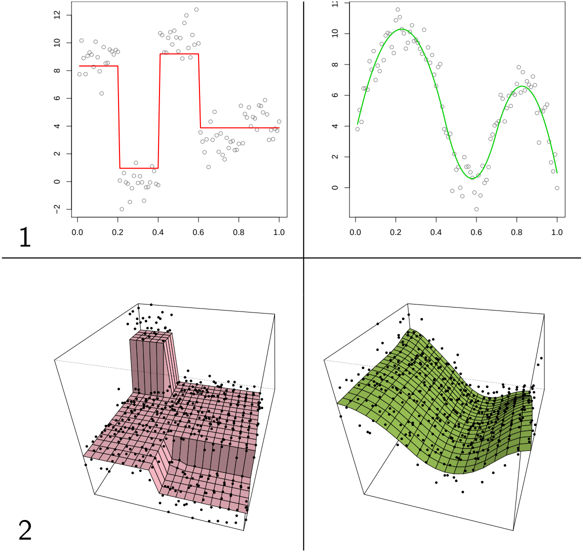
Trend Filtering, Falling Factorial Basis and Locally Adaptive Statistical Estimation on Graphs

Yu-Xiang Wang

Dept. of Machine Learning
Carnegie Mellon University

Collaborators: Ryan Tibshirani, James Sharpnack, Alex Smola, Veeranjaneyulu Sadhanala

Outline



Outline

1 Univariate trend filtering

2 The falling factorial basis

3 Trend filtering on Graphs

1 Univariate trend filtering
(Tibshirani, 2013, Annals of Statistics)

Nonparametric regression

Nonparametric regression: observe $(x_1, y_1), \dots (x_n, y_n) \in \mathbb{R} \times \mathbb{R}$ from model

$$y_i = f_0(x_i) + \epsilon_i, \quad i = 1, \dots n$$

Errors ϵ_i assumed to have zero mean. Want to estimate underlying regression function f_0, assumed to be smooth

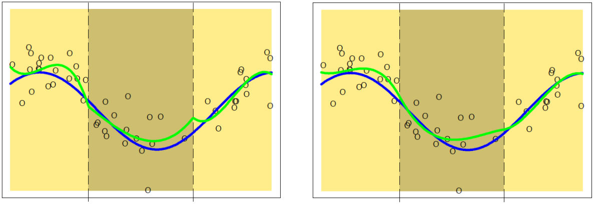
Rich literature, lots of interesting work. E.g.,

- Local polynomials
- Splines
- Kernels
- Wavelets

Relative newcomer in nonparametric regression: trend filtering, a close cousin to splines

Splines

Recall: a kth degree spline is a kth degree piecewise polynomial, that has continuous derivatives of orders $0, 1, \ldots k-1$ at its knots



The added (higher order) continuity constraints make the function smoother

Of course, key question is: how to choose knots?

Two canonical spline estimators

Consider regularized least squares problem:

$$\min_{\text{functions } f} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \cdot R(f)$$

where $\lambda \geq 0$ is a tuning parameter, R is a roughness penalty

Smoothing splines (Wahba 1990, Green & Silverman 1994) use $R(f) = \int \big(f^{(\frac{k+1}{2})}(t)\big)^2 \, dt$. Properties:

- Solution $\hat{f}$ is a (natural) spline of degree k
- Knots at all input points $x_1, \ldots x_n$
- Computationally fast
- Suboptimal rate for estimating functions of heterogeneous smoothness

Two canonical spline estimators

Consider regularized least squares problem:

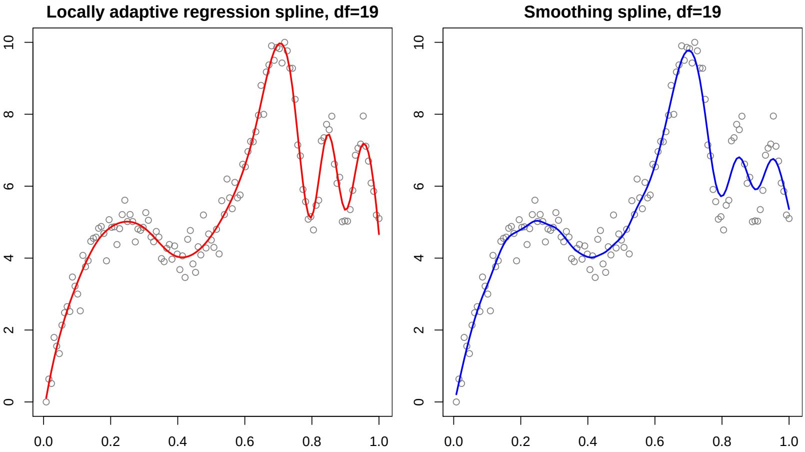
$$\min_{\text{functions } f} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \cdot R(f)$$

where $\lambda \geq 0$ is a tuning parameter, R is a roughness penalty

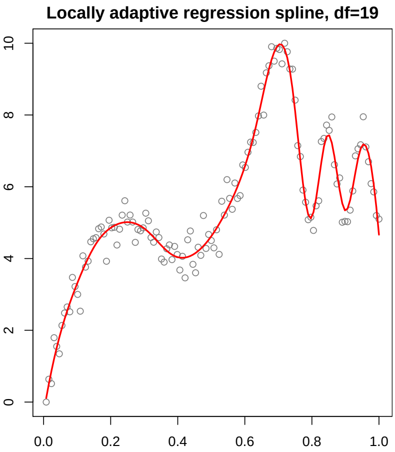
Locally adaptive regression splines (Mammen & van de Geer 1997) use $R(f) = \mathrm{TV}(f^{(k)})$. Properties:

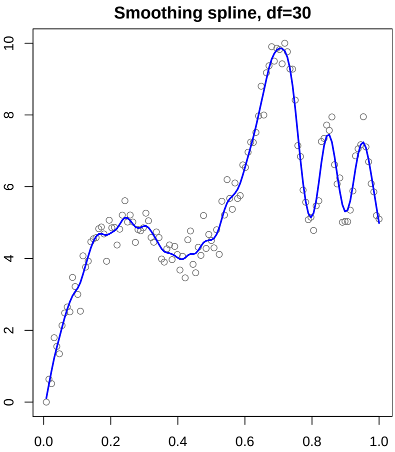
- Solution $\hat{f}$ is a spline of degree k
- Knots adaptively chosen among $x_1, \ldots x_n$
- Computationally slow
- Minimax optimal for estimating functions of heterogeneous smoothness

Example: comparing methods

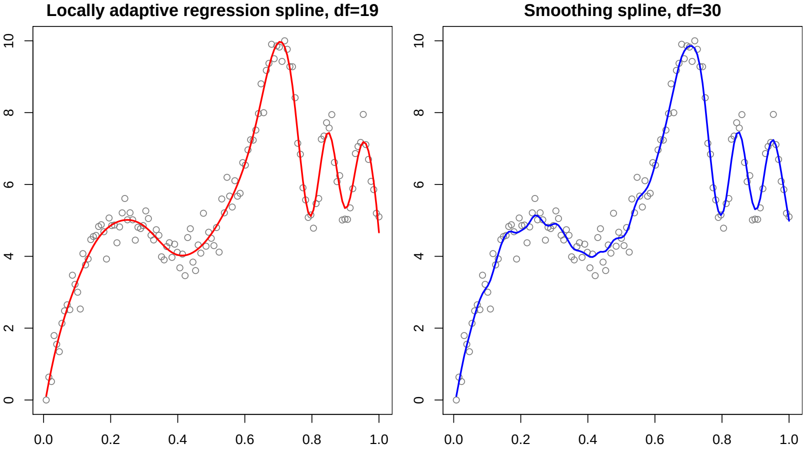


Example: comparing methods



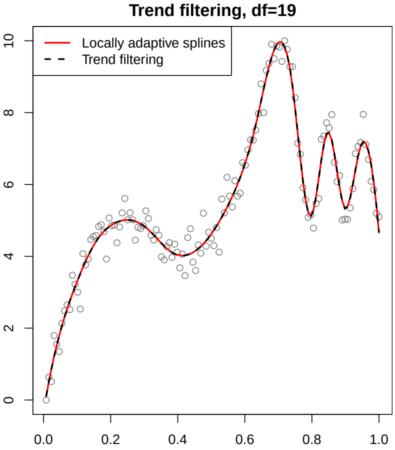


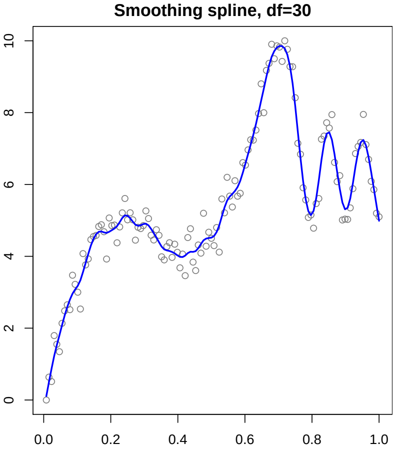
Example: comparing methods



Rates: $n^{-(2k+2)/(2k+3)}$ $n^{-(2k+1)/(2k+2)}$

Example: comparing methods





Rates: $n^{-(2k+2)/(2k+3)}$ (both) $\qquad$ $n^{-(2k+1)/(2k+2)}$ (any linear estimator)

Trend filtering

Trend filtering (Steidl et al. 2006, Kim et al. 2009, Tibshirani 2013) is a discrete approximation to locally adaptive regression splines:

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

Preserves asymptotic properties (e.g., minimax optimality), but is much faster computationally

Rough explanation: $\mathrm{TV}(f^{(k)}) \approx \int |f^{(k+1)}(t)|\,dt \approx \|D^{(k+1)}\beta\|_1$, where $D^{(k+1)}$ is a discrete derivative operator of order $k+1$, i.e.,

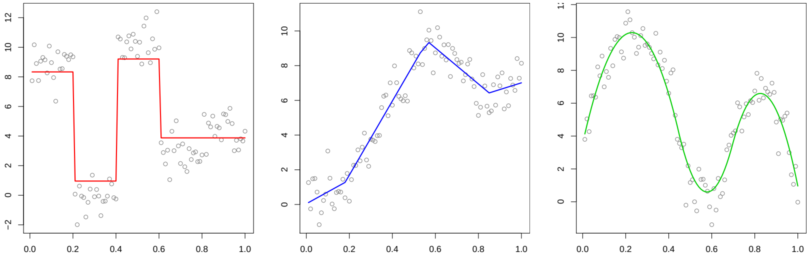
$$D^{(1)} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix},$$

$$D^{(k+1)} = D^{(1)} D^{(k)} \text{ for } k = 1, 2, 3, \dots$$

Fast computation stems from bandedness of these operators

Trend filtering in continuous space

Intuitively, trend filtering solution $\hat{\beta}$ should exhibit the structure of kth degree piecewise polynomial (since it penalizes changes in kth derivatives across inputs)



Constant, $k = 0$
(Fused lasso)

Linear, $k = 1$

Quadratic, $k = 2$

This idea can be formalized using falling factorial functions
(W., Smola, Tibshirani. 2014)

The falling factorial basis

Trend filtering: $\min_{\beta \in \mathbb{R}^n} \|y - \beta\|_2^2 + \lambda \cdot \frac{1}{k!} \|D^{(k+1)} \beta\|_1$

Reformulation: $\min_{\alpha \in \mathbb{R}^n} \|y - H^{(k)} \alpha\|_2^2 + \lambda \sum_{j=k+2}^{n} |\alpha_j|$

Continuous space embedding:

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \mathrm{TV}(f^{(k)})$$

Locally adaptive regression splines:

$$\min_{f \in \mathcal{G}_k} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \mathrm{TV}(f^{(k)})$$

The falling factorial basis

$\mathcal{G}_k$ is spanned by:

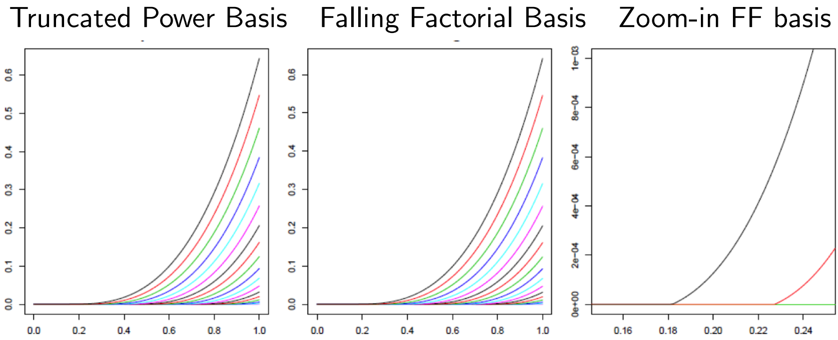
$$g_1(x) = 1, \quad g_2(x) = x, \quad ..., \quad g_k(x) = x^k$$
$$g_{k+1}(x) = (x - t_1)_+^k, \quad ..., \quad g_n(x) = (x - t_{n-k+1})_+^k$$

$\mathcal{H}_k$ is spanned by:

$$h_1(x) = 1, \quad h_2(x) = x - t_1, \quad ..., \quad h_k(x) = \prod_{\ell=1}^{k}(x - t_\ell)$$
$$h_{k+1}(x) = \prod_{\ell=2}^{k+1}(x - t_\ell), \quad ..., \quad h_n(x) = \prod_{\ell=n-k+1}^{n}(x - t_\ell)_+^k$$

Essentially replacing power functions m^k with falling factorial function $m(m-1)...(m-k+1)$.

The falling factorial basis



- Not the same, but close enough!

The falling factorial basis

What is the advantage?

- Same statistical optimality.
- but way faster $O(n^2) \to O(n)$!
- Faster than FFT, Wavelet!

Challenge:

- Other applications?
- Higher order Kolmogorov-Smirnov Test.
- Some use in signal/image processing?
- Generalize to estimate multivariate functions?

2 Graph trend filtering

(W., Sharpnack, Smola, Tibshirani, 2015 AIStats+JMLR)

Nonparametric regression on graphs

Graph smoothing: given a graph $G = (V, E)$, with vertices denoted $V = \{1, \ldots n\}$, we observe

$$y_i = \mu_i + \epsilon_i, \quad i = 1, \ldots n$$

Errors ϵ_i assumed to have zero mean. Want to estimate underlying signal μ, assumed to be smooth with respect to edges E

In comparison to univariate case, a lot less literature. E.g.,

- Eigen-based methods
- Laplacian smoothing
- Wavelets on graphs

Newcomer in this field: graph trend filtering, an extension of the univariate technique with analogous benefits

Graph trend filtering

Graph trend filtering (W., Sharpnack, Smola, Tibshirani, 2015) solves

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

where $\Delta^{(k+1)}$ is a graph difference operator of order $k+1$, over G

Two key properties of univariate trend filtering:

- Computationally fast
- Locally adaptive

With suitably defined difference operators $\Delta^{(k+1)}$, $k = 1, 2, 3, \ldots$, graph trend filtering will share these properties

Discrete differences over graphs

Given graph $G = (V, E)$ with $V = \{1, \ldots n\}$ and $E = \{e_1, \ldots e_m\}$

- Define the first order graph difference operator $\Delta^{(1)}$ to be the edge incidence matrix of G, an $m \times n$ matrix, whose ℓth row is

$$D_\ell = (0, \ldots \underset{\substack{\uparrow \\ i}}{-1}, \ldots \underset{\substack{\uparrow \\ j}}{1}, \ldots 0)$$

  if the ℓth edge is $e_\ell = \{i, j\}$

- For higher orders, use the recursion:

$$\Delta^{(k+1)} = \begin{cases} (\Delta^{(1)})^T \Delta^{(k)} & \text{for } k \text{ odd,} \\ \Delta^{(1)} \Delta^{(k)} & \text{for } k \text{ even} \end{cases}$$

I.e., for D the edge incidence matrix, and $L = D^T D$ the Laplacian:

$$\Delta^{(1)} = D, \;\; \Delta^{(2)} = L, \;\; \Delta^{(3)} = DL, \;\; \Delta^{(4)} = L^2, \;\; \ldots$$

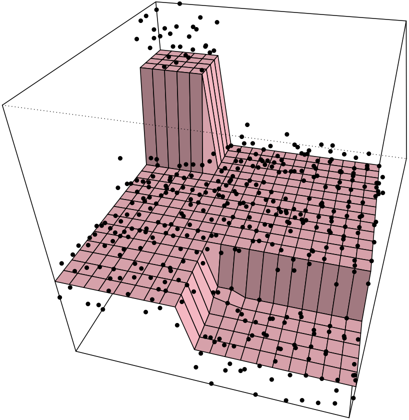
Constant order

The penalty for constant order graph trend filtering:

$$\|\Delta^{(1)}\beta\|_1 = \|D\beta\|_1 = \sum_{\{i,j\}\in E} |\beta_i - \beta_j|$$

Estimate $\hat{\beta}$ is piecewise constant over G

(This is also known as the graph fused lasso)

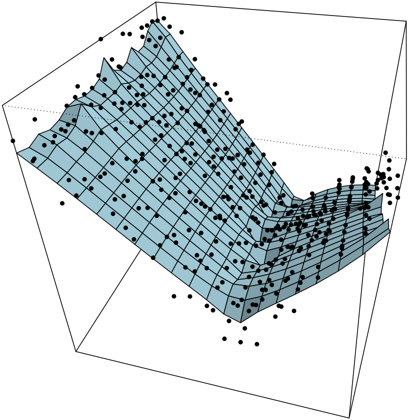


Linear order

The penalty for linear order graph trend filtering:

$$\|\Delta^{(2)}\beta\|_1 = \|L\beta\|_1 = \sum_{i=1}^{n} n_i \left| \beta_i - \frac{1}{n_i} \sum_{\{i,j\}\in E} \beta_i \right|$$

Estimate $\hat{\beta}$ is "piecewise linear" over G

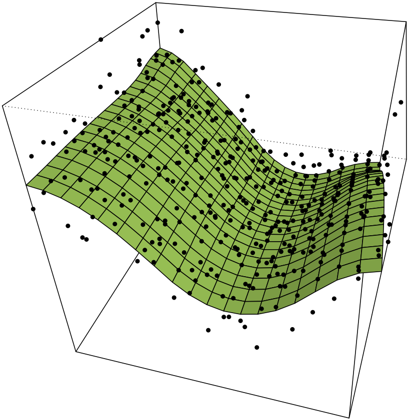


Quadratic order

The penalty for quadratic order graph trend filtering:

$$\|\Delta^{(2)}\beta\|_1 = \|DL\beta\|_1 = \sum_{\{i,j\}\in E} \left| \Big(n_i\beta_i - \sum_{\{i,\ell\}\in E} \beta_\ell\Big) - \Big(n_j\beta_j - \sum_{\{j,\ell\}\in E} \beta_\ell\Big) \right|$$

Estimate $\hat{\beta}$ is "piecewise quadratic" over G



Discrete differences over graphs

To sum up:

- For odd k, the $(k+1)$st order differences are given by taking first differences of kth differences:

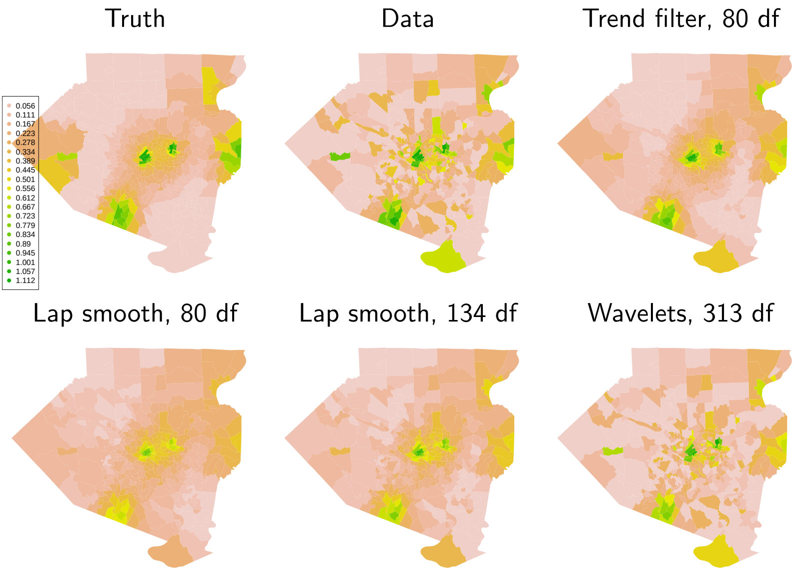
$$\Delta^{(k+1)} = D\Delta^{(k)}$$

- For even k, the $(k+1)$st order differences are given by taking second differences of $(k-1)$st order differences

$$\Delta^{(k+1)} = L\Delta^{(k-1)}$$

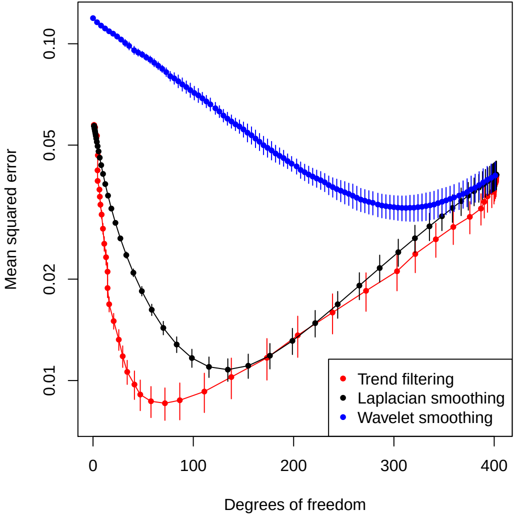
For the chain graph, with edges $E = \{\{i, i+1\} : i = 1, \dots n\}$, this construction exactly gives the difference operators in the univariate case (modulo boundary terms)

Example: comparing methods



Example: comparing methods

Mean squared errors (averaged over 10 simulations):



Examples of extensions

Logistic/Poisson Graph Trend Filtering:

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\arg\min}\ f(y, \beta) + \lambda_1 \|\Delta^{(k+1)} \beta\|_1, \tag{1}$$

Sparse Graph Trend filtering:

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\arg\min}\ \frac{1}{2}\|y - \beta\|_2^2 + \lambda_1 \|\Delta^{(k+1)} \beta\|_1 + \lambda_2 \|\beta\|_1, \tag{2}$$

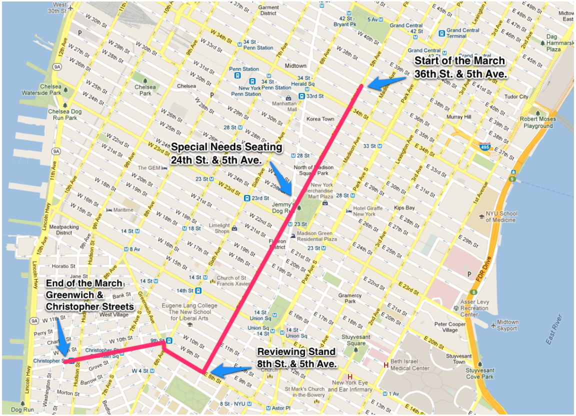
Graph Trend Completion (interpolation):

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\arg\min}\ \frac{1}{2}\|w \circ (y - \beta)\|_2^2 + \lambda \|\Delta^{(k+1)} \beta\|_1 \tag{3}$$

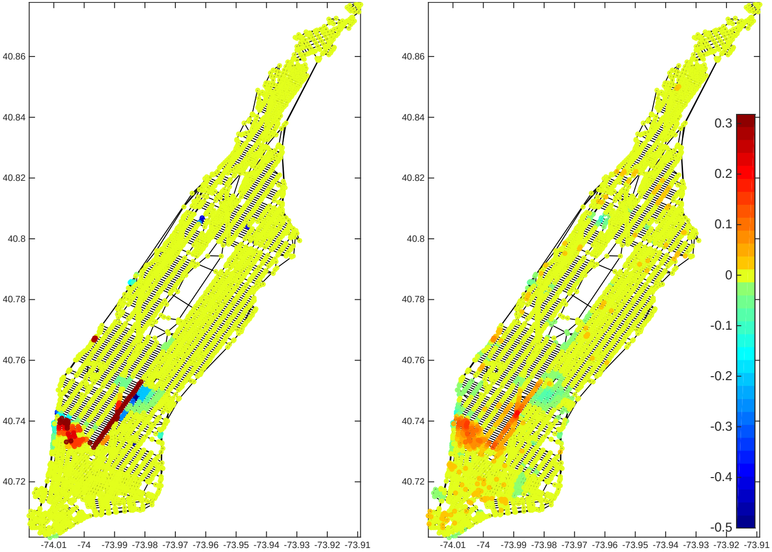
Event detection based on New York City Taxi counts

NYC 2013 Gay Pride Parade

Event detection based on New York City Taxi counts



Event detection on New York City Taxi counts

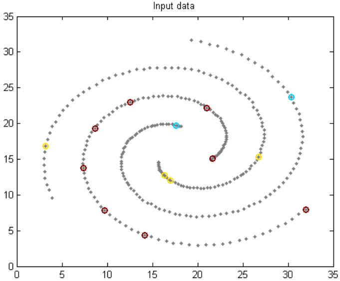


Sparse trend filtering

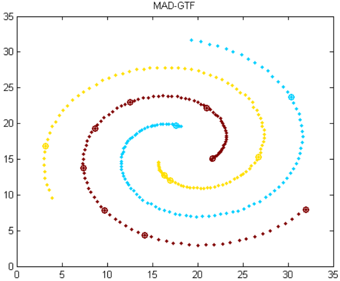
Sparse Laplacian smoothing

Graph-based Transductive Learning

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| {\color{red} w} \circ (y - \beta) \|_2^2 + \lambda \| \Delta^{(k+1)} \beta \|_1$$



Input: partially labelled data



Output: full labels

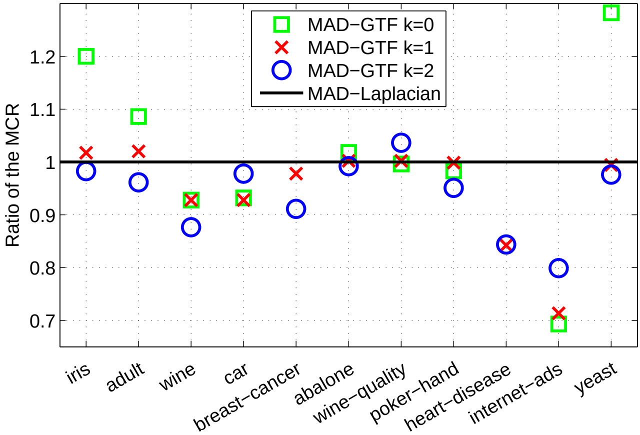
Graph-based Transductive Learning

Examples of this in Interactive Image Segmentation (Li. et. al., 2008)

.

Graph-based Transductive Learning on UCI Datasets

We apply to plain classification problems:



.

Theory

Assume $y \sim \beta_0 + \mathcal{N}(0, I)$, $\|\Delta\beta\|_1$ is small.

How well can we estimate β_0 by solving a generalized lasso problems:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2}\|y - \beta\|_2^2 + \lambda\|\Delta\beta\|_1, \tag{4}$$

Theory

Assume $y \sim \beta_0 + \mathcal{N}(0, I)$, $\|\Delta\beta\|_1 = O(1)$.

How well can we estimate β_0 by solving a generalized lasso problems:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2}\|y - \beta\|_2^2 + \lambda\|\Delta\beta\|_1, \tag{5}$$

Three general recipes in our paper:

- Basic error bound (using $\|\Delta^+\|_{2,\infty}$)
- Strong error bound 1 (using incoherence)
- Strong error bound 2 (using entropy)

Challenges in Theory

Hard to specialize to different graph structures. Case study:

- Specialize to a chain graph, minimax rate is $O(n^{-\frac{2k+2}{2k+3}})$

  - Basic error bound: Suboptimal $O(n^{-1/2})$.

Challenges in Theory

- Strong error bound 1: **Minimax rate!**
  Proof: k-D grids are constant incoherent,
- Strong error bound 2: **Minimax rate!**
  Proof: Manually construct a ϵ-cover set.

Other graphs? A lot of open questions.

Successes and challenges

Successes:

- As defined, the graph difference operators are structured (Laplacian-based) and permit efficient computation
- Empirical examples show superiority of graph trend filtering over other linear estimators like Laplacian smoothing

Challenges:

- Theoretically, we have a few general recipes for proving estimation bound. Not sure how sharp these bounds are except that it attains minimax rate for the chain graph
- Continuous space interpretations are difficult. is there a set of basis functions for each graph?
- Multidimensional (Euclidean) trend filtering is an open topic in general

How to solve the Trend filtering problem?

A clever ADMM decomposition (Ramdas & Tibshirani,2014):

$$\min_{\beta\in\mathbb{R}^n} \frac{1}{2}\|y-\beta\|_2^2 + \lambda\|D^{(1)}\alpha\|_1$$
$$\text{s.t } \alpha = D^{(k)}\beta$$

Solve subroutine using dynamic programming.
GLM loss via Prox. Newton.

glmgen software package in R and C on github!
Falling factorial basis operations in C with Matlab interface on my homepage.

Efficient GTF implementation to come soon!

Challenges in GTF computation

Solve linear systems:

$$(\lambda L^k + I)x = b$$

- This is SDD when $k = 1$, not SDD for $k \geq 2$.
- Fast algorithm exists for grids. Fast DCT.
- No clue in general.

Summary

Takeaway points:

- Trend filtering methods are computationally fast and locally adaptive
- The regularization scheme is also transparent (easy to extend, easy to adapt)
- Many challenges remain (e.g., conducting proper inference)
- But there are several promising leads as well

Acknowledgements

Ryan Tibshirani

James Sharpnack

Alex Smola

Veeru Sadhanala

Thank you for listening!

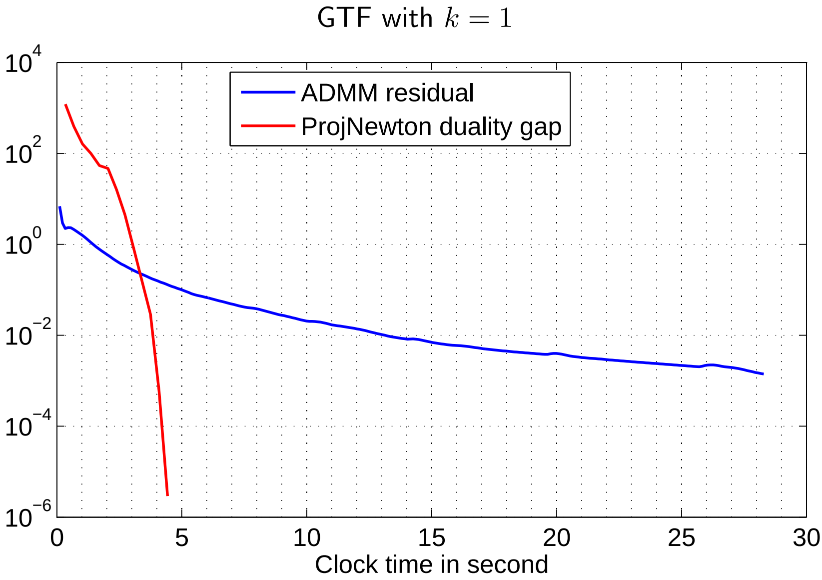
GTF computation

- $k = 0$ Solving Graph Fused Lasso:
  Parametric max flow (Chambolle et. al., 2011)
- $k = 1$ Projected Newton on the dual with SDD solver.
- $k = 2$ Special ADMM with Chambolle's solver as prox.

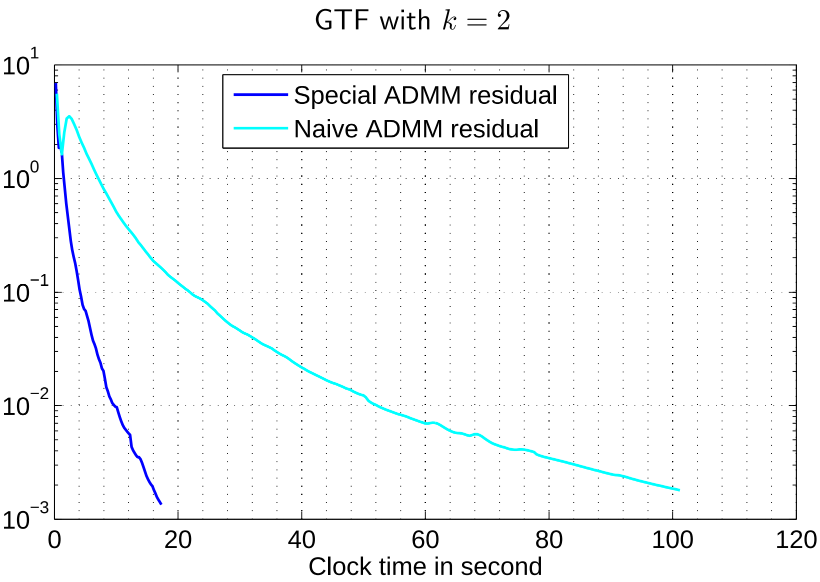
Fast computation

Computational experiments on TV denoising.

ADMM vs. Projected Newton

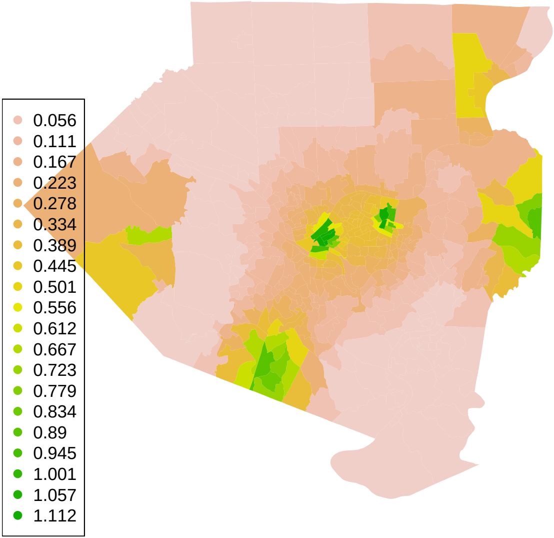


Naive ADMM vs. Special ADMM



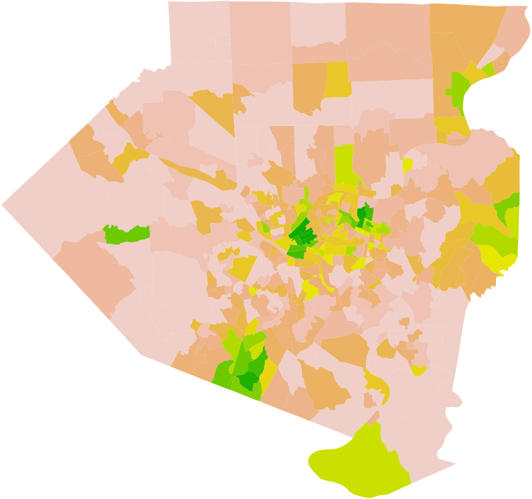
Example: comparing methods

Real graph, from Allegheny County (Pittsburgh). Simulated signal:



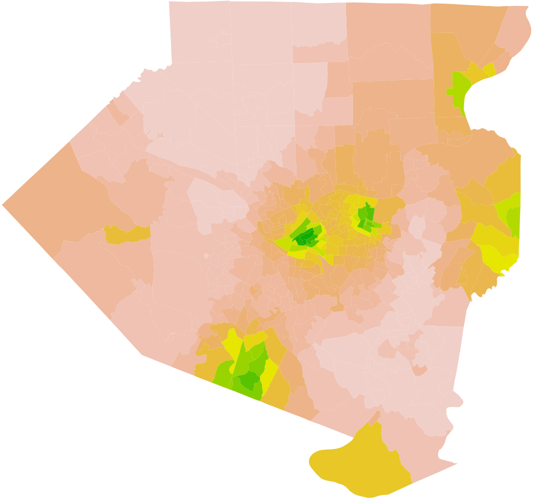
Example: comparing methods

Noisy realization:



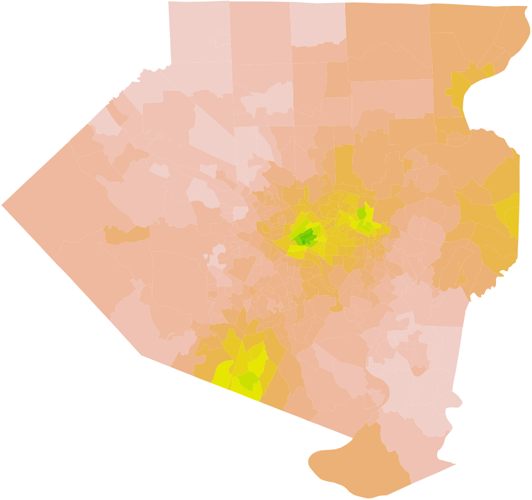
Example: comparing methods

Quadratic graph trend filtering, 80 df:



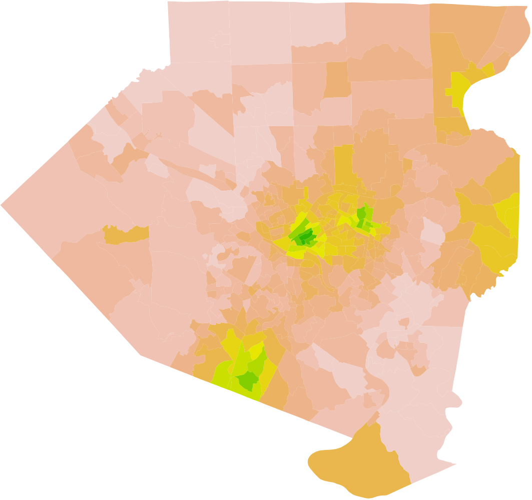
Example: comparing methods

Laplacian smoothing, 80 df:



Example: comparing methods

Laplacian smoothing, 134 df:



Example: comparing methods

Wavelet smoothing, 313 df: