Path algorithms

Geoff Gordon & Ryan Tibshirani
Optimization 10-725 / 36-725

Path algorithms

In this lecture we consider problems of the form

$$\min_{x \in \mathbb{R}^n} \; g(x) + \lambda \cdot h(x)$$

where $\lambda \geq 0$. In particular, we'll look at **path algorithms**, which deliver $\{x^\star(\lambda) : \lambda \in [0, \infty]\}$, called the solution path

Path algorithms start at one end (either $\lambda = 0$ or $\lambda = \infty$) where computing the solution is easy, and essentially trace the solution path by successively satisfying the KKT conditions

Properties:

- They deliver the **exact** solution (no iteration, no error bound guarantees) at all values of λ
- They provide useful platform for tuning parameter selection and statistical analysis

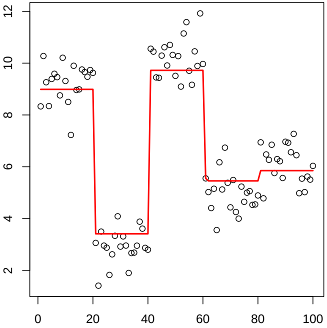
1d fused lasso (total variation denoising)

Given $y \in \mathbb{R}^n$, consider the **1d fused lasso** or **1d total variation denoising** problem:

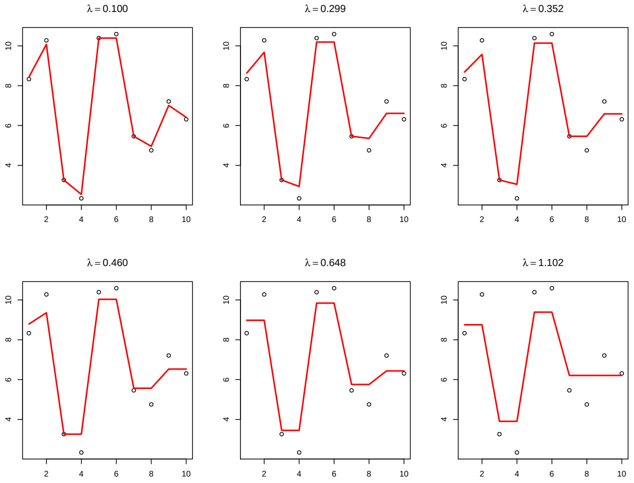
$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i)^2 + \lambda \sum_{i=1}^{n-1} |x_i - x_{i+1}|$$

Example: $n = 100$, plotted in red is solution $x^\star$ at $\lambda = 5$

Solution is piecewise constant with adaptively chosen break points; larger λ, fewer breaks



At $\lambda = 0$, the solution is simply $x^\star(0) = y$



Strategy: construct a sequence of λ values $\lambda_1 \leq \lambda_2 \leq \ldots$ at which adjacent coordinates of $x^\star(\lambda)$ are equal or "fused" (Hoefling, 2009)

In between these critical values $\lambda_1 \leq \lambda_2 \leq \ldots$, solution $x^\star(\lambda)$ is simply linear: if $\lambda \in [\lambda_k, \lambda_{k+1}]$,

$$x^\star(\lambda) = \alpha \cdot x^\star(\lambda_k) + (1-\alpha) \cdot x^\star(\lambda_{k+1})$$

where $\alpha = (\lambda_{k+1} - \lambda)/(\lambda_{k+1} - \lambda_k)$

How many critical values are there? Can there be more than $n-1$?

We will rely on the following useful fact:

Lemma (Friedman et al., 2007): For any coordinate i of the 1d fused lasso solution, if $x_i^\star(\lambda) = x_{i+1}^\star(\lambda)$, then $x_i^\star(\lambda') = x_{i+1}^\star(\lambda')$ for all $\lambda' > \lambda$.

I.e., once two coordinates fuse, they can never unfuse. So there are exactly $n-1$ critical points

Our problem:

$$\min_{x\in\mathbb{R}^n}\ \frac{1}{2}\sum_{i=1}^n (y_i - x_i)^2 + \lambda \sum_{i=1}^{n-1} |x_i - x_{i+1}|$$

The KKT conditions:

$$0 = x_i - y_i + \lambda(s_i - s_{i-1}), \quad i = 1, \dots n$$

where $s_i \in \partial |x_i - x_{i+1}|$, $i = 1, \dots n-1$ (and $s_0 = s_n = 0$)

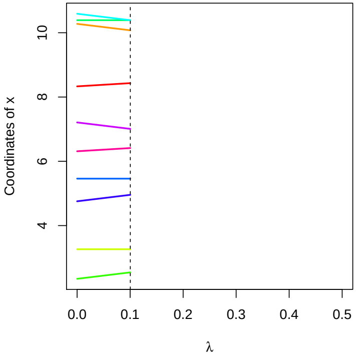
At $\lambda = 0$, $x^\star(0) = y$. For a small value $\lambda > 0$, consider taking

$$x_i^\star(\lambda) = y_i - \lambda\Big(\mathrm{sign}(y_i - y_{i+1}) - \mathrm{sign}(y_{i-1} - y_i)\Big), \quad i = 1, \dots n$$

This satisfies the KKT conditions, and is hence a valid solution at λ, as long as $\mathrm{sign}(x_i^\star(\lambda) - x_{i+1}^\star(\lambda)) = \mathrm{sign}(y_i - y_{i+1})$ for all i

I.e., $x_i^\star(\lambda) = a_i - \lambda \cdot b_i$, linear function of λ, for $i = 1, \ldots n$

This is a valid solution as long as $x_i^\star(\lambda)$ and $x_{i+1}^\star(\lambda)$ don't cross for some i



The critical value of λ at which this happens is

$$\lambda_1 = \min_{i=1,\ldots n-1} \frac{a_i - a_{i+1}}{b_i - b_{i+1}}$$

Therefore we have computed the solution path $x^\star(\lambda)$ exactly for all $\lambda \in [0, \lambda_1]$

Now invoke our lemma: coordinates 4 and 5 (light blue and green) will remain fused for all $\lambda \leq \lambda_1$

Hence, we can define the fused groups

$$g_1 = \{1\}, \ldots g_3 = \{3\}, g_4 = \{4, 5\}, g_5 = \{6\}, \ldots g_{n-1} = \{n\}$$

and rewrite our problem:

$$\min_{x_{g_1}, \ldots x_{g_{n-1}}} \ \frac{1}{2} \sum_{i=1}^{n-1} \sum_{j \in g_i} (y_j - x_{g_i})^2 + \lambda \sum_{i=1}^{n-2} |x_{g_i} - x_{g_{i+1}}|$$

and the KKT conditions:

$$0 = |g_i| \cdot x_{g_i} - \sum_{j \in g_i} y_j + \lambda(s_{g_i} - s_{g_{i-1}}), \quad i = 1, \ldots n-1,$$

where $s_{g_i} \in \partial |x_{g_i} - x_{g_{i+1}}|$, $i = 1, \ldots n-2$

We know that these KKT conditions are satisfied at λ_1 by $x^\star(\lambda_1)$ (simply a reparametrization)

Hence for $\lambda > \lambda_1$, we propose

$$x^\star_{g_i}(\lambda) = a_i - \lambda \cdot b_i, \quad i = 1, \dots n-1$$

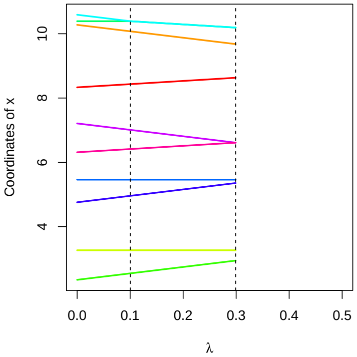
where

$$a_i = \frac{1}{|g_i|} \sum_{j \in g_i} y_j$$

$$b_i = \frac{1}{|g_i|} \Big[\mathrm{sign}\Big(x^\star_{g_i}(\lambda_1) - x^\star_{g_{i+1}}(\lambda_1) \Big) - \mathrm{sign}\Big(x^\star_{g_{i-1}}(\lambda_1) - x^\star_{g_i}(\lambda_1) \Big) \Big]$$

This will satisfy the KKT conditions, and hence be a valid solution, so long as $\mathrm{sign}(x^\star_{g_i}(\lambda) - x^\star_{g_{i+1}}(\lambda)) = \mathrm{sign}(x^\star_{g_i}(\lambda_1) - x^\star_{g_{i+1}}(\lambda_1))$ for all i

I.e., this is a valid solution as long as two adjacent coordinate paths don't cross



The next critical value of λ at which this happens is

$$\lambda_2 = \min_{i=1,\ldots n-2} \frac{a_i - a_{i+1}}{b_i - b_{i+1}}$$

(minimization here is only over values $\geq \lambda_1$)

Now have computed the path $x^\star(\lambda)$ exactly for all $\lambda \in [0, \lambda_2]$

This strategy can be repeated until all coordinates are fused into one single group — the end of the path

Summary of **1d fused lasso path algorithm:**

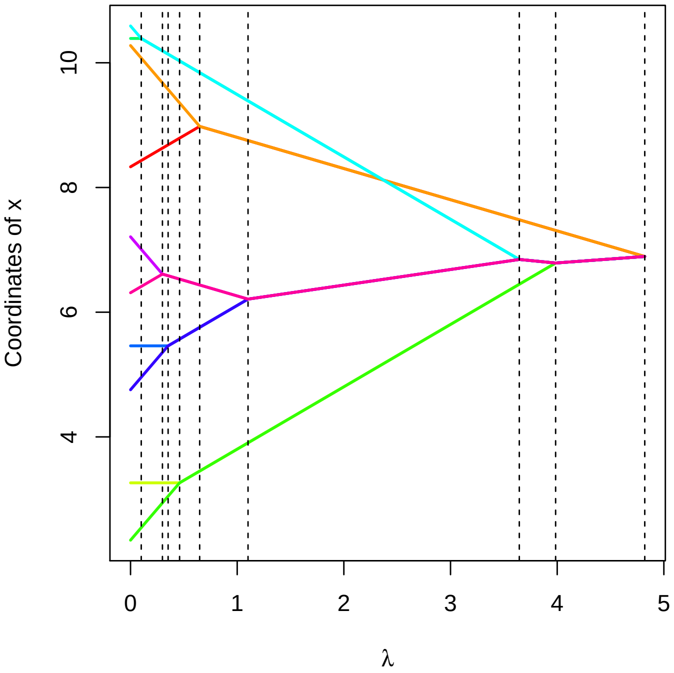
- Start with $\lambda_0 = 0$, $G = n$, $g_i = \{i\}$, $i = 1, \dots n$, $x^\star(0) = y$
- For $k = 1, \dots n-1$:
  - Compute $a_i = \frac{1}{|g_i|} \sum_{j \in g_i} y_j$ and

$$b_i = \frac{1}{|g_i|} \Big[\mathrm{sign}\Big(x^\star_{g_i}(\lambda_{k-1}) - x^\star_{g_{i+1}}(\lambda_{k-1}) \Big) - \mathrm{sign}\Big(x^\star_{g_{i-1}}(\lambda_{k-1}) - x^\star_{g_i}(\lambda_{k-1}) \Big) \Big]$$

  - With $x^\star_{g_i}(\lambda) = a_i - \lambda \cdot b_i$, $i = 1, \dots G$, increase λ until the next critical value:

$$\lambda_k = \min_{i=1,\dots G-1} \frac{a_i - a_{i+1}}{b_i - b_{i+1}}$$

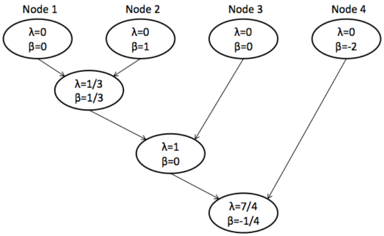
    (minimization is only over values $\geq \lambda_{k-1}$)
  - Merge the appropriate groups, and decrement G


Coordinates of x
10
8
6
4
0
1
2
3
4
5
λ

Computational complexity

Naive implementation: each iteration takes $O(n)$ operations (scan over crossing times of all adjacent fused groups), and there are $O(n)$ iterations (number of fused groups decreases by one at each iteration), so total complexity is $O(n^2)$

Tree-based implementation: note that, after a fusion, crossing times only change for neighbors of modified group. Therefore we can store the solution path in a tree; each leaf tells us a crossing time for an adjacent group, after



[1]

a fusion we can update the tree in constant number of operations, and finding the next (minimum) crossing time requires $O(\log n)$ operations. Hence $O(n \log n)$ total operations

[1] From Hoefling (2009), *A path algorithm for the fused lasso signal approximator*

Extensions

Computing the exact solution path in $O(n \log n)$ operations is **very fast**. Are there extensions beyond 1d case?

- For fused lasso over arbitrary graphs — e.g., consider 2d grid (image denoising) — the key lemma **does not hold:** as we increase λ, fused groups can unfuse. Counterintuitive! Now have to check, at each iteration, for both groups fusing and unfusing; each iteration can be reduced to solving a max flow problem, which is unfortunately more costly
- Alternatively, we could have derived a path algorithm for the dual problem, which is arguably simpler for fused lasso over an arbitrary graph
- Extensions to the regression loss $\|y - Ax\|^2$ (before we were considering $A = I$) are also possible

Path algorithms in statistics and ML

Exact path algorithms can be derived for, e.g., lasso, fused lasso over an arbitrary graph, trend filtering, locally adaptive regression splines, SVMs and kernel SVMs, 1-norm SVMs, relaxed maximum entropy problem

In all these examples, solution path is **piecewise linear** in λ, so problem reduces to finding critical values $\lambda_1, \lambda_2, \ldots \lambda_T$

Unfortunately, for majority of above examples, tight bounds for number critical values T are not known

Empirically, number of critical values grows large with increasing problem sizes, so path algorithms are not scalable to huge problems

Approximate path algorithms can be derived for problems in which path is not piecewise linear, e.g., lasso GLMs

Path algorithms and tuning parameter selection

Recall the general problem form

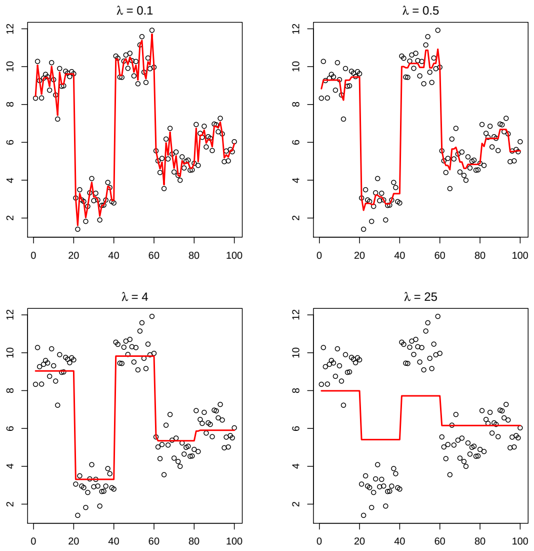
$$\min_{x \in \mathbb{R}^n} \; g(x) + \lambda \cdot h(x)$$

where $\lambda \geq 0$. This parameter balances the effective importance of two terms, controlling the amount of underfitting or overfitting

Path algorithms trace out the solution as a function of tuning parameter λ — hence they provide a complete description of this tradeoff, and often aid in statistical understanding of optimization problem

On a more practical note, the choice of λ is **critical** for essentially any statistical application; when computable, path algorithms can be helpful for this task

Example: 1d fused lasso problem over various choices of λ



At a high level, here is one such way to do this: start off by assuming that we observe

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

where $\mu \in \mathbb{R}^n$ is unknown signal to be estimated, and ϵ_i are i.i.d. errors with $\mathbb{E}[\epsilon_i] = 0$ and $\mathrm{Var}[\epsilon_i] = \sigma^2$

From y, we compute an estimate $\hat{y}$; e.g., this can be the solution $x^\star$ of an optimization problem. Define the associated prediction error

$$\mathrm{PE}(\hat{y}) = \mathbb{E}\|\hat{y} - y'\|^2$$

where y' is an i.i.d. copy of y

Note the expansion

$$\mathbb{E}\|\hat{y} - y'\|^2 = \mathbb{E}\|\hat{y} - y\|^2 + 2\sum_{i=1}^{n} \mathrm{Cov}(\hat{y}_i, y_i)$$

where the quantity

$$\mathrm{df}(\hat{y}) = \frac{1}{\sigma^2}\sum_{i=1}^{n} \mathrm{Cov}(\hat{y}_i, y_i)$$

is called the **degrees of freedom** of $\hat{y}$. Think of as the effective number of parameters used by $\hat{y}$

Suppose that we knew an unbiased estimate for degrees of freedom of $\hat{y}$, i.e., $\mathbb{E}[\widehat{\mathrm{df}}(\hat{y})] = \mathrm{df}(\hat{y})$. Then

$$\widehat{\mathrm{PE}}(\hat{y}) = \|\hat{y} - y\|^2 + 2\sigma^2\widehat{\mathrm{df}}(\hat{y})$$

is an unbiased estimate for $\mathrm{PE}(\hat{y})$, i.e., $\mathbb{E}[\widehat{\mathrm{PE}}(\hat{y})] = \mathrm{PE}(\hat{y})$

Now suppose that our estimate $\hat{y}$ depends on tuning parameter λ, written $\hat{y}_\lambda$. If we could compute $\widehat{\mathrm{df}}(\hat{y}_\lambda)$, then we could **choose $\boldsymbol{\lambda}$ to minimize:**

$$\widehat{\mathrm{PE}}(\hat{y}_\lambda) = \|\hat{y}_\lambda - y\|^2 + 2\sigma^2 \widehat{\mathrm{df}}(\hat{y}_\lambda)$$

our best estimate for prediction error. Note in the case of

- Overfitting: training error $\|\hat{y}_\lambda - y\|^2$ is small, but degrees of freedom $\widehat{\mathrm{df}}(\hat{y}_\lambda)$ is large
- Underfitting: degrees of freedom $\widehat{\mathrm{df}}(\hat{y}_\lambda)$ is small, but training error $\|\hat{y}_\lambda - y\|^2$ is large

I.e., choosing λ to minimize $\widehat{\mathrm{PE}}$ balances performance in training sample with model complexity

So, how to compute $\widehat{\mathrm{df}}(\hat{y}_\lambda)$? Stein's formula (Stein, 1981) provides a way to do this: under some regularity conditions, we can use the estimate $\widehat{\mathrm{df}}(\hat{y}_\lambda) = \sum_{i=1}^n \partial \hat{y}_{\lambda,i} / \partial y_i$

How do path algorithms fit in?

- Because they trace out the exact solution path, it is often easier to compute $\widehat{\mathrm{df}}(\hat{y}_\lambda)$, as λ varies, using a path algorithm. In many cases, this presents no extra work
- To choose λ, we are faced with nonconvex problem

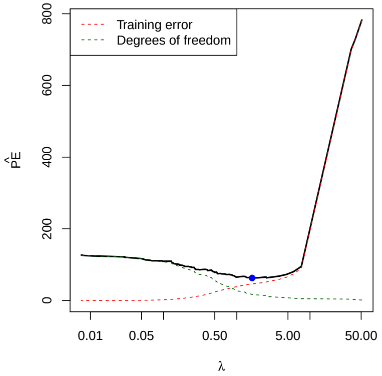
$$\min_{\lambda \geq 0} \|\hat{y}_\lambda - y\|^2 + 2\sigma^2 \widehat{\mathrm{df}}(\hat{y}_\lambda)$$

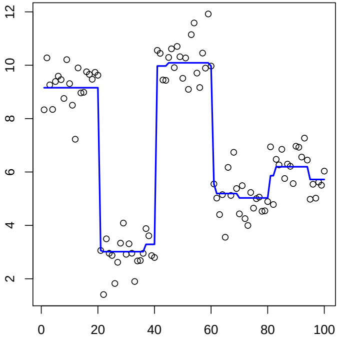
But in many examples, $\widehat{\mathrm{df}}(\hat{y}_\lambda)$ is piecewise constant as with respect to λ, i.e., constant in between critical values; and $\|\hat{y}_\lambda - y\|^2$ is monotone in between critical values. Therefore our problem reduces to

$$\min_{\lambda \in \{\lambda_1, \ldots \lambda_T\}} \|\hat{y}_\lambda - y\|^2 + 2\sigma^2 \widehat{\mathrm{df}}(\hat{y}_\lambda)$$

so minimizing value of λ can be found by simply checking each critical value λ_i visited by path algorithm

Example with 1d fused lasso: here $\widehat{\mathrm{df}}(\hat{y}_\lambda)$ is simply the number of fused groups in $\hat{y}_\lambda$. Choice of tuning parameter: $\lambda = 1.62$





References

- M. Dubiner, M. Gavish, and Y. Singer (2012), *The maximum entropy relaxation path*
- B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani (2004), *Least angle regression*
- T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu (2004), *The entire regularization path for the support vector machine*
- H. Hoefling (2009), *A path algorithm for the fused lasso signal approximator*
- M. Park and T. Hastie (2006), ℓ_1 *regularization path algorithm for generalized linear models*
- S. Rosset and J. Zhu (2007), *Piecewise linear regularized solution paths*
- R. J. Tibshirani and J. Taylor (2011), *The solution path of the generalized lasso*
- J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani (2003), *1-norm support vector machines*