Lecture 22
The Generalized Lasso

07 December 2015

Taylor B. Arnold
Yale Statistics
STAT 312/612

Class Notes

- Midterm II - Due today
- Problem Set 7 - Available now, please hand in by the 16th

Motivation

- Today I am going to introduce the generalized lasso. Rather than just giving a theoretical description without any data-driven motivation, I have instead modified a talk I gave on pricing personal home insurance.
- At a high level, my primary focus was understanding what factors contribute to the relative risk of insuring a person, vehicle, house, boat, apartment, ect.
- Examples of covariates used to study risk include: a person's credit score, an automobile's price, and the method of heating used within a house.
- One particularly important aspect of risk in personal insurance is the location where the insured entity (home or automobile) is located.
- For instance, homes located in flood or fire zones have obvious, direct effects on their risk due to location.
- Location also plays an indirect role on risk, as it is highly correlated with other variables such as income, age, occupation, and vehicle usage, many of which are hard to reliably collect and verify.

Insurance territory models

Given the special and important nature of location, we created territory models to describe the relative insurance risk due to location. Specifically, this is a process of mapping locations in a given area into a prediction of its riskiness (i.e., price).

Specific issues that arise in constructing territory models include:

- Dealing with data that are available at different degrees of granularity (state, city, zip, census block…), with the smaller levels providing more segmentation at the expense of noisier predictions. Furthermore, some data is only available at certain resolutions.
- Implementation concerns, regulatory restrictions, and the need for interpretability makes it difficult to use complex estimators like kriging.
- Determining whether to treat each region as a fixed effect (with its own intercept), or if prediction should be done primarily on the metadata attached to a given region.

Insurance territory models, cont.

- We use aggregated industry data, for each specific coverage specific (house fires, auto collision, …), to build our territory models.
- Unfortunately, due to licensing restrictions and the very competitive nature of personal insurance I am not able to present results from our industry or internal data.
- However, I do have access to a small set of data which almost perfectly resembles the industry data we use, for the home theft coverage within the city of Chicago.

Chicago Crime Data

- There exists an open data feed, which gives several basic pieces of information about individual reported crimes.[1]
- This includes a specific latitude and longitude of the incident, the date and time of the report, and primary categorization of the crime.
- Idea: Spatially join the burglary crimes to census block groups, and divide this number by the number of households in the block group. By modeling this, it should yield the relative territorial risk of insuring a home against burglary.

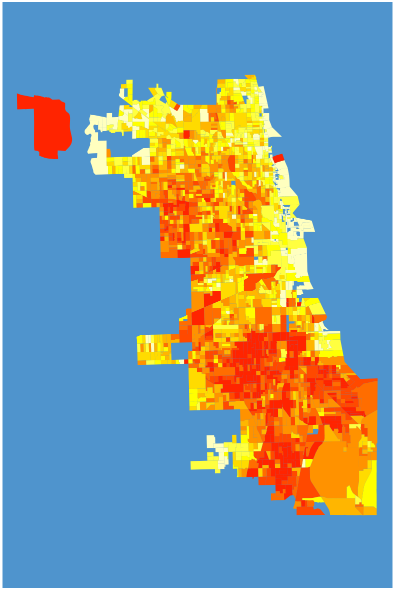
[1]https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present

Chicago Crime Data, Observed.

On the right, are the observed values of the response number of burglaries between 2012 divided by the number of households. Red is the highest percentage and white is the lowest percentage.

We clearly see spatial trends, but also notice that at this level of granularity, there is still quite a bit of noise.

Still, if we were interested purely in a machine learning type algorithm to predict the proportion of burglaries in the year 2013, simply using the observed in 2012 will be a very good guess.



Spatial Clustering: Concepts and goals

Rather than using the observed values, we need to cluster the regions together to form larger clumps of similar risk. This is necessary for the following reasons:

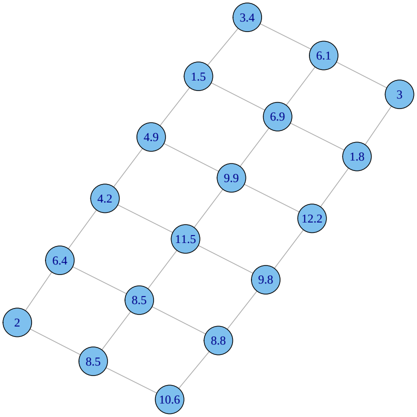
- If we use territory as a fixed effect (possibly even as an interaction), need to reduce the number of regions
- For business purposes and regulatory issues, need to have a model which is easier to understand and manage
- To avoid over-fitting to the mix of demographics in a region, since will not necessarily insure a homogeneous mix within a block group

In particular, we need a spatial clustering algorithm which is locally adaptive. That is, we want the algorithm to locally determine the number of clusters needed to fit the data 'well'.

Since we need adaptability, it makes sense to try to formalize a statistical model which will impose the form of sparsity we are trying to construct.

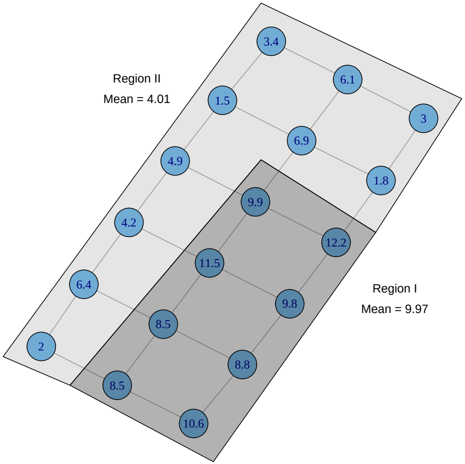
Theoretical model: Basic set-up

Consider the output of independent random variables observed on the nodes of a graph, such as:



Theoretical model: Basic set-up, cont.

If we have reason to believe that the true means of the nodes are piecewise constant along the edges, a reasonable partition of this graph is given by:



The fused lasso

The fused lasso is a statistical estimator for estimating this structure on arbitrary graphs, defined as the minimizer of the residual sum of squares, plus an ℓ_1 penalty term for edges with discordant mean predictions:

$$\widehat{\beta}_{fused} = \arg\min_{\beta} \frac{1}{2}||y - \beta||_2^2 + \lambda \cdot \sum_{(i,j) \in \text{edges}} |\beta_i - \beta_j| \tag{1}$$

As with the traditional lasso, the tuning parameter λ controls the degree of smoothness imposed on the estimator. We can re-write this condition compactly in terms of the of the oriented incidence matrix D; this matrix has a row for every edge and a column for every node, with elements ± 1 to indicate the nodes in an edge (each row has one positive and one negative entry, for our purposes the choice is arbitrary):

$$\widehat{\beta}_{fused} = \arg\min_{\beta} \frac{1}{2}||y - \beta||_2^2 + \lambda \cdot ||D\beta||_1 \tag{2}$$

Simple 1D Example

The simplest example of the fused lasso applies the algorithm to ordered data (corresponding to a chain graph). From this perspective, the fused lasso is essentially solving the classic change point problem, which has a wide range of applications.

Notice that this case can be written as:

$$||y - \beta||_2^2 + \lambda \cdot \sum_{i=1}^{n-1} ||\beta_i - \beta_{i+1}||_1$$

We can write this as:

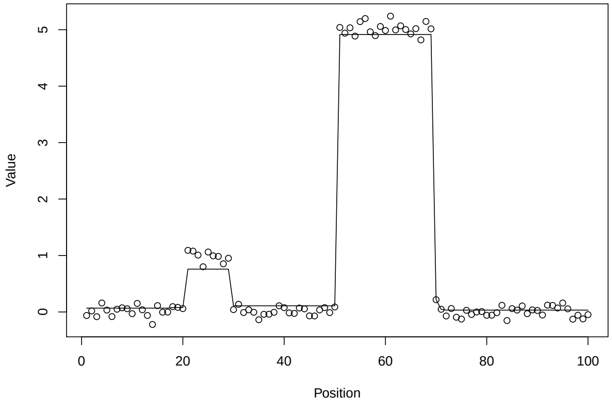
$$||y - \beta||_2^2 + \lambda \cdot ||D\beta||_1$$

For the aforementioned matrix D. In the case of $n = 5$, this looks like:

$$D = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

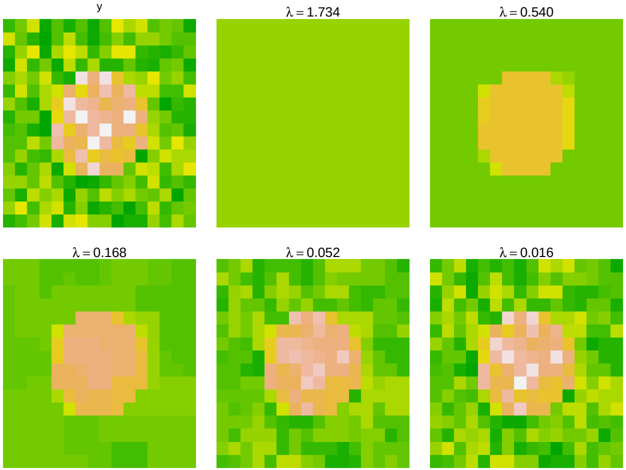
Simple 1D Example

Applying this to data gives a fit like the following:



Simple 2D Example

A small 2d is shown below (dark green is a low value, pink is a high value), for several values of the λ tuning parameter.



The Dual problem

The fused lasso optimisation problem is convex; therefore we can solve it by translating to the dual problem. Re-writing the fused lasso as:

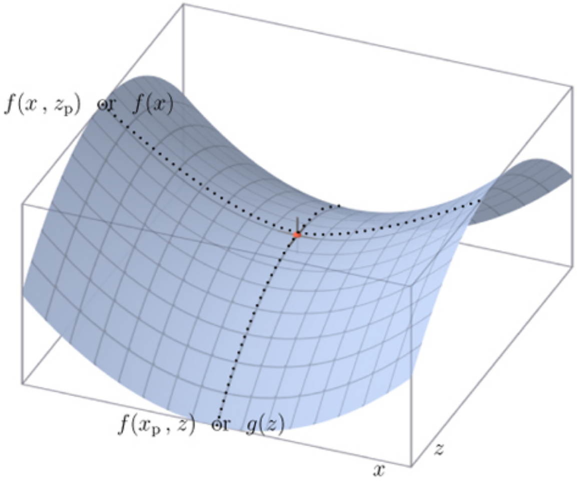
$$\arg\min_{\beta} \frac{1}{2}||y - \beta||_2^2 + \lambda \cdot ||z||_1, \text{ s.t. } z = D\beta \tag{3}$$

The Lagrangian becomes:

$$\begin{aligned} \Lambda(\beta, z, u) &= \frac{1}{2}||y - \beta||_2^2 + \lambda||z||_1 + u^t(D\beta - z) \qquad (4) \\ &= \frac{1}{2}||y - \beta||_2^2 + u^t D\beta + \lambda||z||_1 - u^t z \qquad (5) \end{aligned}$$

Graphical depiction of the Lagrangian

For a visual understanding of the dual problem we can visualize the Lagrangian solution as a saddle point[2]



[2]www.convexoptimization.com

The Dual problem, cont.

The dual function $g(u)$, is the infimum of Λ over β and z.

$$\Lambda(\beta, z, u) = \frac{1}{2}||y - \beta||_2^2 + u^t D\beta + \lambda||z||_1 - u^t z \tag{6}$$

Notice that if $||u||_\infty > \lambda$ then $g(u) = -\infty$. Otherwise, the infimum over z of $\lambda||z||_1 - u^t z$ is equal to zero.

The Dual problem, cont.

In the case when $||u||_\infty \leq \lambda$, the value of $g(u)$ depends only on the β infimum, given by:

$$\min_\beta \frac{1}{2}||y-\beta||_2^2 + u^t D\beta = -\frac{1}{2}||y - D^T u||_2^2 \tag{7}$$

Finally, the dual problem is to find the maximum of $g(u)$ over all u. We can wrap all of the conditions up nicely by stating the dual problem as:

$$\min_u \frac{1}{2}||y - D^T u||_2^2, \text{ s.t. } ||u||_\infty \leq \lambda \tag{8}$$

Given a solution to this, there is a corresponding solution to primal problem by the relationship $\widehat{\beta}_\lambda = y - D^T \widehat{u}_\lambda$.

Concept behind the dual fused lasso problem

- Using duality, we have converted an ℓ_1 constrained problem on the nodes of the graph to an ℓ_∞ constrained problem on the edges of the graph.
- For most graphs (connected, non-trees), this transformation has increased the dimensionality of the problem. For large values of λ, the solution space may have a very high dimensionality.
- Good news: There exists a solution to the dual problem where the coordinates of u are piecewise linear with respect to λ.

Analytic path solution

For $\lambda_0 = \infty$, the dual problem reduces to a least squares optimization. If D^t is not full column rank, there is not a unique least squares solution. However, we can define a canonical solution by using the Moore-Penrose pseudo-inverse:

$$\widehat{u}_{\lambda_0} = (DD^T)^+ Dy \tag{9}$$

In the path solution, the first 'break point' occurs at:

$$\lambda_1 = ||\widehat{u}_{\lambda_0}||_\infty \tag{10}$$

At this point, the coordinate of u on the boundary is set to $\pm\lambda$, and the other coordinates are determined by solving (9), with the corresponding row of D removed.

Computational notes

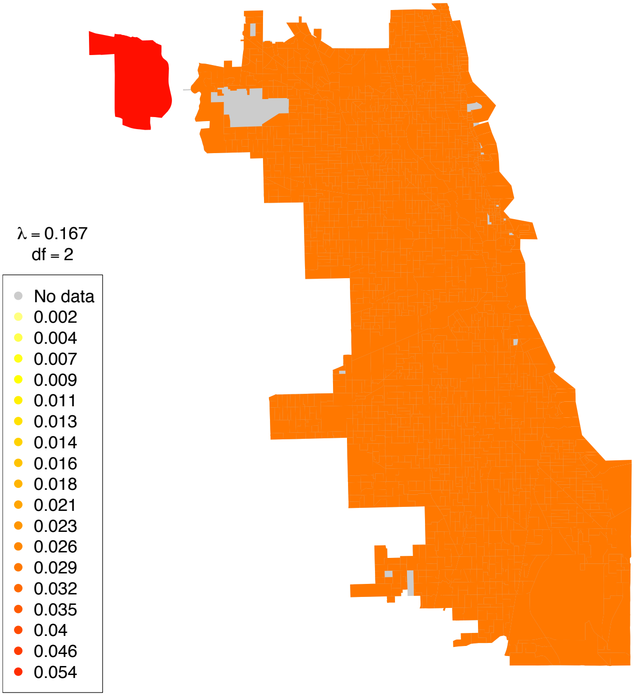
- In the classical lasso, the path starts from simplest model and works towards more complex ones. The dual solution is the opposite; we need to compute the full least squares solution before even starting the path.
- At the start of the path solution, the algorithm is only moving around in the null space of D^T. It is not until enough edges (corresponding the coordinates of u) sit on the λ boundary to disconnect the graph, that we see non-trival solutions in the primal problem.
- The least squares problem is being constantly solved, with one variable removed in each step. We can save the QR-decomposition of D^T, and utilize Given's rotations to drastically increase the speed of this process.
- For most graphs of interest, the matrix D will be sparse (and perhaps banded, for 1D problems). Taking advantage of sparse matrix libraries can increase the computational speed further.
- Once the graph in the primal problem becomes disconnected, the dual problem likewise become decoupled. Keeping track of the graph problems allows for efficiencies by solving the problem independently on each disconnected component.

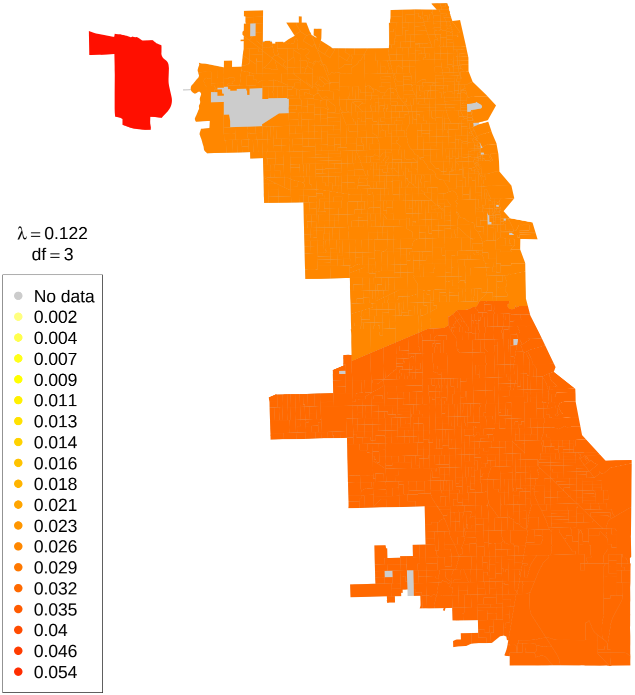
Implementation

The R package genlasso[3], available on CRAN, provides an implementation of this path algorithm. Some particularly notable features of the package:

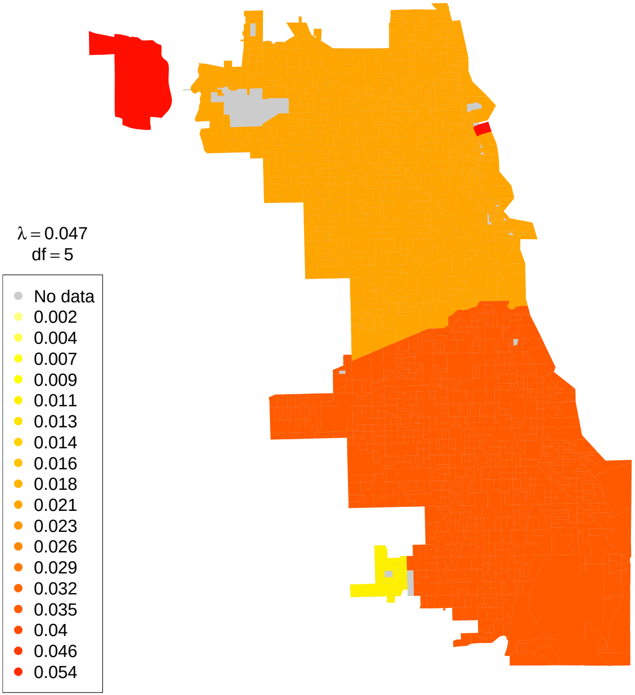
- Works for a generic D matrix, not just the fused lasso.
- Incorporates the computational points in the previous slide.
- Functionality for iterating an incomplete path, filebacking large problems, cross-validation techniques to tune the λ, and the ability to include a design matrix X.
- A standard set of S3 methods for plotting, summarizing, predicting, and extracting coefficients from a genlasso object.

[3]Ryan Tibshirani and Taylor Arnold (2014). genlasso: Path algorithm for generalized lasso problems. R package version 1.2.

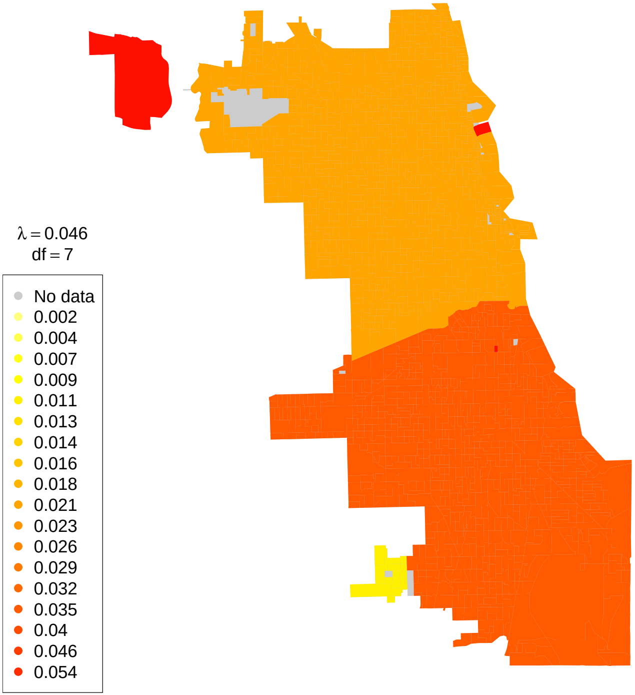

λ = 0.167
df = 2
No data
0.002
0.004
0.007
0.009
0.011
0.013
0.014
0.016
0.018
0.021
0.023
0.026
0.029
0.032
0.035
0.04
0.046
0.054

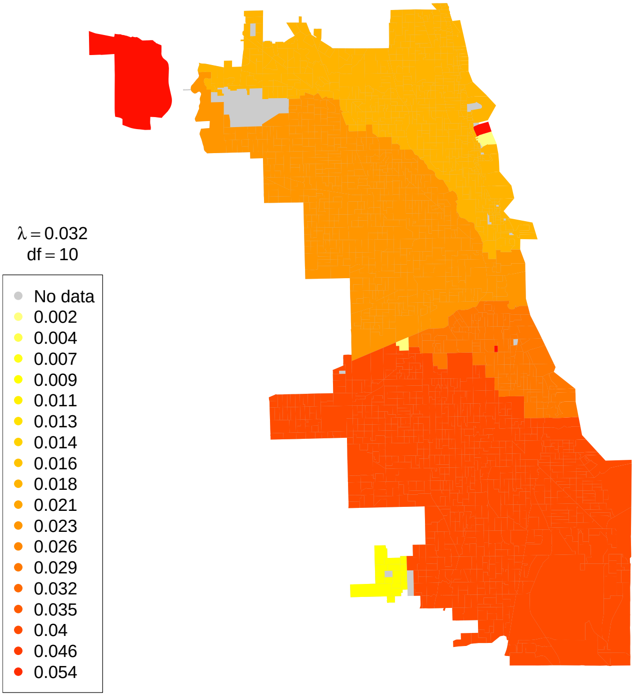


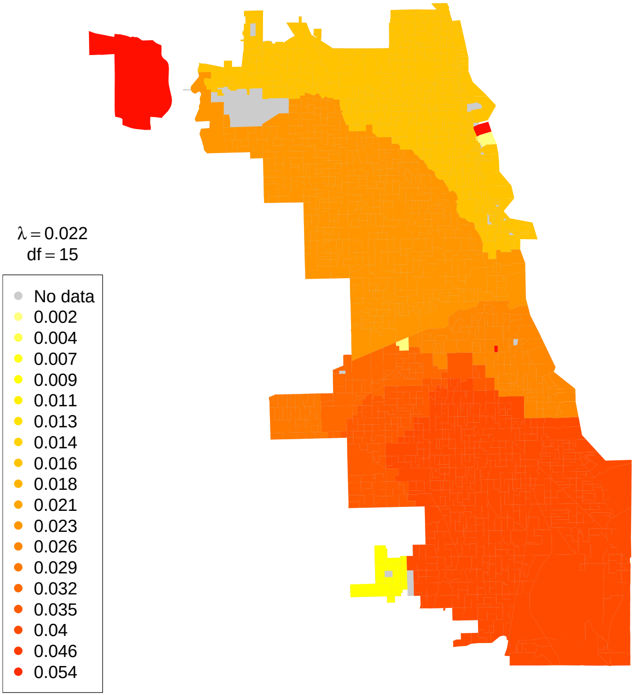
$\lambda = 0.122$
$df = 3$
No data
0.002
0.004
0.007
0.009
0.011
0.013
0.014
0.016
0.018
0.021
0.023
0.026
0.029
0.032
0.035
0.04
0.046
0.054



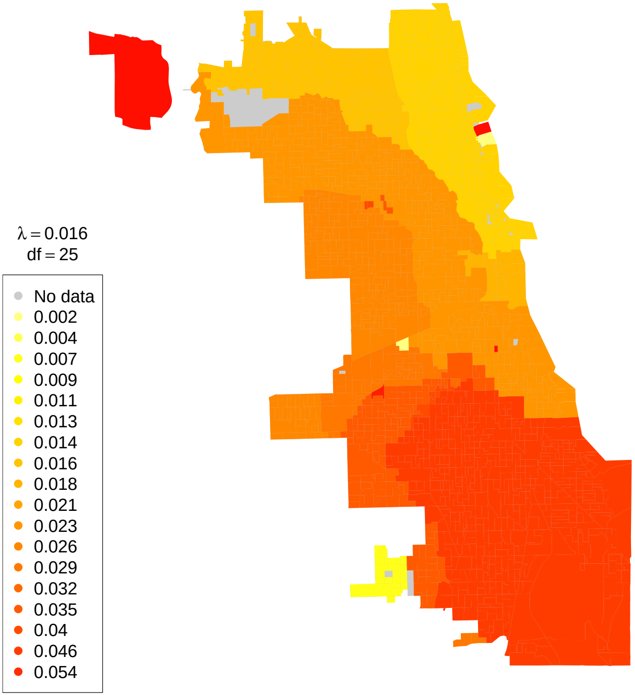
$\lambda = 0.047$
$df = 5$
No data
0.002
0.004
0.007
0.009
0.011
0.013
0.014
0.016
0.018
0.021
0.023
0.026
0.029
0.032
0.035
0.04
0.046
0.054


$\lambda = 0.046$
$\text{df} = 7$
No data
0.002
0.004
0.007
0.009
0.011
0.013
0.014
0.016
0.018
0.021
0.023
0.026
0.029
0.032
0.035
0.04
0.046
0.054


$\lambda = 0.032$
df = 10
No data
0.002
0.004
0.007
0.009
0.011
0.013
0.014
0.016
0.018
0.021
0.023
0.026
0.029
0.032
0.035
0.04
0.046
0.054



$\lambda = 0.022$
$\mathrm{df} = 15$
No data
0.002
0.004
0.007
0.009
0.011
0.013
0.014
0.016
0.018
0.021
0.023
0.026
0.029
0.032
0.035
0.04
0.046
0.054



$\lambda = 0.016$
$df = 25$
No data
0.002
0.004
0.007
0.009
0.011
0.013
0.014
0.016
0.018
0.021
0.023
0.026
0.029
0.032
0.035
0.04
0.046
0.054

Sparse fused lasso

A common variant of the fused lasso employs an additional ℓ_1 penalty on the coefficients themselves:

$$\widehat{\beta} = \underset{\beta \in \mathbb{R}^n}{\arg\min} \; \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{(i,j) \in E} |\beta_i - \beta_j| + \gamma \cdot \lambda \sum_{i=1}^{n} |\beta_i|.$$

Here $\gamma \geq 0$ is another parameter that controls the ratio between the fusion and sparsity penalty terms.

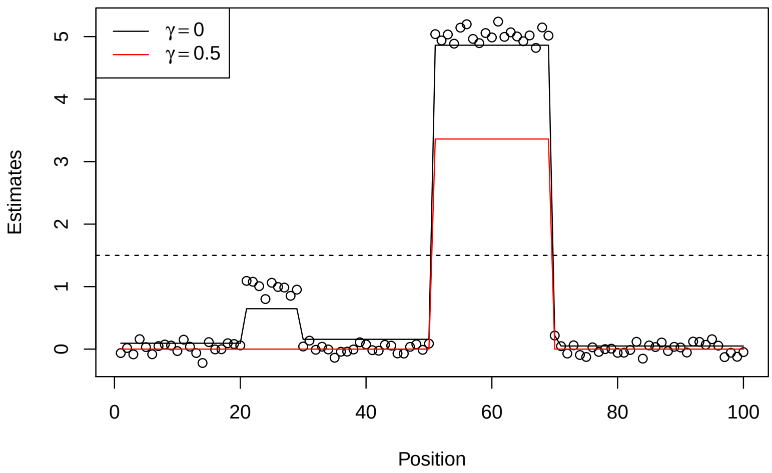
Note that this also fits into the generalized lasso framework, as it simply concatenates (a multiple of) the identity matrix to the rows of a fused lasso penalty matrix.

Take the following example:

```
> library(genlasso)
> set.seed(1)
> n = 100
> i = 1:n
> y = (i > 20 & i < 30) + 5*(i > 50 & i < 70) +
+   rnorm(n, sd=0.1)
> out = fusedlasso1d(y)
> beta1 = coef(out, lambda=1.5)$beta
```

To calculate the soft thresholded solution, we apply soft thresholding to the result:

```
beta2 = softthresh(out, lambda=1.5, gamma=1)
```



$\gamma = 0$
$\gamma = 0.5$
Estimates
Position
0
1
2
3
4
5
0
20
40
60
80
100

Trend filtering

Like the 1d fused lasso, trend filtering in the signal approximator case $X = I$ assumes that the data $y = (y_1, \dots y_n) \in \mathbb{R}^n$ is meaningfully ordered from 1 to n, and fits a piecewise polynomial of a specified degree. For example, linear trend filtering (with $X = I$) solves the following minimization problem:

$$\widehat{\beta} = \underset{\beta \in \mathbb{R}^n}{\arg\min} \ \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-2} |\beta_i - 2\beta_{i+1} + \beta_{i+1}|. \qquad (11)$$

Notice that here the discrete second derivative is penalized, as opposed to the discrete first derivative in the 1d fused lasso criterion. Quadratic and cubic trend filtering are defined similarly, by penalizing the discrete third and fourth derivative, respectively. (In this light, we can think of the 1d fused lasso as constant or zeroth order trend filtering.)

References

- Robert Tibshirani, et al. *Sparsity and smoothness via the fused lasso.* Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67: 91–108.
- Ryan Tibshirani and Taylor Arnold (2014). *genlasso: Path algorithm for generalized lasso problems.* R package version 1.2.
- S. Boyd et al., *Distributed optimization and statistical learning via the alternating direction method of multipliers.* Foundations and Trends in Machine Learning, 3(1):1–122, 2011.
- Ryan Tibshirani and J. Taylor (2011). ”The solution path of the generalized lasso”, Annals of Statistics 39 (3) 1335-1371.