Efficient Implementations of the Generalized Lasso Dual Path Algorithm

or, How to Efficiently Solve Certain Sequences of Structured Linear Systems

Taylor Arnold and Ryan Tibshirani (me)
AT&T Labs Research and Carnegie Mellon University
Sparse linear regression

Given $y \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n \times p}$, suppose that we want to fit a linear model of $y$ on $X$ with few nonzero coefficients. One way: solve the lasso optimization problem (Tibshirani, 1996; Chen et al., 1998)

$$\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

The regularization parameter $\lambda$ controls the tradeoff between goodness-of-fit (low bias) and sparsity (low variance)

*There are so many ways to perform variable selection in the linear model. Why use the lasso?*

- Computational: it can be computed efficiently
- Theoretical: under some (idealized) conditions, can prove that it has favorable estimation and recovery properties
Structured linear regression

Given \( y \in \mathbb{R}^n \), \( X \in \mathbb{R}^{n \times p} \), and \( D \in \mathbb{R}^{m \times p} \), we consider:

\[
\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \| y - X \beta \|_2^2 + \lambda \| D \beta \|_1
\]

called the generalized lasso

Now \( D \beta \) is encouraged to be sparse ... the idea is to choose \( D \) so that this translates to some other desired behavior for \( \beta \)

\textit{Wait a minute, can’t we change variables to } \( \alpha = D^{-1} \beta \), \textit{and then the above is just a regular lasso?}

\begin{itemize}
  \item Possible if \( D \) is invertible, or if \( \text{rank}(D) = m \)
  \item Not possible if \( \text{rank}(D) < m \). In this case, the generalized lasso is a richer framework
\end{itemize}
What are path algorithms?

Our goal is to compute the solution path, \( \hat{\beta}_\lambda \) for \( \lambda \in [0, \infty] \), of the generalized lasso problem. Some history:

- The idea has been around for a while (sometimes referred to as homotopy algorithms in optimization)
- The LARS algorithm (Efron et al., 2004) does this for the lasso problem, i.e., for \( D = I \)
- LARS inspired the development of path algorithms in other settings, e.g., paths for SVM (Hastie et al., 2004), and Dantzig selector (James et al., 2008)

This is possible because the solution paths are piecewise linear in \( \lambda \) (see Rosset and Zhu, 2007)

A path algorithm:
- Delivers the exact solution, exhaustively
- Reveals new possibilities for statistical understanding
Visualization of the path

Can choose $D$ to yield a piecewise quadratic fit

$\hat{\beta}_\lambda$ for $\lambda = 25$
Visualization of the path

Can choose $D$ to yield a piecewise quadratic fit

$\hat{\beta}_\lambda$ for $\lambda = 25$

$\hat{\beta}_\lambda$ for $\lambda \in [0, \infty]$
Visualization of the path

$\lambda = 7496.679$
Snapshots of the path

Can choose $D$ to do spatial denoising: burglaries per household in 2005–2009 in Chicago, over $n = 2167$ census tracts

$\lambda = 0.101, 2 \, df$
Snapshots of the path

Can choose $D$ to do spatial denoising: burglaries per household in 2005–2009 in Chicago, over $n = 2167$ census tracts

$\lambda = 0.059, 5 \, df$
Snapshots of the path

Can choose $D$ to do spatial denoising: burglaries per household in 2005–2009 in Chicago, over $n = 2167$ census tracts

$\lambda = 0.037$, 8 df
Snapshots of the path

Can choose $D$ to do spatial denoising: burglaries per household in 2005–2009 in Chicago, over $n = 2167$ census tracts

$\lambda = 0.025, 10 \, df$
Snapshots of the path

Can choose $D$ to do spatial denoising: burglaries per household in 2005–2009 in Chicago, over $n = 2167$ census tracts

$\lambda = 0.019, 15\ df$
Snapshots of the path

Can choose $D$ to do spatial denoising: burglaries per household in 2005–2009 in Chicago, over $n = 2167$ census tracts

$\lambda = 0.016, 25\ df$
Signal approximation problems

Consider first $X = I$, the signal approximation case:

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

Typically:

- The coordinates of $y \in \mathbb{R}^n$ obey some underlying structure
- The rows of $D \in \mathbb{R}^{m \times n}$ are chosen to reflect this structure

It helps to see some examples. For the sake of time, focus on two problem classes: fused lasso and trend filtering
Fused lasso

\[ \hat{\beta}_\lambda = \arg \min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|_2^2 + \lambda \| D \beta \|_1 \]

Let \( D = \begin{bmatrix} -1 & 1 & 0 & 0 & \ldots \\ 0 & -1 & 1 & 0 & \ldots \\ 0 & 0 & -1 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \). This is the 1d fused lasso
Fused lasso

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|_2^2 + \lambda \| D \beta \|_1 \]

Suppose \( D \) gives “adjacent” differences in \( \beta \):

\[ D_i = (0, 0, \ldots - 1, \ldots, 1, \ldots 0), \]

where adjacency is defined according to a graph \( G \). For a 2d grid, this is the 2d fused lasso
Fused lasso

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|_2^2 + \lambda \| D \beta \|_1 \]

Suppose \( D \) gives “adjacent” differences in \( \beta \):

\[ D_i = (0, 0, \ldots - 1, \ldots, 1, \ldots 0), \]

where adjacency is defined according to a graph \( G \). Can be applied to an arbitrary graph structure.
Fused lasso

\[
\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|_2^2 + \lambda \| D\beta \|_1
\]

Graph derived from spatially adjacency in US states
Fused lasso

\[
\hat{\beta}_\lambda = \arg \min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|^2_2 + \lambda \| D \beta \|^1_1
\]

Graph derived from spatially adjacency in US states
Trend filtering

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|_2^2 + \lambda \| D \beta \|_1 \]

Let \( D = \begin{bmatrix} -1 & 2 & -1 & 0 & \ldots \\ 0 & -1 & 2 & -1 & \ldots \\ 0 & 0 & -1 & 2 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \). This is linear trend filtering
Trend filtering

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|_2^2 + \lambda \| D\beta \|_1 \]

Let \( D = \begin{bmatrix} -1 & 3 & -3 & 1 & \ldots \\ 0 & -1 & 3 & -3 & \ldots \\ 0 & 0 & -1 & 3 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \end{bmatrix} \). Get quadratic trend filtering.
Trend filtering

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \| y - \beta \|_2^2 + \lambda \| D \beta \|_1 \]

Let \( D_{ij} = (-1)^{j-i} \binom{k+1}{j-i} \), for \( j = i, \ldots i + k + 1 \), and \( D_{ij} = 0 \) otherwise. This is polynomial trend filtering of order \( k \).
Regression problems

Now with a general regression matrix $X$:

$$
\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \| y - X \beta \|_2^2 + \lambda \| D \beta \|_1
$$

Interesting examples involve $X$ that is somehow in sync with the underlying structure. For the sake of time, we’ll only look at one such example.
Back to the fused lasso

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \| y - X \beta \|^2_2 + \lambda \| D \beta \|_1 \]

Consider an MRI study:

- **Response** \( y = \) Alzheimer’s score

- Each row of \( X \) is an MRI image, \( X_i = \)

- Penalty matrix \( D = \begin{bmatrix} D_{3d} \\ I \end{bmatrix} \)

- Solution \( \hat{\beta}_\lambda \) also has the structure of a 3d image, and uses **sparse, contiguous** regions of the brain to explain Alzheimer’s
Back to the fused lasso

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|D\beta\|_1 \]

Figure 4: Consistency of selected voxels in different trials of cross-validations. The results of 5 different folds of cross-validations are shown in (a)-(e) and the overlapping voxels in all 10 folds are shown in (f). The top row shows the results for GFL and the bottom row shows the results for $L_1$. The percentages of the overlapping voxels were: GFL(66%) vs. $L_1$ (22%).

Table 1: Comparison of the accuracy of AD classification.

<table>
<thead>
<tr>
<th>Task</th>
<th>LR SVM</th>
<th>LR+</th>
<th>LR+GFL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD/NC</td>
<td>80.45%</td>
<td>82.71%</td>
<td>81.20%</td>
</tr>
<tr>
<td>MCI</td>
<td>63.83%</td>
<td>67.38%</td>
<td>68.79%</td>
</tr>
</tbody>
</table>

We compared GFL to logistic regression (LR), support vector machine (SVM), and logistic regression with an $L_1$ regularizer. The classification accuracies obtained based on a 10-fold cross validation (CV) are shown in Table 1, which shows that GFL yields the highest accuracy in both tasks. Furthermore, compared with other reported results, our performance are comparable with the state-of-the-art. In (Cheng, Zhang, and Shen 2012), the best performance with MCI tasks is 69.4% but our method reached 70.92%. In (Chu et al. 2012), a similar sample size is used as in our experiments, the performance of our method with AD/NC tasks is comparable to or better than their reported results (84.21% vs. 81-84%) whereas our performance with MCI tasks is much better (70.92% vs. 65%).

We applied GFL to all the samples where the optimal parameter settings were determined by cross-validation. Figure 5 compares the selected voxels with non-structured sparsity (i.e. $L_1$), which shows that the voxels selected by GFL clustered into several spatially connected regions, whereas the voxels selected by $L_1$ were more scattered. We considered the voxels that corresponded to the top 50 negative $i$’s as the most atrophied voxels and projected them onto a slice. The results show that the voxels selected by GFL were concentrated in hippocampus, parahippocampal gyrus, which are believed to be the regions with early damage that are associated with AD. By contrast, $L_1$ selected either less critical voxels or noisy voxels, which were not in the regions with early damage (see Figure 5(b) and 5(c) for details).

The voxels selected by GFL were also much more consistent than those selected by $L_1$, where the percentages of overlapping voxels according to the 10-fold cross-validation were: (a) GFL = 66% vs. $L_1$ = 22%, as shown in Figure 4.

Conclusions

In this study, we proposed an efficient and scalable algorithm for GFL. We demonstrated that the proposed algorithm performs significantly better than existing algorithms. By exploiting the efficiency and scalability of the proposed algorithm, we formulated the diagnosis of AD as GFL. Our evaluations showed that GFL delivered state-of-the-art classification accuracy and the selected critical voxels were well structured.
Path algorithm and the dual problem

T. and Taylor (2011) constructed a path algorithm that solves the generalized lasso for any $X, D$. This is like LARS, but extended to a larger class of problems.

The algorithm operates on the generalized lasso dual problem. For simplicity, assume that $X = I$ (signal approximator case). Primal problem:

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

Dual problem:

$$
\hat{u}_\lambda = \arg\min_{u \in \mathbb{R}^m} \frac{1}{2} \|y - D^T u\|_2^2 \text{ subject to } \|u\|_\infty \leq \lambda
$$

The solutions satisfy the relationship: $\hat{\beta}_\lambda = y - D^T \hat{u}_\lambda$
Overview of the algorithm

Our problem: \[ \hat{u}_\lambda = \arg\min_{u \in \mathbb{R}^m} \frac{1}{2} \| y - D^T u \|_2^2 \text{ subject to } \| u \|_\infty \leq \lambda \]

- Start with \( \lambda = \infty \). This is simply least squares, and a solution is given by \( \hat{u}_\infty = (DD^T)^+ D^T y \)
- Let \( i = \arg\max_{j=1,...,m} |\hat{u}_\infty,j| \). Note we can decrease \( \lambda \) down to \( \lambda_1 = |\hat{u}_\infty,j| \) and \( \hat{u}_\infty \) will still be the solution. At \( \lambda_1 \), we say coordinate \( i \) is on the boundary
- Now for \( \lambda \leq \lambda_1 \), set \( \hat{u}_{\lambda,i} = s\lambda \) with \( s = \text{sign}(\hat{u}_\infty,i) \) (i.e., keep it on the boundary) and solve the reduced problem:
  \[ \hat{u}_{\lambda,-i} = \arg\min_{u_{-i} \in \mathbb{R}^m} \frac{1}{2} \| y - D_i^T s\lambda - D_i^T u_{-i} \|_2^2 \text{ subject to } \| u_{-i} \|_\infty \leq \lambda \]
  \[ = (D_{-i}D_{-i}^T)^+ D_{-i}(y - D_i^T s\lambda), \text{ simply least squares} \]
- Decrease \( \lambda \) until another coordinate hits the boundary, or until coordinate \( i \) leaves the boundary (whichever happens first)
Simple example
Simple example

Coordinates of $u$

Coordinates of $\beta$
Fundamental computational problem

Computations in the path algorithm essentially boil down to the following:

- Solve a linear system in $x$,
  \[ DD^T x = Db \]
- If this is system underdetermined, we require minimum $\ell_2$ norm solution
- Delete or add a row to $D$
- Repeat until $D$ has no rows (or max iterations reached)

Two key aspects: high relatedness of linear systems, and possible structure inherent to $D$
Summary of iteration complexities

At any given iteration, let $r$ be the number of interior coordinates in dual solution (not on the boundary). Iteration complexities:

<table>
<thead>
<tr>
<th></th>
<th>$X = I$</th>
<th>General $X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>General $D$, $\text{rank}(D) = m$</td>
<td>$O(rn)$</td>
<td>$O(rn)$</td>
</tr>
<tr>
<td>General $D$, $\text{rank}(D) &lt; m$</td>
<td>$O(\max{r^2, n^2})$</td>
<td>$O(\max{r^2, n^2})$</td>
</tr>
<tr>
<td>Trend filtering</td>
<td>$O(r)$</td>
<td>$O(r + nq^2)$</td>
</tr>
<tr>
<td>Fused lasso</td>
<td>$O(\max{r, n})$</td>
<td>$O(\max{r, n} + nq^2)$</td>
</tr>
</tbody>
</table>

For general $D$, our implementation maintains and updates matrix factorizations. We’ll focus on two special classes: trend filtering and fused lasso problems.

We’ll skip general $X$ case entirely, but interesting to point out that here we have $q \ll p$, and we avoid inverting full $X^T X$. 

In $k$th order trend filtering, $D$ begins as $(n - k - 1) \times n$, and it is banded with bandwidth $k + 2$. Hence

$$DD^T x = Db$$

is a banded system and solvable in $O(n)$ time. If rows are removed or added, this system is still banded, and takes $O(r)$ time.

Can use sparse QR to solve each banded system, and then update this decomposition after a row of $D$ is deleted or added.

<table>
<thead>
<tr>
<th></th>
<th>Dense</th>
<th>Sparse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute QR</td>
<td>$O(n^3)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Apply QR</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Update QR</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>
Fused lasso iterations

For fused lasso on a graph, the linear system

$$DD^T x = Db$$

is generically underdetermined. Thus we need minimum $\ell_2$ norm solution. Write this as

$$x = (DD^T)^+ D^T = D(D^T D)^+ b$$

That is, can compute $z = (D^T D)^+ b$, and then $x = Dz$. We have accomplished two things:

- Can actually find any solution of $D^T Dz = b$, as multiplication by $D$ afterwards will eliminate anything in null space
- Can utilize the fact that $L = D^T D$ is the Laplacian matrix of the underlying graph, which has key structure
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: \( L \) is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: \( L \) is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components.
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: \( L \) is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: \( L \) is block diagonal over connected components.
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: \( L \) is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
Illustration of fused lasso iterations

Key structure: $L$ is block diagonal over connected components
The sad truth

Path algorithms are not meant for very large problems. Roughly, can compute:

- General $D$ path for $n$ in the hundreds
- Trend filtering, fused lasso paths for $n$ in the thousands

Larger problems are intractable, but not because of iteration cost. We run into too many events along the path (iterations needed)

Pointwise (rather than pathwise) solvers, at fixed value of $\lambda$, scale much better. Powerful tools from convex optimization:

- First-order methods (proximal gradient, acceleration)
- Second-order methods (interior point, projected Newton)
- Splitting methods (ADMM, coordinate descent)

So what room does this leave for path algorithms?
Selective inference is a small but growing field, that studies formal inference procedures for hypotheses that were adaptively selected

E.g.,

- Sparse regression, test the significance of a variable that has been made active by the lasso
- Change-point detection, test the significance of a changepoint that has been fit by the 1d fused lasso

One particular line of work taken by Lockhart et al. (2013), Taylor et al. (2014) focuses on a detailed characterization of the selection procedure (lasso, 1d fused lasso), as provided by path algorithms
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[
\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}|
\]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[
\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}|
\]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[
\hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}|
\]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[
\hat{\beta}_\lambda = \arg \min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}|
\]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[ \hat{\beta}_\lambda = \arg \min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \]
Selective inference example

P-values for changepoints detected by the 1d fused lasso problem:

\[ \hat{\beta}_\lambda = \arg\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \]
The future of path algorithms?

What lies ahead for path algorithms?

On inference front:

- Running a path algorithm for inferential purposes is actually considerably more expensive
- Fundamental computational problem: express a set of linear inequalities $a_i^T x \geq 0, i = 1, 2, \ldots$ in minimally redundant form

On estimation front:

- Can we jumpstart a path algorithm at arbitrary value of $\lambda$?
- Can we skip some events (deletion events?) along the path, to quickly make our way through more iterations?
Acknowledgements

Taylor Arnold  Jonathan Taylor

Thank you for listening!