Improved Dynamic Graph Learning through Fault-Tolerant Sparsification

Chun Jiang Zhu, Sabine Storandt, Kam-Yiu Lam, Song Han, Jinbo Bi
Motivations

• Consider the problem of solving certain graph regularized learning problems
  • For example, suppose vector $\beta^*$ is a smooth signal over vertices in a graph $G$, and $y$ is the corresponding observations
  
  $$
  \min_{\beta \in \mathbb{R}^n} \|y - \beta\|^2 + \lambda \beta^T L_G \beta.
  $$
  
  • Solve

• Solution $\hat{\beta} = (I + \lambda L_G)^{-1} y$ can be obtained in $\tilde{O}(m)$ time by an optimal SDD matrix solver
Motivations

• Solving systems in Laplacians matrices can be performed approximately more efficiently if a sparse approximation $H$ to the Laplacian is maintained

$$\min_{\beta \in \mathbb{R}^n} \|y - \beta\|^2 + \lambda' \beta H \beta^T$$

$$\tilde{\beta} = (I + \lambda' L_H)^{-1}$$

which can be obtained in $\tilde{O}(n)$ time

• How about when the graph changes?
Motivations

• We introduce the notion of fault-tolerant sparsifiers, that is sparsifiers that stay sparsifiers even after the removal of vertices / edges

• Specifically, we
  • Prove that these sparsifiers exist
  
  • Show how to compute them efficiently in nearly linear time

• Improve upon previous work on dynamically maintaining sparsifiers in certain regimes
Fault-Tolerant Sparsifiers

**Definition 1.** For a graph $G(V, E)$, a positive integer $f$ and parameter $\epsilon \in (0, 1)$, a re-weighted subgraph $H(V, E' \subseteq E)$ is an $f$-**VFT** ($f$-**EFT**) $(1 \pm \epsilon)$-**spectral sparsifier**, if for all vertex (edge) sets $F \subseteq V$ ($F \subseteq E$) of size $|F| \leq f$, $(1 - \epsilon)L_{G-F} \leq L_{H-F} \leq (1 + \epsilon)L_{G-F}$ holds.

**Definition 3.** For a graph $G(V, E)$, a positive integer $f$ and parameter $\epsilon \in (0, 1)$, a re-weighted subgraph $H(V, E' \subseteq E)$ is an $f$-**VFT** ($f$-**EFT**) $(1 \pm \epsilon)$-**cut sparsifier** if, for all vertex (edge) sets $F \subseteq V$ ($F \subseteq E$) of size $|F| \leq f$, $(1 - \epsilon)L_{G-F} \preceq \{0,1\} L_{H-F} \preceq \{0,1\} (1 + \epsilon)L_{G-F}$ holds.
Example

Figure 1: 1-\textit{FT} cut sparsifiers of $G$: $H_1$ and $H_2$. (a) $G$ with 36 edges and edge weight 1. (b) $H_1$ with 18 edges and edge weight 2. (c) $H_2$ with 12 edges and edge weight 3. Without loss of generality consider that $v$ is faulty. The \textit{Min-Cut} of $G - \{v\}$ is 5, while the \textit{Min-Cut} of $H_1 - \{v\}$ and $H_2 - \{v\}$ are 4 and 3, respectively. Then $H_1$ and $H_2$ are 1-\textit{FT} $(1 \pm 0.2)$-cut sparsifier and $(1 \pm 0.4)$-cut sparsifier of $G$, respectively.
Main Theorems

**Theorem 1.** For an $n$-vertex $m$-edge graph $G$, a positive integer $f$, a parameter $\epsilon \in (0, 1)$ and $\rho > 1$, an $f$-VFT ($f$-EFT) $(1 \pm \epsilon)$-spectral sparsifier for $G$ of expected size $O(fn \log \rho + n \log^2 n \log^3 \rho/\epsilon^2 + m/\rho)$ w.h.p. can be constructed.

**Theorem 7.** For an $n$-vertex $m$-edge graph $G$, a positive integer $f$, a parameter $\epsilon \in (0, 1)$, $\rho > 1$ a constant $C_\epsilon > 0$ and a parameter $c > 1$, Algorithm 4 constructs an $f$-VFT ($f$-EFT) $(1 \pm \epsilon)$-cut sparsifier for $G$ of expected size $O(fn \log \rho + n \log^2 n \log^3 \rho/\epsilon^2 + m/\rho)$, with probability at least $1 - n^{-c}$.
Main Techniques for $FT$ spectral sparsifiers

- Use $FT$ spanners and random sampling for constructing $FT$ sparsifiers

- Inspired by the sparsification algorithm (Koutis & Xu, 2016)

- (1) First constructs an $(f + t)$-$FT$ spanner for the input graph $G$ by any $FT$ graph spanner algorithms

- (2) Then uniformly samples each non-spanner edge with a fixed probability $1/4$, and multiplies the edge weight of each sampled edge by 4, to preserve the edge’s expectation

Main Techniques for FT spectral sparsifiers

• The $(f + t)$-FT spanner guarantees that even in the presence of at most $f$ faults, each edge not in the spanner has $t$ edge-disjoint paths between its endpoints in the spanner, showing its small effective resistance in $G$

• By the matrix concentration bounds (Harvey, 2012), we can prove that the resulting subgraph is a sparse $FT$ spectral sparsifier


Figure 2: A faulty vertex set $F = \{w_1, \cdots, w_f\}$ of size $\hat{f}$ can invalidate at most $\hat{f}$ paths out of $f$ vertex-disjoint paths between endpoints $u$ and $v$ of an edge $e(u,v)$. Here $f = 5$ and $\hat{f} = 3$. 
Using $FT$ sparsifiers in subsequent learning tasks

• At a time point $t > 0$,
  • For each vertex $v$ (edge $e$) insertion into $G_{t-1}$, if $v$ ($e$) is in $H$, add $v$ and its associated edges in $H$ ($e$ itself) to $H_{t-1}$
  • For each vertex $v$ (edge $e$) deletion from $G_{t-1}$, if $v$ ($e$) is in $H_{t-1}$, remove $v$ and its associated edges ($e$) from $H_{t-1}$

• These only incur a **constant** computational cost per edge update

• More importantly, the resulting subgraph is guaranteed to be a spectral sparsifier of the graph $G_t$ at the time point $t$, under the assumption that $G_t$ differs from $G_0$ by a bounded amount

• We give stability bounds to quantify the impact of the $FT$ sparsification on the accuracy of subsequent graph learning tasks
**FT Cut Sparsifiers**

- There exists graph-based learning based on graph cuts and using cut-based algorithms, instead of spectral methods
  - *Min-Cut* for SSL (Blum & Chawla, 2001), *Max-Cut* for SSL (Wang et al., 2013), *Sparsest-Cut* for hierarchical learning (Moses & Vaggos, 2017), and *Max-Flow* for SSL (Rustamov & Klosowski, 2018)

- Construction:
  - The same framework as that for *FT* spectral sparsifiers
  - Define and use a variant of maximum spanning trees, called *FT α-MST*, to preserve edge connectivities

---


Experiments

• Dataset: Facebook social network data with 4309 vertices and 88234 edges from the SNAP

• Method: Compared our algorithm FTSPA with a baseline SPA, which constructs a spectral sparsifier from scratch at every time point, and the exact method EXACT

• The speedup is over $10^5$, while the accuracies are not significantly affected by the FT sparsification!

<table>
<thead>
<tr>
<th>Methods</th>
<th>Update Time</th>
<th>Speedup</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPA</td>
<td>34.2 s</td>
<td>1</td>
<td>12978 ± 30</td>
</tr>
<tr>
<td>FTSPA</td>
<td>0.3 ms</td>
<td>$&gt;10^5$</td>
<td>16502 ± 41</td>
</tr>
</tbody>
</table>

Table 1: Update time and # edges of SPA and FTSPA
Accuracy of Laplacian-regularized estimation (\(\sigma\) is the SD of Gaussian noises added to y)

(a) Estimation, \(\sigma = 0.1\)  
(b) Estimation, \(\sigma = 0.01\)