Graph Signal Processing

Methods and Applications

James Sharpnack

JSM 2018

UC Davis, Statistics Department
Work supported by NSF DMS-12-23137
Table of contents

1. Introduction

2. Methods for Graph Signal Processing

3. Non-parametric estimation with K-nearest neighbors GTF

4. Density estimation over transportation networks

5. Matrix denoising via Cartesian product GTF
Collaborators

Oscar Hernan Shitong Wei Veeranjaneyulu Sadhanala
Madrid Padilla (Postdoc) (UCD Grad Student) (CMU Grad Student)

And many others... Ryan Tibshirani, Robert Bassett, Yu-Xiang Wang, Alex Smola, Aarti Singh, Akshay Krishnamurthy, Daniela Witten, Yanzhen Chen
Introduction
Graph Signal Processing

Figure 1: Water contamination sensors

Figure 2: Air traffic graph for flu counts

Graph signal processing: noisy measurement $y_v$ on vertex $v$ in graph $G$
Social Media Applications

Anomalous Behavior in Groups

Best Ad Placement

Detecting Viral Content

Image from http://williamturkel.net
Applications in this Talk

GSP is a *flexible* framework that admits *efficient algorithms*. In this talk we will see...

- non-parametric function estimation,
- road network segmentation,
- network modeling with graphons
Methodologies for GSP...

- graph kernels [Smola and Kondor, 2003],
- graph wavelets [Gavish et al., 2010, Crovella and Kolaczyk, 2003],
- graph trend filtering [Wang et al., 2016].

—— representative publications above
Methods for Graph Signal Processing
Graph Wavelets

Haar wavelet basis based on random spanning trees
[Sharpnack et al., 2013]
Challenges: determining a good hierarchical decomposition of graph to produce wavelets.
Graph Derivatives

Let $j$ be an edge between vertices $i_0, i_1$ then $\nabla \in \mathbb{R}^{m\times n}$ (the graph has $m$ edges, $n$ vertices)

$$\nabla_{j, i_0} = \sqrt{W_{i_0, i_1}}, \quad \nabla_{j, i_1} = -\sqrt{W_{i_0, i_1}}.$$

Then $\Delta^{(1)} := \nabla$ is the graph derivative and it turns out that $\Delta^{(2)} := \Delta = \nabla^\top \nabla$ is the combinatorial Laplacian. We can define higher order graph derivatives

$$\Delta^{(k+1)} = \nabla^\top \Delta^{(k)}, \quad k \text{ is odd}$$

$$\Delta^{(k+1)} = \nabla \Delta^{(k)}, \quad k \text{ is even}.$$

$k$ is called the ‘order’ of the derivative.

- Many graph kernels are functions of these operators.
- Spectrum of derivatives yield graph Fourier transform.
Graph Trend Filtering

Define the $k$th order **Graph Trend Filter** to be

$$
\min_{x \in \mathbb{R}^n} \|y - x\|^2_2 + \lambda \|\Delta^{(k+1)} x\|_1
$$

which evaluates to (for unweighted graphs)

$$
\min_{x \in \mathbb{R}^n} \|y - x\|^2_2 + \lambda \sum_{i,j : i \sim j} |x_i - x_j|, \quad k = 0 \quad \text{(Fused Lasso)}
$$

$$
\min_{x \in \mathbb{R}^n} \|y - x\|^2_2 + \lambda \sum_{i} \left| \sum_{j : j \sim i} (x_i - x_j) \right|, \quad k = 1 \quad \text{(2nd order)}
$$

Graph Sobolev norm regularization with $p = 1$. [Wang et al., 2016]
Piecewise Polynomials on Graphs

GTF with $k = 0$
Piecewise Polynomials on Graphs

GTF with $k = 1$
Piecewise Polynomials on Graphs

GTF with $k = 2$
Local Adaptivity

True signal  Noisy observations  GTF, 80 df

Laplacian smoothing  80 df  Laplacian smoothing  134 df  Wavelet smoothing  313 df
Non-parametric estimation with K-nearest neighbors GTF
Given samples \( \{x_i, y_i\}_{i=1}^n \) over a metric space \( (\mathcal{X}, d_{\mathcal{X}}) \times \mathbb{R} \) do:

1. Form the K-nearest neighbor graph \( G_K \) over \( \{x_i\}_{i=1}^n \) with an undirected edge for every K-NN.
2. Run the graph trend filtering over \( y, G_K \) yielding \( \hat{y} \).
3. Predict for new points \( x^* \) by standard KNN prediction with \( \{x_i, \hat{y}_i\}_{i=1}^n \).
Figure 3: Ratio of the misclassification rate of 5NN-GTF to 5NN Laplacian regularization, for graph-based semi-supervised learning, on the 11 most popular UCI classification data sets (MAD refers to modified absorption problem with prior). [Wang et al., 2016]
Figure 4: A heatmap of $n = 5000$ draws from a density (left) that is higher around where there is a change in the regression function (right) [Padilla et al., 2018].
**Figure 5:**  *Top Left:* The regression function evaluated at an evenly-spaced grid of size $100 \times 100$ in $[0, 1]^2$. *Top Right:* The estimate obtained via $K$-NN-FL. *Bottom Left:* The estimate of obtained via CART. *Bottom Right:* The estimate of obtained via $K$-NN regression [Padilla et al., 2018].
Figure 6: The MSE comparisons for the previous slide simulation.
Figure 7: A new simulation with uniform X density, but boundary that is not axis aligned.
Figure 8: Computational time in seconds (left), averaged over 150 Monte Carlo simulations and optimized MSE (right).
Density estimation over transportation networks
Processes on Geometric Networks
Processes on Geometric Networks
Processes on Geometric Networks
Log-density Estimation on Geometric Networks

\[
\min_{g \in BV} \frac{-1}{n} \sum_{i=1}^{n} g(x_i) + \lambda TV(g)
\]

s.t. \[ \int_{G} e^g = 1 \]

- \( g \) is log-density
- TV is the total variation (if \( g \) is differentiable then this is \( \int |g'| \)) over the geometric network
- Geometric network is a collection of 1D manifolds that end and start at vertices.
Proposition [Bassett and Sharpnack, 2018]

Total variation penalized density estimation can be solved with the following program,

$$\min_{f \in \mathbb{R}^n} \frac{1}{2} f^T S f + \nu^T f + \| \nabla f \|_1$$

has the same optimality conditions $e^g$ where $S$ is a diagonal matrix and $\nu$ is a vector.
∇ is the graph derivative for an expanded graph over the observed points and the original vertices.
Simulated crash data
Simulated crash data
Terrorist incidents from 2013 to 2016, according to the Global Terrorism Database within a road network in Baghdad.
Matrix denoising via Cartesian product GTF
Cartesian Power Graphs

**Figure 9:** KNN-PGFL method: the 2-NN graph $G_2$ (left) is learned from the adjacency matrix $A$ (middle) of the network $H$, then fused lasso is applied to Cartesian power graph, $G_2^{□2}$, with the $A_{ij}$ dyadic labels (right).
Power Graph Fused Lasso

(PGFL) is the solution to

$$\min_{P \in \mathbb{R}^{n \times n}} \|A - P\|_F^2 + \lambda (\|\nabla P\|_1 + \|\nabla P^\top\|_1).$$

(1)

To see that the RHS of (1) is the TV penalty over the Cartesian power graph, notice that

$$\|\nabla P\|_1 + \|\nabla P^\top\|_1 = \sum_{k=1}^{n} \|\nabla P_k\|_1 + \|\nabla (P^\top)_k\|_1$$

$$= \sum_{(i,j) \in E, k \in V} (|P_{k,i} - P_{k,j}| + |P_{i,k} - P_{j,k}|),$$

where $P_k$ is the $k$th column of $P$. 
Graphon Models

Graphon model

- Incidence matrix $A_{ij}$ has independent Bernoulli entries with probability matrix $P_{ij}$.
- Uniform $[0, 1]$ latent variables $\xi_i$ for each vertex $i$
- Graphon $f$ is bivariate function such that $P_{ij} = f(\xi_i, \xi_j)$

[Zhang et al., 2015] estimated a smooth graphon $f$ by first estimating a distance metric $\hat{d}$ between vertices and smoothing the matrix $A$ wrt $\hat{d}$. 


Fused Graphon Estimation

**Input:** network $H$ with adjacency matrix $A$, tuning parameter $\lambda > 0$

**Output:** partition of $V \times 2$, $S$, and an estimate $\hat{P}$ of $EA$.

1. Calculate the $\hat{d}_1$ distance matrix $\hat{D}_1 = (\hat{d}_1(i,j))_{i,j}$ (or any distance);
2. Generate the undirected KNN graph $G_K$: $(i,j) \in E$ if $i$ is a KNN of $j$ or vice versa;
3. Calculate $\hat{P}$ with Distributed PGFL on $G_K$, $A$, $\lambda$.

——see details in [Wei et al., 2018]
Figure 10: Plots left to right are the true probability matrix, the estimates using PGFL, Neigh Smoothing, Sorting-and-smoothing, SBM, and USVT.
### Table 1: Mean-square error comparisons

<table>
<thead>
<tr>
<th>Method</th>
<th>Graphon A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN-PGFL</td>
<td>7.39</td>
<td><strong>3.10</strong></td>
<td>17.54</td>
<td><strong>34.91</strong></td>
<td><strong>61.08</strong></td>
</tr>
<tr>
<td>Neigh. Smooth</td>
<td>13.68</td>
<td>9.55</td>
<td>17.16</td>
<td>45.18</td>
<td>66.76</td>
</tr>
<tr>
<td>SAS</td>
<td><strong>6.29</strong></td>
<td>9.20</td>
<td>23.68</td>
<td>97.90</td>
<td>190.38</td>
</tr>
<tr>
<td>SBM</td>
<td>37.65</td>
<td>6.60</td>
<td>35.77</td>
<td>44.45</td>
<td>62.68</td>
</tr>
<tr>
<td>USVT</td>
<td>7.05</td>
<td>9.61</td>
<td><strong>12.24</strong></td>
<td>50.34</td>
<td>71.94</td>
</tr>
</tbody>
</table>
Thanks for your time

http://jsharpna.github.io


References


