is an undirected weighted graph with weights $W : E \rightarrow \mathbb{R}^+$ ($w_{ij} \geq 0$ for edge $(i, j)$), the capacity of a cut $(S, \overline{S})$: $W_G(S, \overline{S}) = \sum_{(i,j) \in E, i \in S, j \notin S} w_{ij}$

2.1 $\varepsilon$-cut sparsifier

Definition 1. Graph $H$ is an $\varepsilon$-cut sparsifier of $G$, if for every cut $(S, \overline{S})$

$$(1 - \varepsilon)W_G(S, \overline{S}) \leq W_H(S, \overline{S}) \leq (1 + \varepsilon)W_G(S, \overline{S})$$

Remark 1. Graph $G$ and $H$ share the same vertices.

Remark 2. Want $H$ to have fewer edges. Usually edges in $H$ are subset of edges in $G$ (not required by definition).

Theorem 1. [Benczur – Karger] For every graph $G$, there exists an $\varepsilon$-cut sparsifier $H$ with at most $O\left(\frac{n \log n}{\varepsilon^2}\right)$ weighted edges (this bound is not tight, we can do $O\left(\frac{n}{\varepsilon^2}\right)$ edges).

3 Spectrum of Graph

3.1 Laplacian Definition

Definition 2. Laplacian matrix $L_G$ of a graph $G$ is an symmetric $n \times n$ matrix such that for any vector $\bar{x} \in \mathbb{R}^n$,

$$\bar{x}^T L_G \bar{x} = \sum_{(i,j) \in E} w_{ij}(x_i - x_j)^2$$
Example 1. Laplacian matrix of a graph with only one edge:

\[ \vec{x} \in E, \quad \vec{x}^T L_G \vec{x} = w(x_1 - x_2)^2 = w(x_1^2 + x_2^2 - 2x_1x_2) = (x_1 \quad x_2) \begin{pmatrix} w & -w \\ -w & w \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \]

\[ \therefore L_G = \begin{pmatrix} w & -w \\ -w & w \end{pmatrix} = w \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \]

3.2 Constructing Laplacian

For edge \((i, j)\), define \(\vec{b}_{ij} = \vec{e}_i - \vec{e}_j\), in which \(\vec{e}_i\) and \(\vec{e}_j\) are basis vectors. Specifically, \(\vec{b}_{ij} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ -1 \\ \vdots \\ 0 \end{pmatrix}\) with \(i\)th row being 1 and \(j\)th row being -1.

Define \(L_{ij} = \vec{b}_{ij}^T \vec{w}_{ij}\), so we have \(L_G = \sum_{(i, j) \in E} L_{ij} = \sum_{(i, j) \in E} \vec{b}_{ij}^T \vec{w}_{ij}\)

3.3 \(\varepsilon\) – spectral sparsifier

Definition 3. Graph \(H\) is an \(\varepsilon\) – spectral sparsifier of \(G\), if for every \(\vec{x} \in \mathbb{R}^n\)

\[ (1 - \varepsilon) \vec{x}^T L_G \vec{x} \leq \vec{x}^T L_H \vec{x} \leq (1 + \varepsilon) \vec{x}^T L_G \vec{x} \]

Claim: \(\varepsilon\) – spectral sparsifier \(\Rightarrow\) \(\varepsilon\) – cut sparsifier

for every cut \((S, \bar{S})\), take indicator vector \(\mathbb{1}_S(i) = \begin{cases} 1 & i \in S \\ 0 & i \notin S \end{cases}\)

\[ \mathbb{1}_S^T L_G \mathbb{1}_S = \sum_{(i, j) \in E} w_G(i, j)(x_i - x_j)^2 = W_G(S, \bar{S}) \quad (x_i - x_j)^2 = \begin{cases} 1 & (i, j) \in (S, \bar{S}) \\ 0 & \text{otherwise} \end{cases} \]

Similarly, we have \(\mathbb{1}_S^T L_H \mathbb{1}_S = W_H(S, \bar{S})\)

4 Constructing Sparsifier

There are two ideas of constructing sparsifier for a graph:

1. Uniform Sampling.

   We could uniformly sample edges from original graph and form a sparsifier. However, this cannot work. Intuitively, if an edge is in small cut, it is more important to have this edge in the sparsifier than other edges.
2. Important Sampling.

First we assign a probability \( P_{ij} \) for each edge \((i, j)\), which means we keep edge \((i, j)\) with probability \( P_{ij} \). If the edge is kept, set its weight to be \( \frac{w_{ij}}{P_{ij}} \).

**Claim:** Let \( H \) be a random graph generated by above process, we have \( \mathbb{E}[L_H] = L_G \).

we know \( L_G = \sum_{(i,j) \in E} b_i b_j^T w_{ij} \). After this process, \( w_{ij} = \begin{cases} \frac{w_{ij}}{P_{ij}} & \text{with probability } P_{ij} \\ 0 & \text{with probability } 1 - P_{ij} \end{cases} \).

So the expectation of \( L_H \) is the same as \( L_G \).