Algorithms for Lipschitz Learning on Graphs

Sushant Sachdeva
Yale Institute of Network Sciences

Rasmus Kyng
Anup Rao
Dan Spielman
Learning on Graphs
Learning on Graphs
Learning on Graphs
Learning on Graphs
Learning on Graphs
Learning on Graphs

Methods for smooth (Lipschitz) learning
Learning on Graphs

Methods for smooth (Lipschitz) learning

Theoretically interesting
Learning on Graphs

Methods for smooth (Lipschitz) learning

Theoretically interesting

Quickly computable
Learning on Graphs

Methods for smooth (Lipschitz) learning

Theoretically interesting

Quickly computable

Noise tolerant
Learning on Graphs

Methods for smooth (Lipschitz) learning

- Theoretically interesting
- Quickly computable
- Noise tolerant
- Performs well on real-world data
THE BASICS
Preliminaries

Graph $G(V,E,\text{len})$
Graph \( G(\mathcal{V}, \mathcal{E}, \mathcal{I}) \)

\[ |\mathcal{V}| = n, \quad |\mathcal{E}| = m \]
Preliminaries

Graph $G(V,E,\text{len})$

$|V| = n, |E| = m$

$\text{len} = \text{edge lengths}$
Preliminaries

Graph $G(V,E,\text{len})$

$|V| = n$, $|E| = m$

$\text{len} = \text{edge lengths}$

Undirected (for now)
Preliminaries

\[ T \subseteq V, \text{ terminals} \]
Preliminaries

\[ T \subseteq V, \text{ terminals} \]

\[ \nu : T \rightarrow \mathbb{R}, \text{ labels} \]
$T \subseteq V$, terminals

$\nu : T \rightarrow \mathbb{R}$, labels

A (partial) assignment
Preliminaries

Guess labels $w$ at all vertices:
Guess labels $w$ at all vertices:

1. $w$ agrees with $v$ on terminals ($w$ extends $v$)
Preliminaries

Guess labels $w$ at all vertices:

1. $w$ agrees with $v$ on terminals ($w$ extends $v$)

2. $w$ is smooth across edges
Preliminaries

**Goal:** Compute a *smooth* extension of $\nu$
Preliminaries

**Goal:** Compute a *smooth* extension of $\nu$

What is *smooth*?
Preliminaries

Goal: Compute a smooth extension of $\nu$

What is smooth?

For $(x, y) \in E$,

Define gradient

$$\text{grad}[w](x, y) = \frac{w(x) - w(y)}{\text{len}(x, y)}$$
Two Smooth Extensions

Inf-minimizer
Two Smooth Extensions

Inf-minimizer

Find $\omega$ that extends $\nu$
Two Smooth Extensions

Inf-minimizer

Find $w$ that extends $v$

and minimizes

$$\|\text{grad}[w]\|_{\infty} = \max_{(x,y) \in E} \frac{|w(x) - w(y)|}{\text{len}(x,y)}$$
Two Smooth Extensions

Find \( w \) that extends \( v \) and minimizes

\[
\| \text{grad}[w] \|_\infty = \max_{(x,y) \in E} \frac{|w(x) - w(y)|}{\text{len}(x, y)}
\]

Lipschitz constant
Two Smooth Extensions

Inf-minimizer

Not necessarily unique!

Best Lipschitz constant = 1
Two Smooth Extensions

Not necessarily unique!

Inf-minimizer

Best Lipschitz constant = 1
Two Smooth Extensions

Inf-minimizer

Not necessarily unique!

Best Lipschitz constant = 1
Two Smooth Extensions
Two Smooth Extensions

Amongst all $w$ that extend $v$

minimize the largest gradient
Two Smooth Extensions

Amongst all $w$ that extend $v$

minimize the largest gradient

then, minimize the second largest gradient
Two Smooth Extensions

Amongst all $w$ that extend $v$
minimize the largest gradient
then, minimize the second largest gradient
then, the third largest gradient, etc.
Two Smooth Extensions

Amongst all $w$ that extend $v$

minimize the largest gradient

then, minimize the second largest gradient

then, the third largest gradient, etc.
Two Smooth Extensions

Lex-minimizer

Amongst all $w$ that extend $v$

minimize the largest gradient

then, minimize the second largest gradient

then, the third largest gradient, etc.

Lex-minimizer is unique!
Two Smooth Extensions

Lex-minimizer

Amongst all $w$ that extend $v$

minimize the largest gradient

then, minimize the second largest gradient

then, the third largest gradient, etc.

Lex-minimizer is unique!
Other Smooth Extensions
Other Smooth Extensions

2-minimizer

[Zhu et al. ‘03]
Other Smooth Extensions

Find $w$ that extends $v$

and minimizes $\|\text{grad}[w]\|_2$  

[Zhu et al. ‘03]
Other Smooth Extensions

Find $w$ that extends $v$

and minimizes $\|\text{grad}[w]\|_2$

minimizes $\sum_{(x,y) \in E} \left( \frac{w(x) - w(y)}{\text{len}(x, y)} \right)^2$

[Zhu et al. ‘03]
Other Smooth Extensions

Find $w$ that extends $v$

and minimizes $\|\text{grad}[w]\|_2$

\[ \sum_{(x,y) \in E} \left( \frac{w(x) - w(y)}{\text{len}(x,y)} \right)^2 \]

Fast algorithms via Laplacian Solvers

[Zhu et al. ‘03]
Concern with 2-Minimizer

[Nadler et al. ‘09] Large geometric graphs with few terminals 2-minimizer collapses to a constant
Concern with 2-Minimizer

[Nadler et al. ‘09] Large geometric graphs with few terminals 2-minimizer collapses to a constant

Simple example: 2-D grids, with 2 terminals
2-Minimizer vs Lex
2-Minimizer vs Lex
2-Minimizer vs Lex
Other Smooth Extensions
Other Smooth Extensions

\[ p\text{-minimizer} \]

[Alamgir et al. ‘11]
Other Smooth Extensions

Find \( w \) that extends \( v \)
and minimizes \( \| \text{grad}[w] \|_p \)

minimizes \( \sum_{(x,y) \in E} \left| \frac{w(x) - w(y)}{\text{len}(x,y)} \right|^p \)

[Alamgir et al. ‘11]
Other Smooth Extensions

Find $w$ that extends $v$ and minimizes $\|\text{grad}[w]\|_p$

Don’t flatten out for large $p$. Very costly to compute.

[Alamgir et al. ‘11]

$\sum_{(x,y) \in E} \left| \frac{w(x) - w(y)}{\text{len}(x, y)} \right|^p$
\( w_p = \text{p-minimizer amongst extensions of } v \)
p-Minimizer and Lex

\[ w_p = \text{p-minimizer amongst extensions of } v \]

\[ \lim_{p \to \infty} w_p = \text{lex}[v] \]

Follows from [Egger et al. ’90]
More Connections

A local definition for lex
  Analogous to 2-minimizers

Studied in Functional Analysis / PDE theory

[Jensen ‘93, Crandall et al. ‘01, Barles et al. ’01, Aronsson et al. ‘04, Milman ‘99, Peres et al. ‘11, Naor et al. ‘10, Sheffield et al. ‘10, etc]
How should we compute it?

ALGORITHMS
Some Definitions
Some Definitions

Given 2 terminals $x, y$, define

$$\nabla v(x, y) = \frac{v(x) - v(y)}{\text{dist}(x, y)}$$
Some Definitions

Given 2 terminals $x, y$, define

$$\nabla v(x, y) = \frac{v(x) - v(y)}{\text{dist}(x, y)}$$

Metric defined by $\text{len}$
Some Definitions

Given 2 terminals \( x, y \), define

\[
\nabla v(x, y) = \frac{v(x) - v(y)}{\text{dist}(x, y)}
\]

\( \nabla v(x, y) = 1 \)

Metric defined by len
Some Definitions

Given 2 terminals $x, y$, define

$$\nabla v(x, y) = \frac{v(x) - v(y)}{\text{dist}(x, y)}$$

Metric defined by $\text{len}$

$$\|\text{grad}[w_{\infty}]\|_{\infty} = \max_{x, y} \nabla v(x, y)$$

$\nabla v(x, y) = 1$
Steepest Terminal Pair

**Goal**: Find a terminal pair \((x,y)\) with maximum gradient \(\nabla v(x, y)\)
Steepest Terminal Pair

**Goal:** Find a terminal pair \((x, y)\) with maximum gradient \(\nabla v(x, y)\)

Lex-minimizer \(\rightarrow\) \(n\) calls to Steepest Terminal Pair
**Steepest Terminal Pair**

**Goal:** Find a terminal pair \((x,y)\) with maximum gradient \(\nabla v(x, y)\)

- Lex-minimizer \(\rightarrow\) \(n\) calls to Steepest Terminal Pair
- Inf-minimizer \(\rightarrow\) \(1\) call to Steepest Terminal Pair
Finding a Steepest Pair

**Goal**: Find a steepest terminal pair
Finding a Steepest Pair

**Goal:** Find a steepest terminal pair

First Attempt

Compute distances between all terminals \( O(mn) \) time
Finding a Steepest Pair

**Goal:** Find a steepest terminal pair

First Attempt

Compute distances between all terminals \(O(mn)\) time

\(O(mn^2)\) algorithm for lex

[Lazarus et al. ‘99]
Goal: Find a steepest terminal pair
Finding a Steepest Pair

**Goal**: Find a steepest terminal pair

**[Theorem]**
Algorithm to find a steepest terminal pair in expected $O(m)$ time
Finding a Steepest Pair

**Goal:** Find a steepest terminal pair

**Theorem:**
Algorithm to find a steepest terminal pair in expected $O(m)$ time

**Theorem:**
Can compute the lex-minimizer in expected $O(mn)$ time
Can compute an inf-minimizer in expected $O(m)$ time
Simple Case : Star Graph

**Goal:** Find a steepest terminal pair in $O(n)$ time

Simple case : a star graph
**Simple Case : Star Graph**

**Goal:** Find a steepest terminal pair in $O(n)$ time

Simple case: a star graph

$\binom{n}{2}$ terminal pairs

$\binom{n}{2} \gg n$
Directed Graphs

Surprisingly, the theory/algorithms extend to directed graphs

Consider difference only along edge direction

$$\text{grad}^+[w](x, y) = \max \left\{ \frac{w(x) - w(y)}{\text{len}(x, y)}, 0 \right\}$$
Directed Graphs

Surprisingly, the theory/algorithms extend to directed graphs

Consider difference only along edge direction

$$\text{grad}^+[w](x, y) = \max \left\{ \frac{w(x) - w(y)}{\text{len}(x, y)}, 0 \right\}$$

Lex-minimizer not necessarily unique!
Directed Graphs

Surprisingly, the theory/algorithms extend to directed graphs

Consider difference only along edge direction

\[ \text{grad}^+[w](x, y) = \max \left\{ \frac{w(x) - w(y)}{\text{len}(x, y)}, 0 \right\} \]

Lex-minimizer not necessarily unique!

[Theorem]

Can compute a directed lex-minimizer in expected \(O(mn)\) time

Can compute a directed inf-minimizer in expected \(O(m)\) time
Can we handle noise?

STABILITY AND REGULARIZATION
Noise Stability

What if the original labels are noisy?
Noise Stability

What if the original labels are noisy?

[Theorem]
Suppose \( w, v \) are s.t. for all terminals \( t \)
\[ |v(t) - w(t)| \leq \epsilon \]
Then,
\[ \| \text{lex}[v] - \text{lex}[w] \|_{\infty} \leq \epsilon \]
$\ell_1$ Regularization

$\ell_1$ Regularization: Allowed to relax lengths by total budget $B$
Find best possible inf-minimizer
$\ell_1$ Regularization

$\ell_1$ Regularization: Allowed to relax lengths by total budget $B$
Find best possible inf-minimizer

[Theorem]
Can solve $\ell_1$ regularization in $O\left(m^{3/2}\right)$ time

Uses interior point methods & fast Laplacian solvers
$l_0$ Regularization

**Outlier Removal**: Allowed to discard any $k$ terminals
Find best possible inf-minimizer
\( l_0 \) Regularization

**Outlier Removal**: Allowed to discard any \( k \) terminals
Find best possible inf-minimizer

**[Theorem]**
Can perform outlier removal for inf-minimizer in poly-time
\( \ell_0 \) Regularization

**Outlier Removal** : Allowed to discard any k terminals
  Find best possible inf-minimizer

---

**[Theorem]**

Can perform outlier removal for inf-minimizer in poly-time

Analogous problem for \( \ell_2 \)-minimizer is \( \text{NP} \)-hard
How well does this work?

EXPERIMENTS
Fast Implementations

Lex-minimizer has a lot of structure
Leads to faster implementations
Fast Implementations

Lex-minimizer has a lot of structure
Leads to faster implementations

Code available on GitHub
1. Theoretically correct
2. Run fast in practice
Fast Implementations

Lex-minimizer has a lot of structure
Leads to faster implementations

**Code available on GitHub**
1. Theoretically correct
2. Run fast in practice

<table>
<thead>
<tr>
<th></th>
<th>250k</th>
<th>500k</th>
<th>1m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random regular/</td>
<td>12 s</td>
<td>~30 s</td>
<td>80-90 s</td>
</tr>
<tr>
<td>Random Delauney</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tested on grid graphs, random graphs, random regular graphs, real world network graphs from SNAP etc...
Detecting Spam Webpages

**Objective:** Detect spam webpages

**Data Set:** webspam-uk2006-2.0 [Castillo et al. ‘06]

**Comparison:** Random walk based methods
[Zhou et al. ‘07]
Detecting Spam Webpages

**Objective:** Detect spam webpages

**Data Set:** webspam-uk2006-2.0 [Castillo et al. ‘06]

**Comparison:** Random walk based methods [Zhou et al. ‘07]
Detecting Spam Webpages

Objective: Detect spam webpages

Data Set: webspam-uk2006-2.0 [Castillo et al. ‘06]

Comparison: Random walk based methods [Zhou et al. ‘07]
Detecting Spam Webpages

Label samples with trust values - 0/1
Detecting Spam Webpages

Label samples with trust values - 0/1

Compute lex-minimizer to extend observations to all nodes
Detecting Spam Webpages

Label samples with trust values - 0/1

Compute lex-minimizer to extend observations to all nodes

Flag all below a threshold as ‘Spam’
Comparison

5% labels used for training

- RandWalk
- DirectedLex

Fraction of all spam flagged (RECALL)

Fraction correctly flagged as spam (PRECISION)
Conclusion

✓ Suggest using lex and inf minimizer for graph inference

✓ Fast and practical algorithms for computing them

✓ Efficient algorithms for regularization, and robustness to noise
Conclusion

✓ Suggest using lex and inf minimizer for graph inference

✓ Fast and practical algorithms for computing them

✓ Efficient algorithms for regularization, and robustness to noise

? Can we prove theoretical learning guarantees?
? More interesting data-sets to test performance
  (Looking for suggestions)
Conclusion

✔ Suggest using lex and inf minimizer for graph inference

✔ Fast and practical algorithms for computing them

✔ Efficient algorithms for regularization, and robustness to noise

❓ Can we prove theoretical learning guarantees?

❓ More interesting data-sets to test performance
  (Looking for suggestions)

Thanks!