Case Studies: Sum of Norms Regularization Problems

Ryan Tibshirani
Convex Optimization 10-725/36-725

Last time: review of optimization toolbox

So far we've learned about:

- First-order methods
- Newton methods
- Interior point methods

These comprise a good part of the core tools in optimization, and are a big focus in this field

(Still, there's a lot more out there. Before the course is over we'll see dual methods and ADMM, coordinate descent, fast stochastic methods, projected Newton ...)

Given the number of available tools, it may seem overwhelming to choose a method in practice. A fair question: how to know what to use when?

It's not possible to give a complete answer to this question. But the big algorithms table from last time gave guidelines. It covered:

- Assumptions on criterion function
- Assumptions on constraint functions/set
- Ease of implementation (how to choose parameters?)
- Cost of each iteration
- Number of iterations needed

Other important aspects, that it didn't consider: parallelization, data storage issues, statistical interplay

Here, as an example, we walk through some of the high-level reasoning for related but distinct regularized estimation problems

Sum of norms regularization

We will consider problems of the form

$$\min_{\beta} \; f(\beta) + \lambda \sum_{j=1}^{J} \|D_j \beta\|_{q_j}$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a smooth, convex function, $D_j \in \mathbb{R}^{m_j \times p}$ is a penalty matrix, $q_j \geq 1$ is a norm parameter, for $j = 1, \ldots J$. Also, $\lambda \geq 0$ is a regularization parameter

An obvious special case: the lasso fits into this framework, with

$$f(\beta) = \|y - X\beta\|_2^2$$

and $J = 1$, $D = I$, $q = 1$. To include an unpenalized intercept, we just add a column of zeros to D

Outline

Today:

- Notable examples
- Algorithmic considerations
- Case studies

} Back and forth

- Implementation tips

Fused lasso or total variation denoising, 1d

Special case: fused lasso or total variation denoising in 1d, where $J = 1$, $q = 1$, and

$$D = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}, \text{ so } \|D\beta\|_1 = \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}|$$

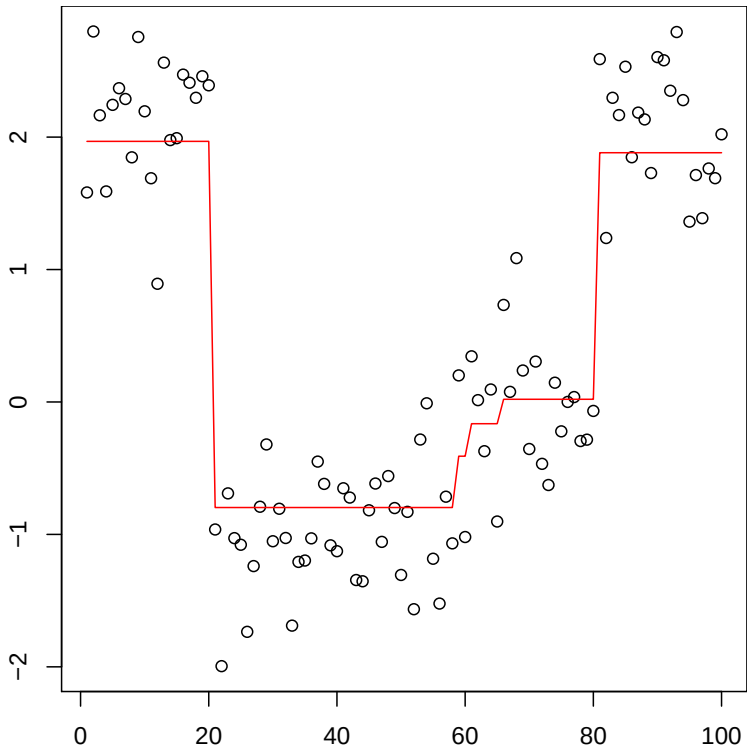
Now we obtain sparsity in adjacent differences $\hat{\beta}_i - \hat{\beta}_{i+1}$, i.e., we obtain $\hat{\beta}_i = \hat{\beta}_{i+1}$ at many locations i

Hence, plotted in order of the locations $i = 1, \dots n$, the solution $\hat{\beta}$ appears piecewise constant

Typically used in "signal approximator" settings, where $\hat{\beta}$ estimates (say) the mean of some observations $y \in \mathbb{R}^n$ directly. Examples:

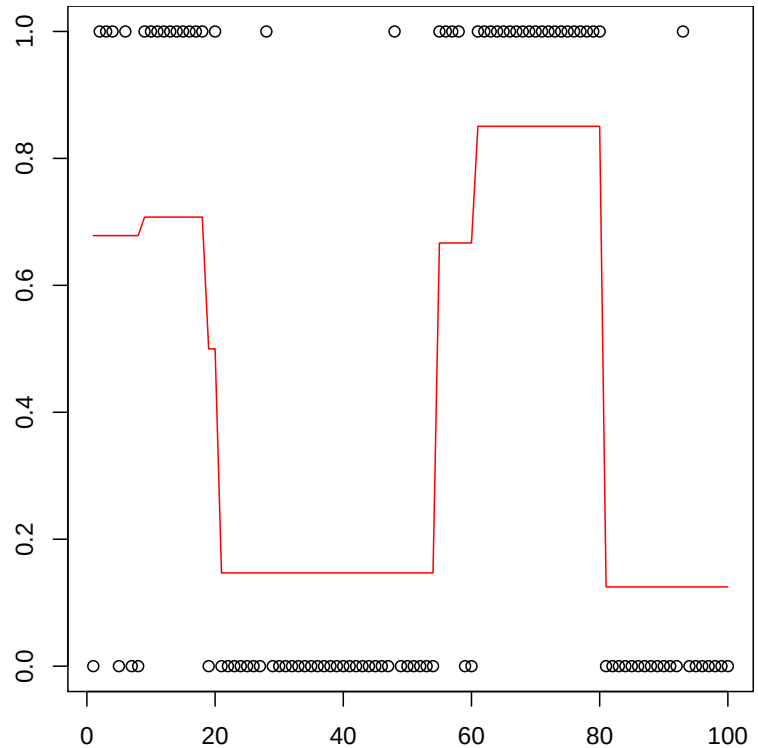
Gaussian loss

$$f(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_i)^2$$



Logistic loss

$$f(\beta) = \sum_{i=1}^{n}(-y_i\beta_i + \log(1 + e^{\beta_i}))$$



Fused lasso or total variation denoising, graphs

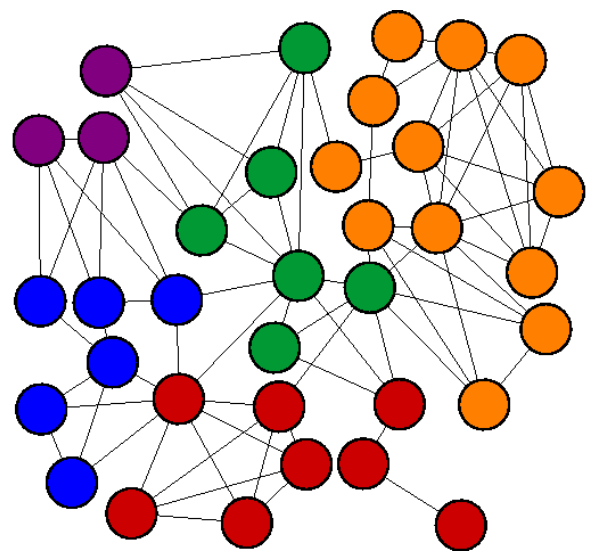
Special case: fused lasso or total variation denoising over a graph, $G = (\{1, \ldots n\}, E)$. Here D is $|E| \times n$, and if $e_\ell = (i, j)$, then D has ℓth row

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

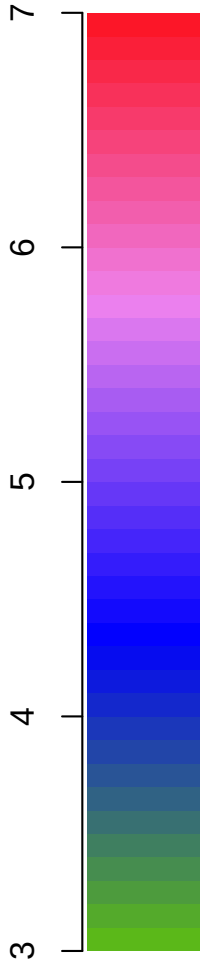
so

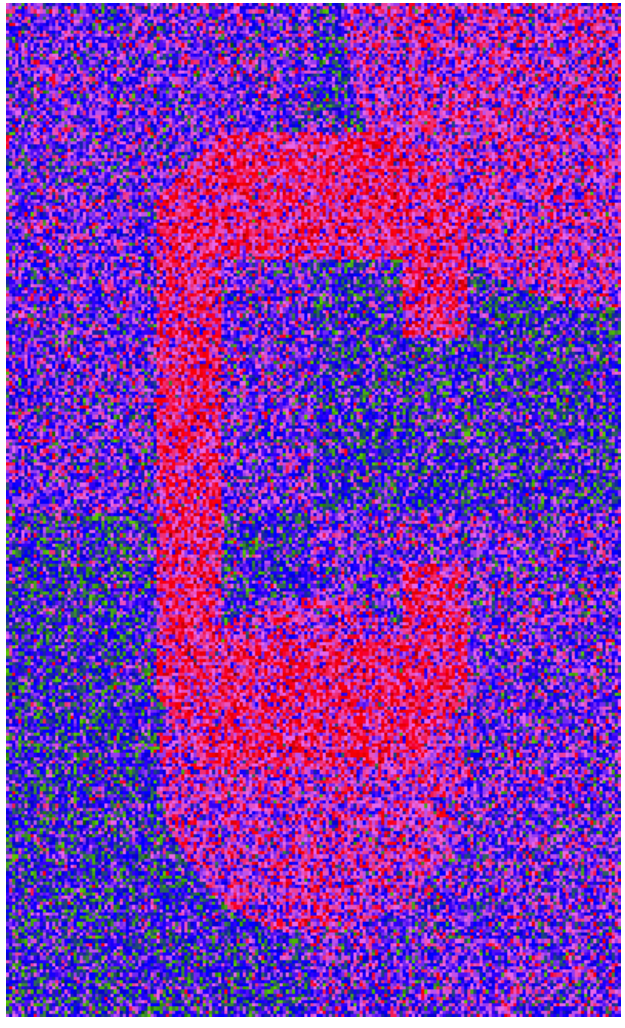
$$\|D\beta\|_1 = \sum_{(i,j) \in E} |\beta_i - \beta_j|$$

Now at the solution, we get $\hat{\beta}_i = \hat{\beta}_j$ across many edges $(i, j) \in E$, so $\hat{\beta}$ is piecewise constant over the graph G

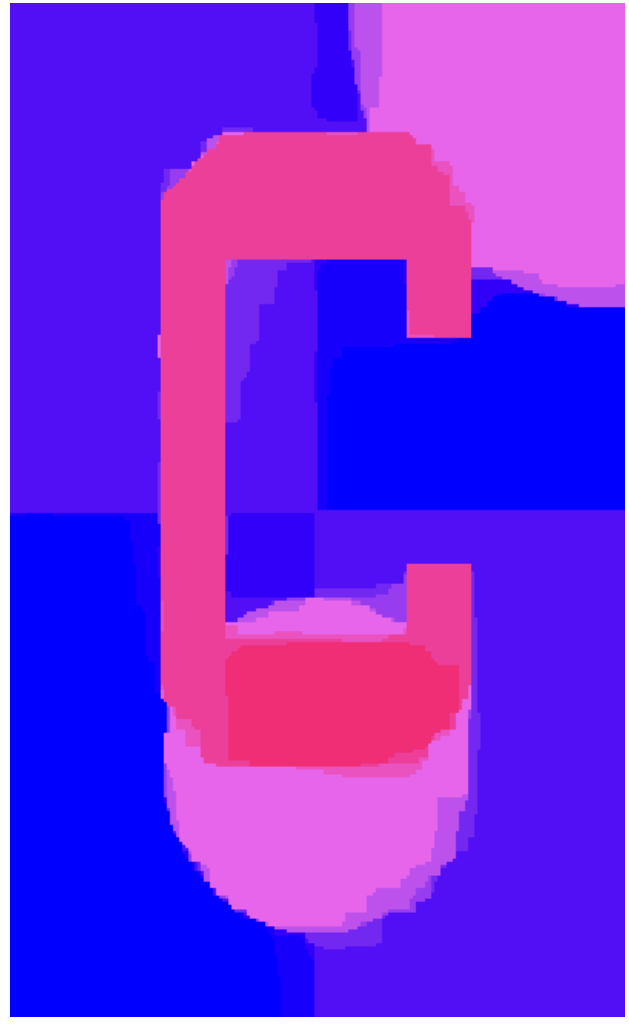


Example: Gaussian loss, $f(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_i)^2$, 2d grid graph



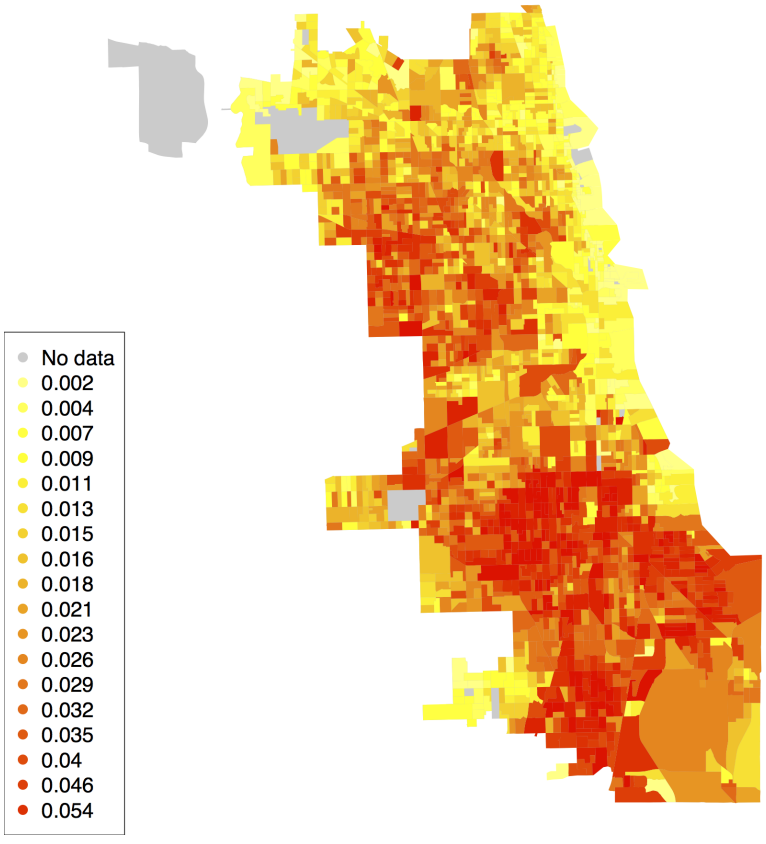


Data (noisy image)

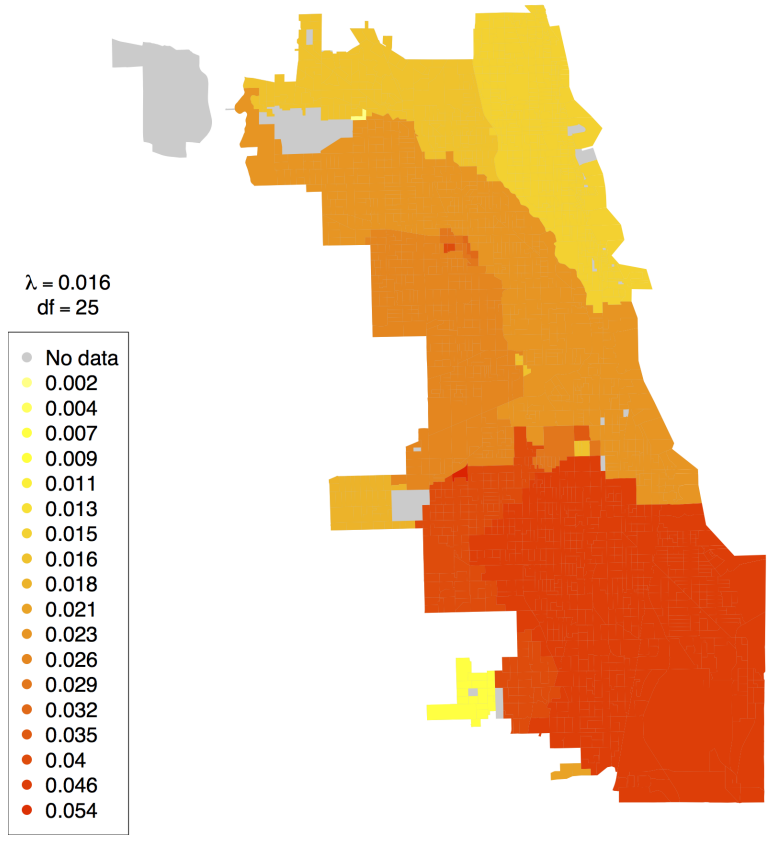


Solution (denoised image)

Example: Gaussian loss, $f(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_i)^2$, Chicago graph



Data (observed crime rates)



Solution (estimated crime rates)

Fused lasso with a regressor matrix

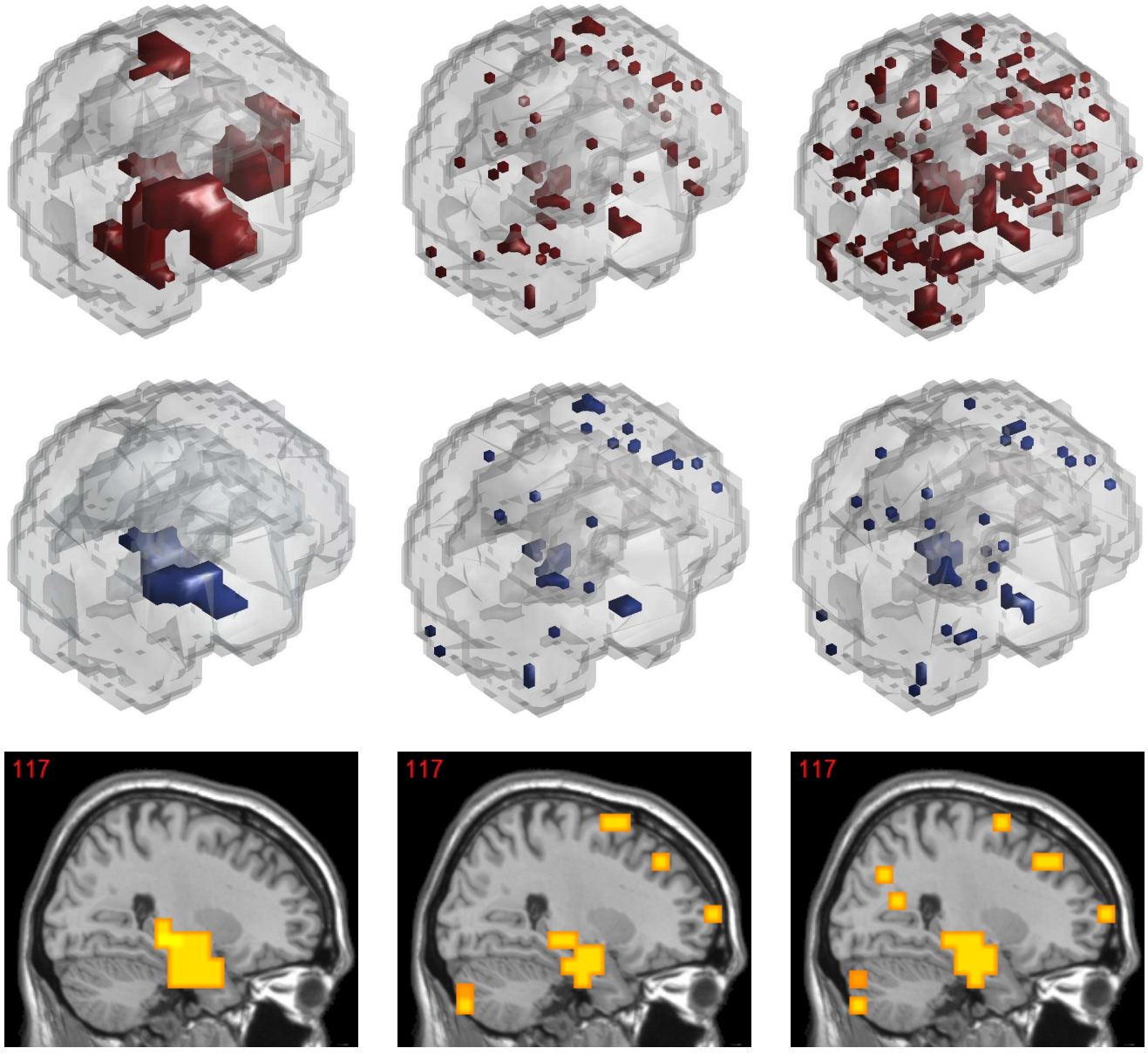
Suppose $X \in \mathbb{R}^{n \times p}$ is a predictor matrix, with structure present in the relationships between its columns. In particular, suppose that the columns have been measured over nodes of a graph

Consider $J = 1$, $q = 1$, fused lasso matrix D over graph, and

$$f(\beta) = \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 \quad \text{or}$$

$$f(\beta) = \sum_{i=1}^{n} \big(- y_i x_i^T \beta + \log(1 + \exp(x_i^T \beta)) \big)$$

(where x_i, $i = 1, \ldots n$ are the rows of X). Here we are performing linear or logistic regression with estimated coefficients that will be constant over regions of the graph (this reveals groups of related predictors)



(From Xin et al. (2014), "Efficient generalized fused lasso and its application to the diagnosis of Alzheimer's disease")

Algorithms for the fused lasso

Let's go through our toolset, to figure out how to solve

$$\min_{\beta} \; f(\beta) + \lambda \|D\beta\|_1$$

Subgradient method: subgradient of criterion is

$$g = \nabla f(\beta) + \lambda D^T \gamma$$

where $\gamma \in \partial \|x\|_1$ evaluated at $x = D\beta$, i.e.,

$$\gamma_i \in \begin{cases} \{\mathrm{sign}\big((D\beta)_i\big)\} & \text{if } (D\beta)_i \neq 0 \\ [-1, 1] & \text{if } (D\beta)_i = 0 \end{cases}, \quad i = 1, \ldots m$$

Downside (as usual) is that convergence is slow. Upside is that g is easy to compute (provided ∇f is): if $S = \mathrm{supp}(D\beta)$, then we let

$$g = \nabla f(\beta) + \lambda \sum_{i \in S} \mathrm{sign}\big((D\beta)_i\big) \cdot D_i$$

Proximal gradient descent: prox operator is

$$\text{prox}_t(\beta) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|\beta - z\|_2^2 + \lambda\|Dz\|_1$$

This is not easy for a general difference operator D (compare this to soft-thresholding, if $D = I$). Prox itself is the fused lasso signal approximator problem, with a Gaussian loss!

Could try reparametrizing the term $\|D\beta\|_1$ to make it linear, while introducing inequality constraints. We could then apply an interior point method

But we will have better luck going to the dual problem. (In fact, it is never a bad idea to look at the dual problem, even if you have a good approach for the primal problem!)

Fused lasso dual problem

Our problems are

$$\text{Primal} \;:\; \min_{\beta} \; f(\beta) + \lambda \|D\beta\|_1$$

$$\text{Dual} \;:\; \min_{u} \; f^*(-D^T u) \;\; \text{subject to} \;\; \|u\|_\infty \leq \lambda$$

Here f^* is the conjugate of f. Note that $u \in \mathbb{R}^m$ (where m is the number of rows of D) while $\beta \in \mathbb{R}^n$

The primal and dual solutions $\hat{\beta}$, $\hat{u}$ are linked by KKT conditions:

$$\nabla f(\hat{\beta}) + D^T \hat{u} = 0, \;\; \text{and}$$

$$\hat{u}_i \in \begin{cases} \{\lambda\} & \text{if } (D\hat{\beta})_i > 0 \\ \{-\lambda\} & \text{if } (D\hat{\beta})_i < 0 \\ [-\lambda, \lambda] & \text{if } (D\hat{\beta})_i = 0 \end{cases}, \quad i = 1, \ldots m$$

Second property implies that: $\hat{u}_i \in (-\lambda, \lambda) \Longrightarrow (D\hat{\beta})_i = 0$

Let's go through our toolset, to think about solving dual problem

$$\min_u \; f^*(-D^T u) \;\; \text{subject to} \;\; \|u\|_\infty \leq \lambda$$

Note the eventually we'll need to solve $\nabla f(\hat{\beta}) = -D^T \hat{u}$ for primal solution, and tractability of this depends on f

Proximal gradient descent: looks much better now, because prox is

$$\text{prox}_t(u) = \underset{z}{\text{argmin}} \; \frac{1}{2t}\|u - z\|_2^2 \;\; \text{subject to} \;\; \|z\|_\infty \leq \lambda$$

is easy. This is projection onto a box $[-\lambda, \lambda]^m$, i.e., prox returns $\hat{z}$ with

$$\hat{z}_i = \begin{cases} \lambda & \text{if } u_i > \lambda \\ -\lambda & \text{if } u_i < -\lambda \\ u_i & \text{if } u_i \in [-\lambda, \lambda] \end{cases}, \quad i = 1, \ldots m$$

Interior point method: rewrite dual problem as

$$\min_u \; f^*(-D^T u) \;\; \text{subject to} \;\; -\lambda \leq u_i \leq \lambda, \;\; i = 1, \ldots m$$

These are just linear constraints, so we can easily form log barrier[1] as in

$$\min_u \; t \cdot f^*(-D^T u) + \phi(u)$$

where

$$\phi(u) = -\sum_{i=1}^{m} \big(\log(\lambda - u_i) + \log(u_i + \lambda) \big)$$

We either solve above problem with Newton's method, or take one Newton step, and then increase t

How efficient are Newton updates?

[1] There could be extra constraints from the domain of f^*, e.g., this happens when f is the logistic loss, so these add extra log barrier terms

Define the barrier-smoothed dual criterion function

$$F(u) = tf^*(-D^T u) + \phi(u)$$

Newton updates follow direction $H^{-1}g$, where

$$g = \nabla F(u) = -t \cdot D\big(\nabla f^*(-D^T u)\big) + \nabla\phi(u)$$
$$H = \nabla^2 F(u) = t \cdot D\big(\nabla^2 f^*(-D^T u)\big)D^T + \nabla^2\phi(u)$$

How difficult is it to solve a linear system in H?

- First term: if Hessian of the loss term $\nabla^2 f^*(v)$ is structured, and D is structured, then often $D\nabla^2 f^*(v)D^T$ is structured
- Second term: Hessian of log barrier term $\nabla^2\phi(u)$ is diagonal

So it really depends critically on first term, i.e., on conjugate loss f^* and penalty matrix D

Putting it all together:

- Primal subgradient method: iterations are cheap (we sum up rows of D over active set S), but convergence is slow
- Primal proximal gradient: iterations involve evaluating

$$\text{prox}_t(\beta) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|\beta - z\|_2^2 + \lambda\|Dz\|_1$$

  which can be very expensive, convergence is medium
- Dual proximal gradient: iterations involve projecting onto a box, so very cheap, convergence is medium
- Dual interior point method: iterations involve a solving linear $Hx = g$ system in

$$H = t \cdot D\big(\nabla^2 f^*(-D^T u)\big)D^T + \nabla^2\phi(u)$$

  which may or may not be expensive, convergence is rapid

Case study: fused lasso, Gaussian or logistic signal approximation

Suppose that we wish to solve, for D a difference operator over a general graph,

$$\min_{\beta} \ \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_i)^2 + \lambda\|D\beta\|_1, \quad \text{or}$$

$$\min_{\beta} \ \sum_{i=1}^{n}\big(-y_i\beta_i + \log(1+e^{\beta_i})\big) + \lambda\|D\beta\|_1$$

Suppose further that we desire solution at a high level of accuracy, otherwise, we notice undesired artifacts and bumps when plotting $\hat{\beta}$. What algorithm should we use?

Primal subgradient and primal proximal gradient are out (slow and intractable, respectively)

As for dual algorithms, one can check that the conjugate f^* has a closed-form for both the Gaussian and logistic cases:

$$f^*(v) = \frac{1}{2}\sum_{i=1}^n y_i^2 - \frac{1}{2}\sum_{i=1}^n (y_i + v_i)^2 \quad \text{and}$$

$$f^*(v) = \sum_{i=1}^n \Big((v_i + y_i)\log(v_i + y_i) + (1 - v_i - y_i)\log(1 - v_i - y_i) \Big)$$

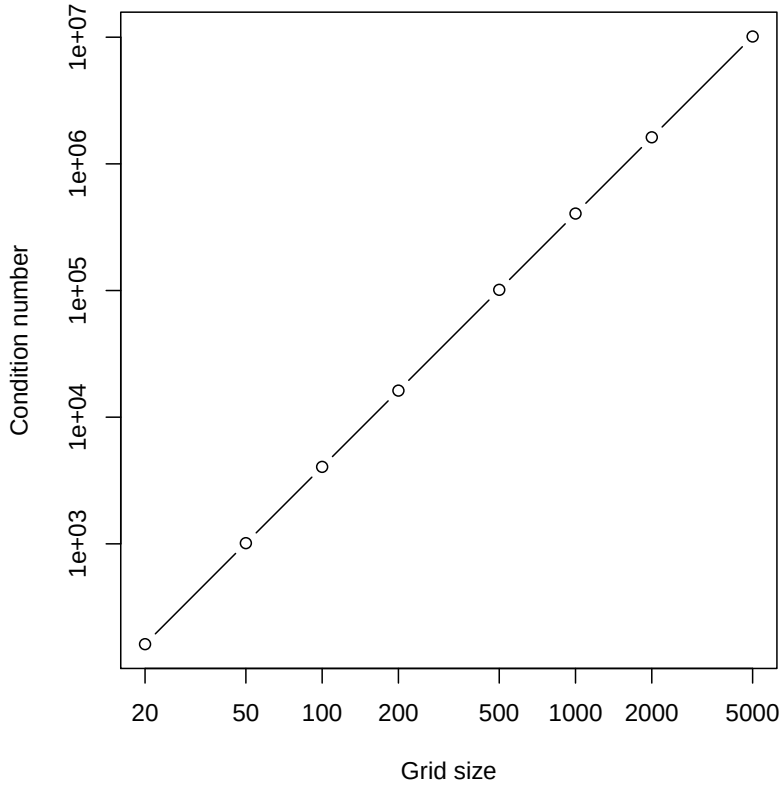
respectively. We also the have expressions for primal solutions

$$\hat{\beta} = y - D^T \hat{u} \quad \text{and}$$

$$\hat{\beta}_i = -y_i \log\big(y_i (D^T \hat{u})_i\big) + y_i \log\big(1 - y_i (D^T \hat{u})_i\big), \quad i = 1, \ldots n$$

respectively

Dual proximal gradient descent admits very efficient iterations, as it just projects $u + tD\nabla f^*(-D^T u)$ onto a box, repeatedly. But it takes far too long to converge to high accuracy; depending on the graph, this can even be worse than usual, because of conditioning



(This shows condition numbers for a 2d grid graph ... one of the better conditioned cases ... still not great!)

Importantly, $\nabla^2 f^*(v)$ is a diagonal matrix in both the Gaussian and logistic cases:

$$\nabla^2 f^*(v) = I \quad \text{and}$$

$$\nabla^2 f^*(v) = \mathrm{diag}\left(\frac{1}{v_i + y_i} + \frac{1}{1 - v_i - y_i},\ i = 1, \ldots m\right)$$

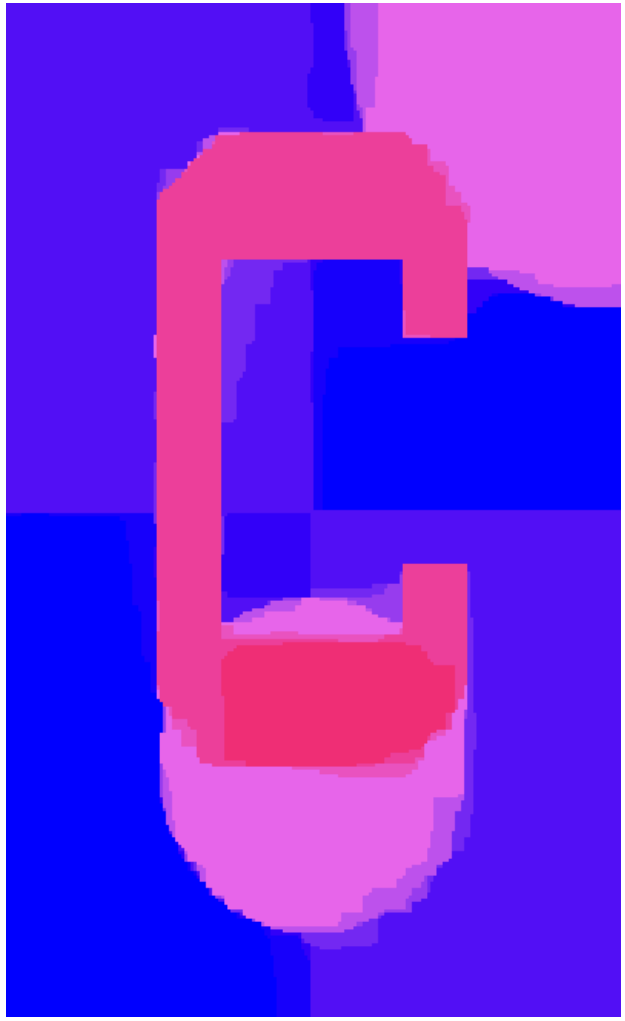
respectively. Therefore the Newton steps in a dual interior point method involve solving a linear system $Hx = g$ in

$$H = DA(u)D^T + B(u)$$

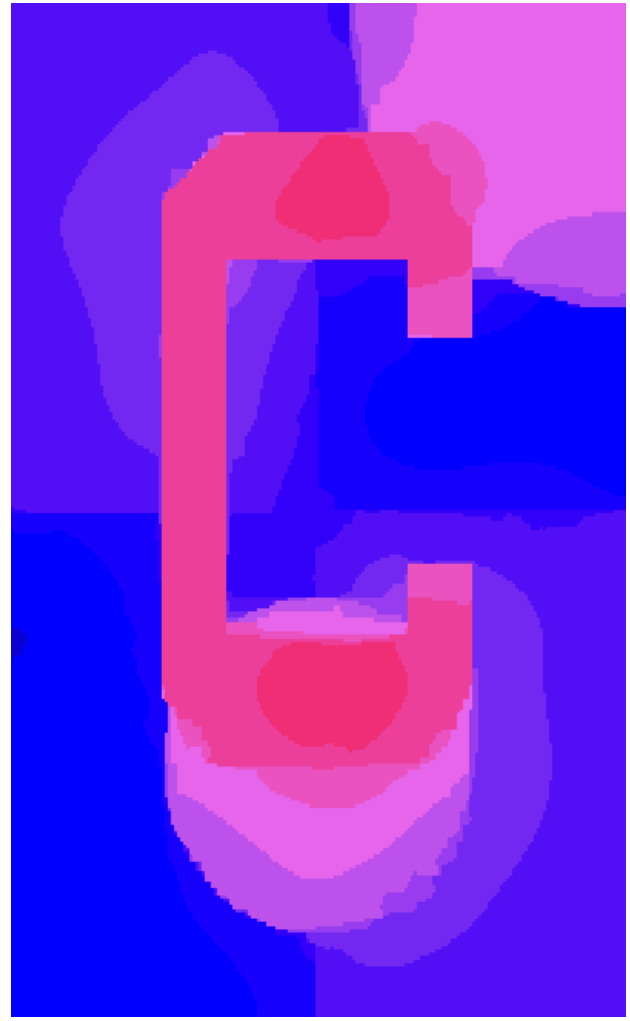
where $A(u), B(u)$ are both diagonal. This is very structured, D is the difference operator over a graph; can be solved efficiently, in close to $O(n)$ flops

Hence, an interior point method on the dual problem is the way to go: cheap iterations, and convergence to high accuracy is very fast

Recall example from our first lecture:



Dual interior point method
10 iterations



Dual proximal gradient
1000 iterations

Case study: fused lasso, linear or logistic regression

(How the story can suddenly change, with a tweak to the problem!)

Consider the same D, but now with a regression matrix $X \in \mathbb{R}^{n \times p}$ (rows x_i, $i = 1, \ldots n$), and losses

$$f(\beta) = \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2, \quad \text{or}$$

$$f(\beta) = \sum_{i=1}^{n} \big(- y_i x_i^T \beta + \log(1 + \exp(x_i^T \beta)) \big)$$

Assume that the predictor matrix X is arbitrary. Everything in the dual is much more complicated now!

Denote by $f(\beta) = h(X\beta)$ the loss. Our problems are

$$\begin{aligned} \text{Primal} \ : \ & \min_{\beta} \ h(X\beta) + \lambda \|D\beta\|_1 \\ \text{Dual} \ : \ & \min_{u,v} \quad h^*(v) \\ & \text{subject to} \ \ X^T v + D^T u = 0, \ \|u\|_\infty \leq \lambda \end{aligned}$$

Here h^* is the conjugate of h. Note that we have $u \in \mathbb{R}^m$, $v \in \mathbb{R}^p$. Furthermore, the primal and dual solutions $\hat{\beta}$ and $\hat{u}, \hat{v}$ satisfy

$$\nabla h(X\hat{\beta}) - \hat{v} = 0 \quad \text{or equivalently}$$
$$X^T \nabla h(X\hat{\beta}) + D^T \hat{u} = 0$$

Computing $\hat{\beta}$ from $\hat{u}$ requires solving a linear system in X, not cheap for generic X

Dual proximal gradient descent has become intractable, because the prox operator is

$$\text{prox}_t(u, v) = \underset{X^T w + D^T z = 0}{\text{argmin}} \frac{1}{2t}\|u - z\|_2^2 + \frac{1}{2t}\|v - w\|_2^2 + \|u\|_\infty$$

This is finding the projection of (u, v) onto the intersection of a plane and a (lower-dimensional) box

Dual interior point methods also don't look nearly as favorable as before, because the equality constraint

$$X^T v + D^T u = 0$$

must be maintained, so we augment the inner linear systems, and this ruins their structure, since X is assumed to be dense

Primal subgradient method is still very slow. Must we use it?

In fact, for large and dense X, our best option is probably to use primal proximal gradient descent. The gradient

$$\nabla f(\beta) = X^T \nabla h(X\beta)$$

is easily computed via the chain rule, and the prox operator

$$\text{prox}_t(\beta) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|\beta - z\|_2^2 + \lambda \|Dz\|_1$$

is not evaluable in closed-form, but it is precisely the same problem we considered solving before: graph fused lasso with Gaussian loss, and without regressors

Hence to (approximately) evaluate the prox, we run a dual interior point method until convergence. We have freed ourselves entirely from solving linear systems in X

Case study: 1d fused lasso, linear or logistic regression

Let's turn to the chain graph in particular, i.e., the case where

$$D = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}$$

The prox function in the primal is

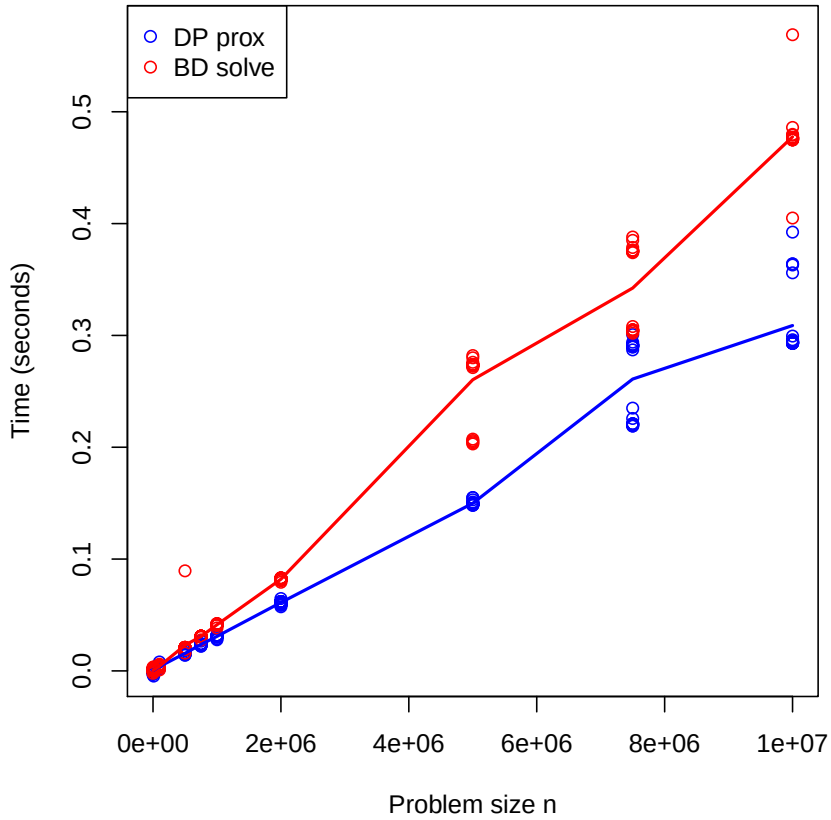
$$\text{prox}_t(\beta) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|\beta - z\|_2^2 + \lambda \sum_{i=1}^{n} |z_i - z_{i+1}|$$

This can be directly computed (!) using specialized approaches such as dynamic programming[2] or taut-string methods[3] in $O(n)$ operations

[2] Johnson (2013), "A dynamic programming algorithm for the fused lasso and L_0-segmentation"

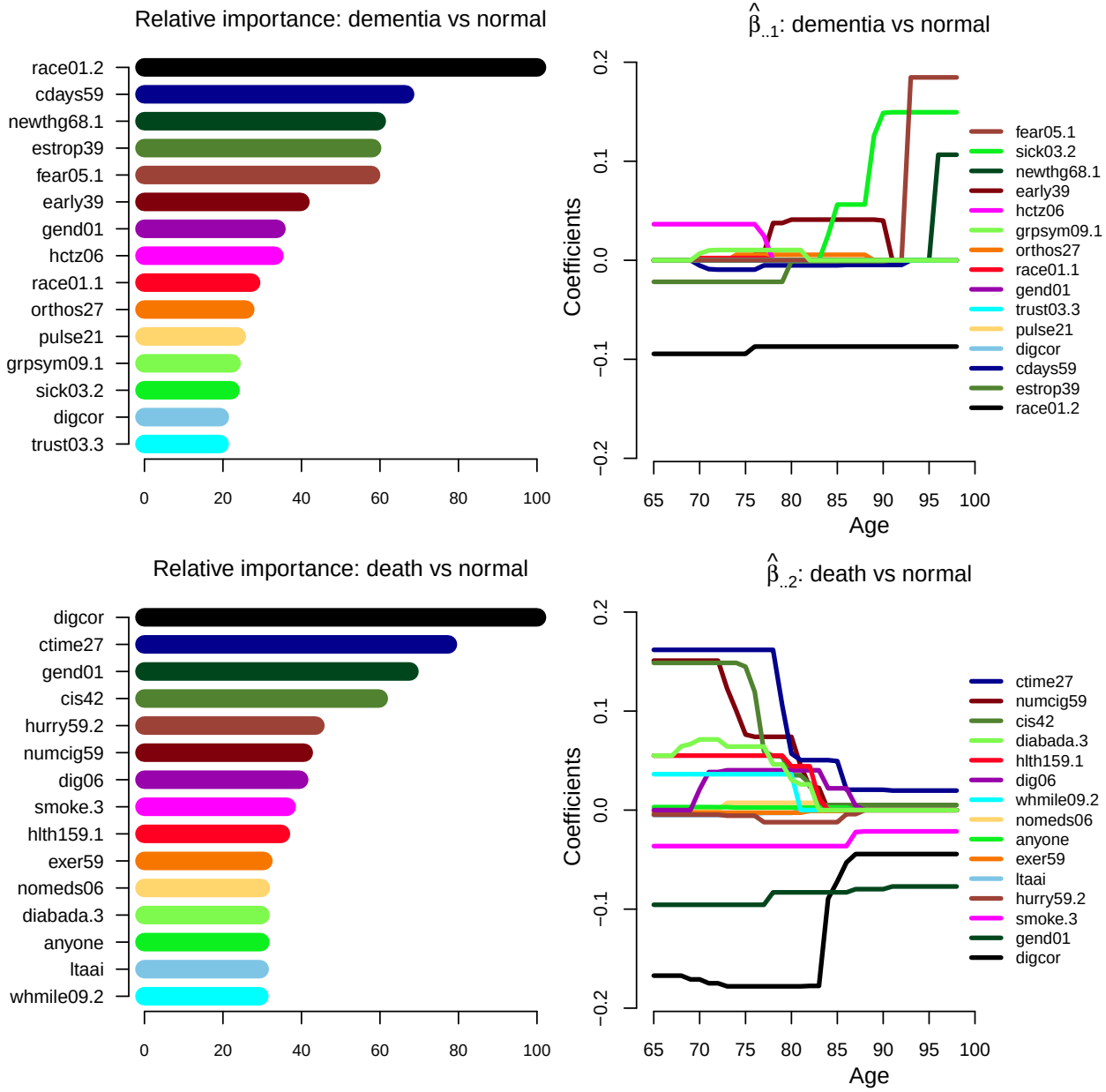
[3] Davies and Kovac (2001), "Local extremes, runs, strings, multiresolution"

How fast is this prox operation, say with dynamic programming?



Dynamic programming
versus
Banded matrix solve

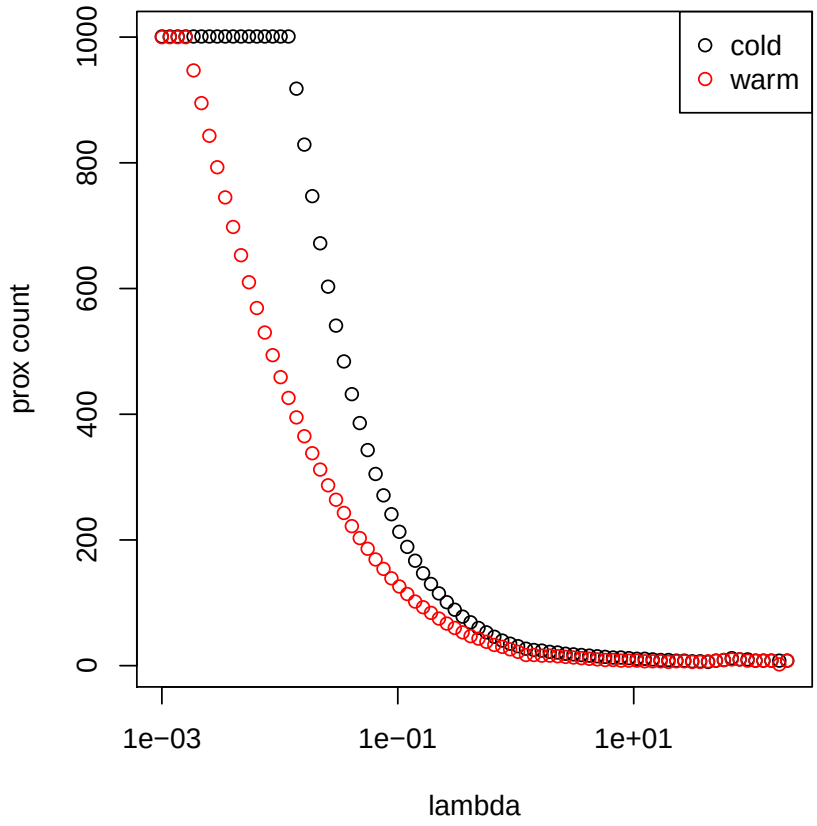
In short, really fast! Hence, primal proximal gradient descent looks especially appealing, because the primal prox is so efficient

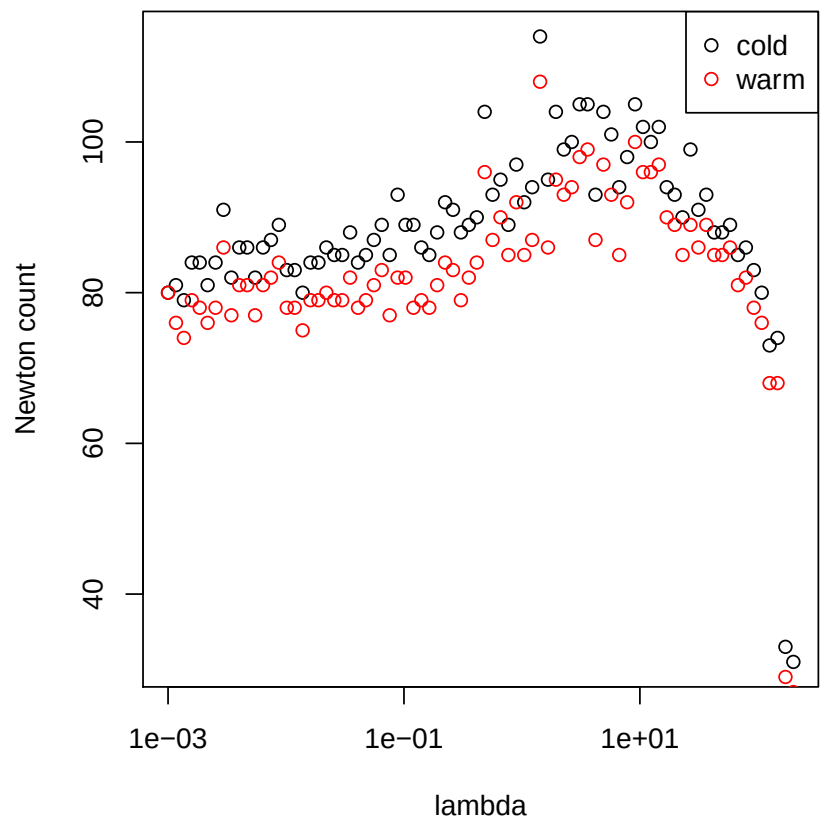


(From Adhikari et al. (2015), "High-dimensional longitudinal classification with the multinomial fused lasso")

Case study: 1d fused lasso, logistic signal approximation

Here both primal proximal gradient and dual interior point method are strong choices. How do they compare? Example, for $n = 2000$:





Primal prox gradient is better for large λ, dual interior point better for small λ. Why this direction?

$$\text{Primal} \;:\; \min_{\beta} \; f(\beta) + \lambda \|D\beta\|_1$$

$$\text{Dual} \;:\; \min_{u} \; f^*(-D^T u) \;\; \text{subject to} \;\; \|u\|_\infty \leq \lambda$$

Observe:

- Large λ: many components $(D\hat{\beta})_i = 0$ in primal, and many components $\hat{u}_i \in (-\lambda, \lambda)$ in dual
- Small λ: many components $(D\hat{\beta})_i \neq 0$ in primal, and many components $|\hat{u}_i| = \lambda$ in dual

When many $(D\hat{\beta})_i = 0$, there are fewer "effective parameters" in the primal optimization; when many $|\hat{u}_i| = \lambda$, the same is true in the dual

Hence, generally:

- Large λ: easier for primal algorithms
- Small λ: easier for dual algorithms

Case study: denser differencing operator D

Consider replacing fused lasso difference operator D with a denser matrix, that gives differences of averages (e.g., think of differences between patches of an image)

Primal prox is intractable, and dual interior point method has too costly Newton steps

But, provided that we can form f^* (and relate the primal and dual solutions), dual proximal gradient still features efficient iterations: the gradient computation $D\nabla f^*(-D^T u)$ is more expensive than it would be if D were sparse, but still not anywhere as expensive as solving a linear system in D

Its iterations simply repeat projecting $u + tD\nabla f^*(-D^T u)$ onto the box $[-\lambda, \lambda]^m$, hence, especially if we do not need a highly accurate solution, dual proximal gradient is the best method

Finally, consider a twist on this problem in which D is dense and massive (many, many rows), and even fitting it in memory is hard

Depending on f and its gradient, primal subgradient method might be the only feasible algorithm; recall the subgradient calculation

$$g = \nabla f(\beta) + \lambda \sum_{i \in S} \text{sign}\big((D\beta)_i\big) \cdot D_i$$

where S is the set of all i such that $(D\beta)_i \neq 0$

If λ is large enough so that many $(D\beta)_i = 0$, then we only need to fit a small part of D in memory (or, read a small part of D from a file) to perform subgradient updates

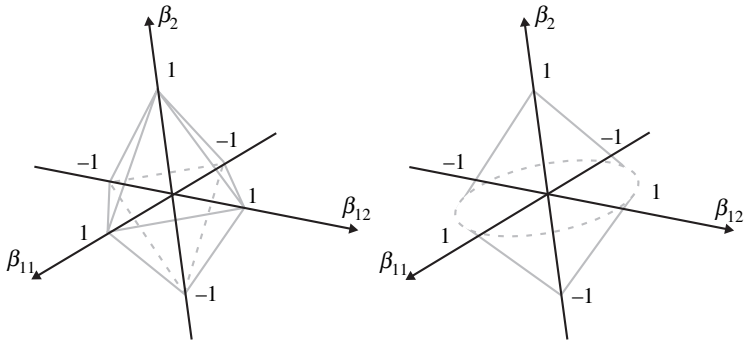
Combined with perhaps a stochastic trick in evaluating either part of g above, this could be effective at large scale

Group lasso problems

Suppose predictors $X = [X_{(1)}\ X_{(2)}\ \ldots\ X_{(J)}]$, split up into groups. To achieve sparsity over groups rather than individual predictors, we write $\beta = (\beta_{(1)}, \ldots \beta_{(J)})$, and solve the group lasso problem:

$$\min_{\beta=(\beta_{(1)},\ldots\beta_{(J)})\in\mathbb{R}^p} h(X\beta) + \lambda \sum_{j=1}^{J} w_j \|\beta_{(j)}\|_2$$

This fits into our framework with $q_j = 2$, and D_j being the matrix that selects out group j (and multiplies by w_j), for $j = 1, \ldots J$



(From Yuan and Lin (2006), "Model selection and estimation in regression with grouped variables")

Note: $q_j = \infty$, $j = 1, \ldots J$ will also work for group penalty

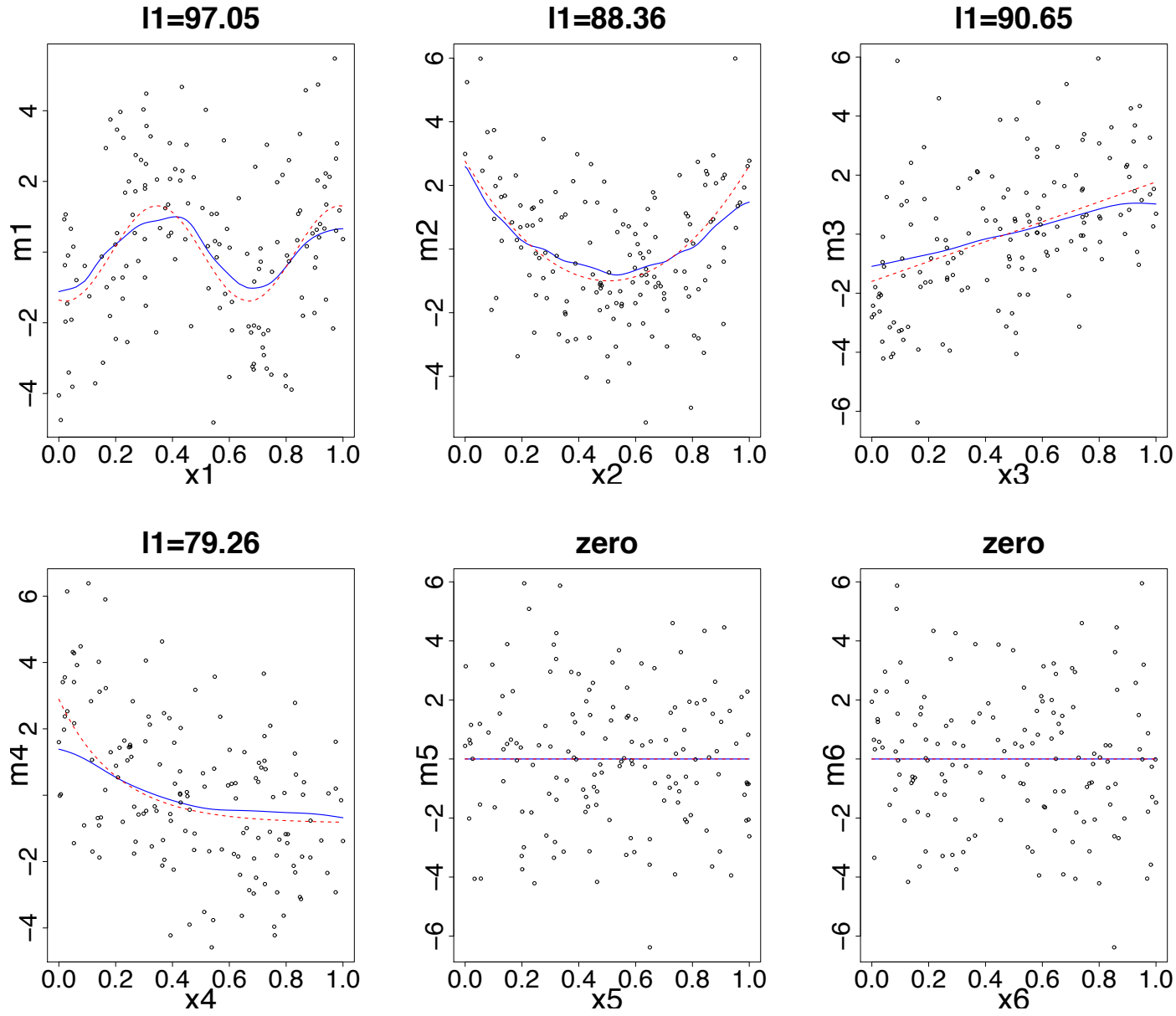
Example: sparse additive models. Suppose that we want to model $y = (y_1, \ldots y_n)$ as a nonparametric function of some input data $z = (z_1, \ldots z_n)$, where $z_i \in \mathbb{R}^d$, $i = 1, \ldots n$ and d is large

For each dimension $j = 1, \ldots d$, we construct a matrix $X_{(j)}$ whose columns contains basis functions (e.g., splines) evaluated over the data $z_1, \ldots z_n$; the coefficients $\beta_{(j)}$ index these basis functions

With the loss

$$h(X\beta) = \|y - X\beta\|_2^2 = \left\| y - \sum_{j=1}^{J} X_{(j)}\beta_{(j)} \right\|_2^2$$

we are fitting a nonparametric function of y on z, and selecting out groups means selecting out relevant dimensions, among $j = 1, \ldots d$



(From Ravikumar et al. (2007), "Sparse additive models")

Algorithms for the group lasso

Consider either the ℓ_2 or ℓ_∞ penalty on groups. Let's walk through our toolset

Primal subgradient method: easy, just slow (as usual)

Primal proximal gradient: this is pretty appealing. Gradient of the smooth part is easy: $X^T \nabla h(X\beta)$, and the prox operator is

$$\text{prox}_t(\beta) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|\beta - z\|_2^2 + \sum_{j=1}^J w_j \|\beta_{(j)}\|_2, \quad \text{or}$$

$$\text{prox}_t(\beta) = \underset{z}{\text{argmin}} \ \frac{1}{2t}\|\beta - z\|_2^2 + \sum_{j=1}^J w_j \|\beta_{(j)}\|_\infty$$

Both of these can be computed in closed-form in $O(n)$ operations, with variants of group-wise shrinkage (check this!)

Primal interior point method: consider ℓ_∞ version. Reparametrize the problem as:

$$\begin{aligned} \min_{\beta,z} \quad & h(X\beta) + \lambda \sum_{j=1}^{J} w_j z_j \\ \text{subject to} \quad & -z_j \leq \beta_{(j)} \leq z_j, \; j = 1, \ldots J \end{aligned}$$

Now the barrier smoothed criterion is

$$F(\beta, z) = t \cdot h(X\beta) + t \cdot \lambda \sum_{j=1}^{J} w_j z_j - \sum_{j=1}^{J} \sum_{\ell=1}^{p_j} \Big(\log(z_j - \beta_{(j),\ell}) + \log(z_j + \beta_{(j),\ell}) \Big)$$

Important consideration: what is the structure of $\nabla^2 F(\beta, z)$?

First consider the log barrier term

$$\phi(\beta, z) = -\sum_{j=1}^{J} \sum_{\ell=1}^{p_j} \Big(\log(z_j - \beta_{(j),\ell}) + \log(z_j + \beta_{(j),\ell}) \Big)$$

Can see that $\nabla^2 \phi(\beta, z)$ is block diagonal, when arranged properly. The other terms in $F(\beta, z)$ contribute only

$$\begin{bmatrix} tX^T \nabla^2 h(X\beta) X & 0 \\ 0 & 0 \end{bmatrix}$$

to the Hessian, and it is $X^T \nabla^2 h(X\beta) X$ that could cause trouble. Would probably only want to use an interior point method here if this was structured (say banded)

Note: ℓ_2 group penalties are more complicated under this approach

Lastly, how about dual algorithms? Exercise: try yourself and see!

Case study: very large group lasso

Suppose we want to fit a very large group lasso, e.g., think about a sparse additive model, in a high dimension d

Interior point methods make no sense, because iterations are much too costly (not enough structure in constructed basis matrix X)

Subgradient method is slow and really not much more cheaper (per iteration) than proximal gradient, so there's no real reason not to use the latter

If n is itself huge, our best bet is stochastic proximal gradient, as

$$X^T \nabla h(X\beta) = \sum_{i=1}^n x_i \cdot \nabla_i h(X\beta)$$

with x_i, $i = 1, \ldots n$ being rows of X. We can apply the prox to $\beta - t \sum_{i \in \mathcal{I}} \nabla_i h(X\beta) x_i$, for random subset $\mathcal{I} \subset \{1, \ldots n\}$

Case study: autoregressive group lasso

Consider an autoregressive setup over time: we regress the current observation against groups of past observations. Roughly this is

$$y_i \approx \sum_{t=1}^{\Delta} \beta_t y_{i-t} + \sum_{t=\Delta+1}^{2\Delta} \beta_t y_{i-t} + \ldots + \sum_{t=(J-1)\Delta+1}^{J\Delta} \beta_t y_{i-t}$$

and with group lasso, we want to select relevant groups

Proximal gradient descent still looks favorable

But we have a banded predictor matrix X (each row composed of past y_i values), and interior point methods look much better than they did before, since the Hessian has structure

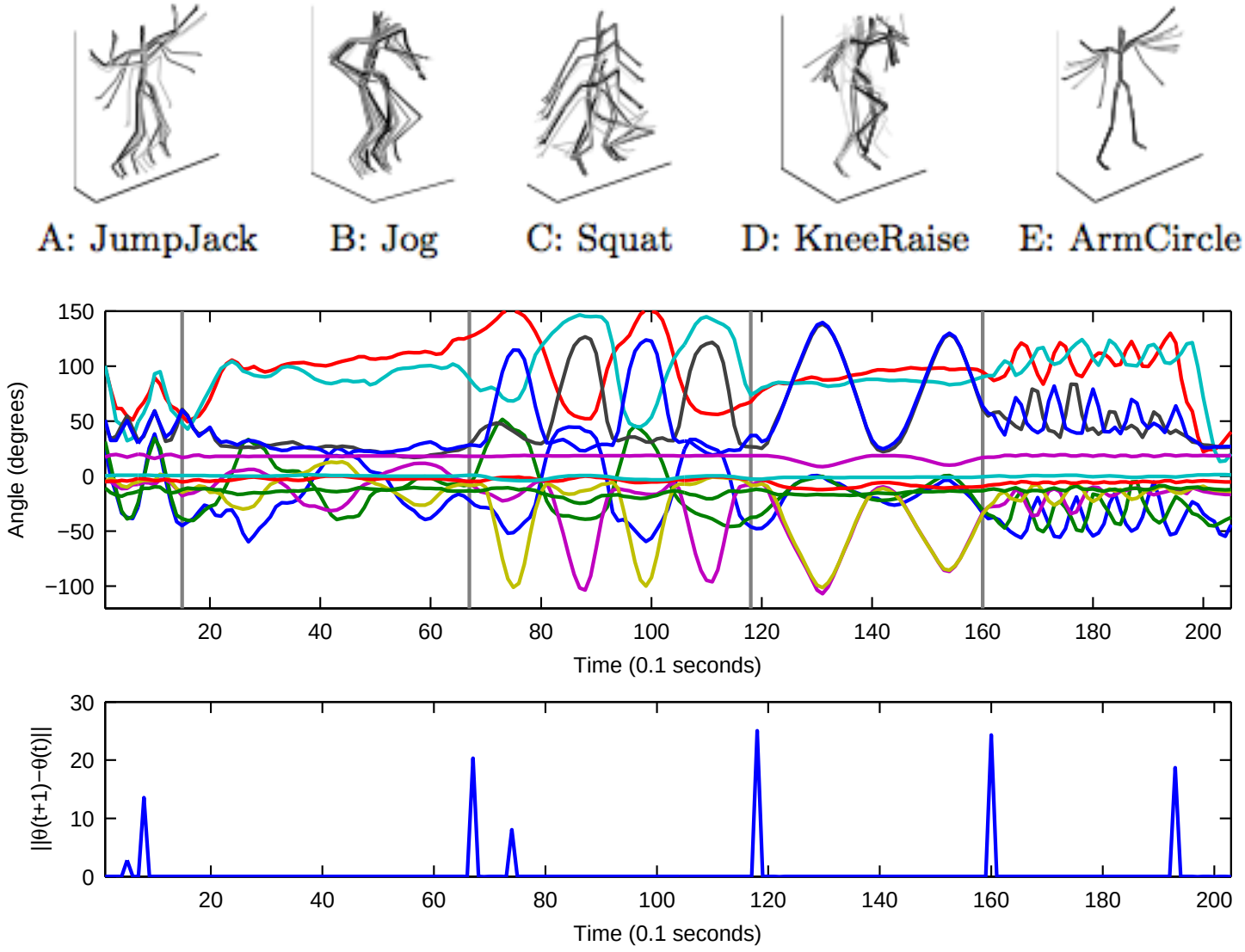
Group fused lasso problems

A marriage of two of the last ideas is the group fused lasso. Again we write $\beta = (\beta_{(1)}, \ldots \beta_{(J)})$, and we penalize differences between subblocks of coeffcients, according to a graph:

$$\min_{\beta=(\beta_{(1)},\ldots\beta_{(J)})\in\mathbb{R}^p} f(\beta) + \lambda \sum_{(i,j)\in E} \|\beta_{(i)} - \beta_{(j)}\|_q$$

- As with the fused lasso, common graph choices are chains, grids, or fully-connected graphs
- As with the group lasso, common norm choices are $q = 2$ or $q = \infty$

Example: segmenting a dynamical system linear model for human motion, across time (next slide)



(From Wytock and Kolter (2014), "Probabilistic segmentation via total variation regularization.")

Example: convex clustering. Write $y = (y_{(1)}, \ldots y_{(n)})$, where each $y_{(i)} \in \mathbb{R}^d$, $i = 1, \ldots n$, and we want to cluster these points. We set $J = n$ groups, choose a loss

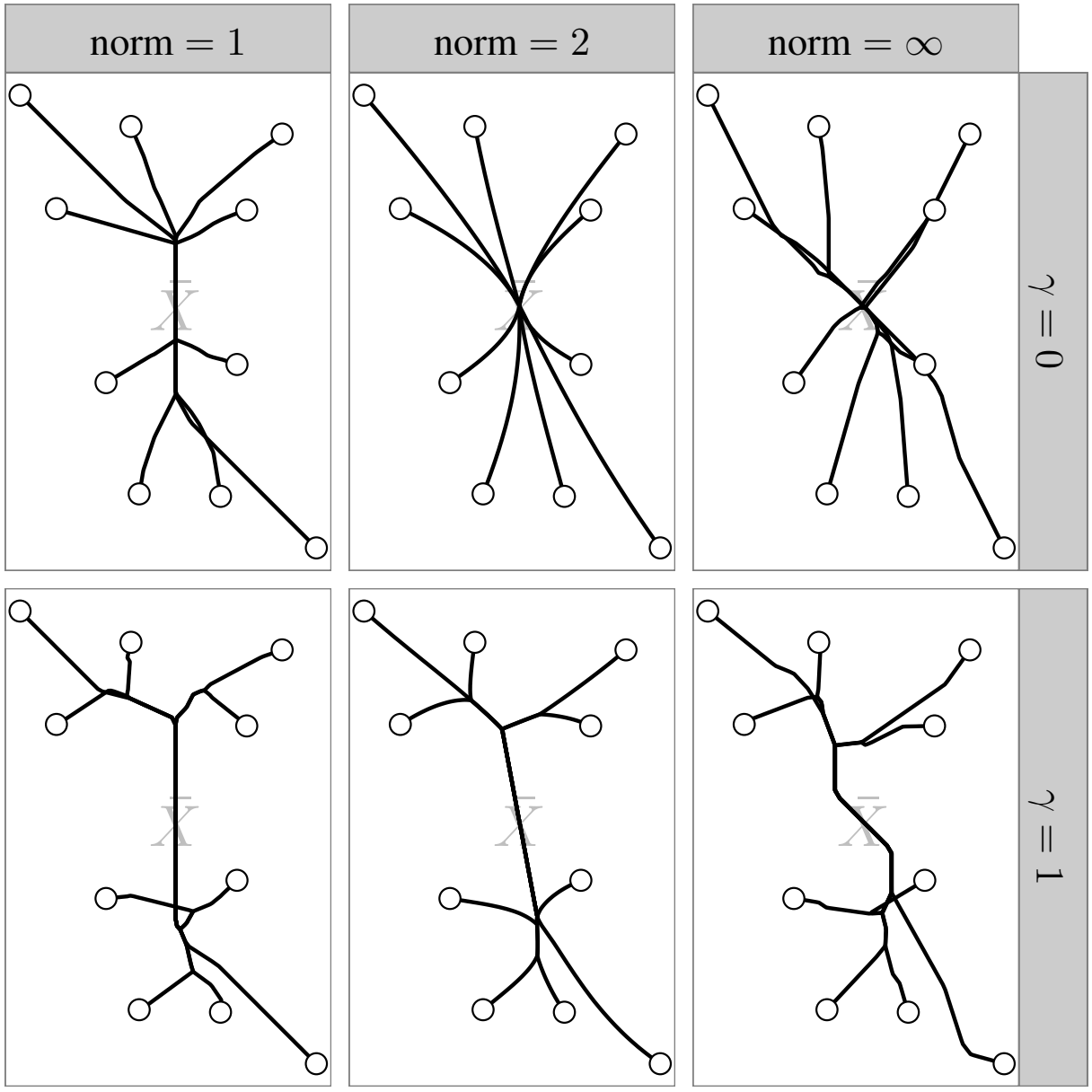
$$f(\beta) = \sum_{i=1}^{n} \|y_{(i)} - \beta_{(i)}\|_2^2$$

and solve a group fused lasso problem with a complete graph that joins each pair $\beta_{(i)}, \beta_{(j)}$, i.e., the penalty term is

$$\sum_{i<j} w_{ij} \|\beta_{(i)} - \beta_{(j)}\|_2 \quad \text{or} \quad \sum_{i<j} w_{ij} \|\beta_{(i)} - \beta_{(j)}\|_\infty$$

Note that this defines a clustering assignment, by looking at which pairs satisfy $\hat{\beta}_{(i)} = \hat{\beta}_{(j)}$ at the solution

(And a common choice is $w_{ij} = \exp(-\gamma \|y_{(i)} - y_{(j)}\|_2^2)$, for all i, j)



(From Hocking et al. (2011), "Clusterpath: an algorithm for clustering using convex fusion penalties")

Algorithms for the group fused lasso

Primal proximal gradient requires evaluating the prox operator

$$\text{prox}_t(\beta) = \underset{z=(z_{(1)},\dots z_{(J)})}{\text{argmin}} \ \frac{1}{2t}\sum_{i=1}^{J}\|\beta_{(i)} - z_{(i)}\|_2^2 + \sum_{(i,j)\in E}\|z_{(i)} - z_{(j)}\|_q$$

which is difficult, especially for general graphs

Let's instead consider the dual, as we did for fused lasso. Define the coefficient matrix $B = [\beta_{(1)}, \dots \beta_{(J)}]$, and note that we have

$$\begin{aligned} \text{Primal} \ &: \ \min_B \ f(B) + \lambda\|DB\|_{1,q} \\ \text{Dual} \ &: \ \min_U \ f^*(-D^T U) \ \text{ subject to } \ \|U\|_{\infty,q*} \le \lambda \end{aligned}$$

where $U = [u_{(1)}, \dots u_{(|E|)}]$

Dual proximal gradient requires a simple projection onto a set of ℓ_{q^*} balls and is thus an appealing approach for moderate accuracy

Interior point methods can be difficult to apply here even when the loss function is simple. E.g., with Gaussian signal approximator loss

$$\begin{aligned} \text{Primal} \quad &: \quad \min_B \ \frac{1}{2}\|B - Y\|_F^2 + \lambda \|DB\|_{1,q} \\ \text{Dual} \quad &: \quad \min_U \ \frac{1}{2}\|D^T U\|_F^2 - \operatorname{tr}(Y^T D^T U) \\ &\qquad \text{subject to} \quad \|U\|_{\infty,q^*} \leq \lambda \end{aligned}$$

The Hessian for dual objective is $I_p \otimes DD^T$ but the log barrier for $q^* = 1$ or $q^* = 2$ complicates this structure

Operator splitting (which we'll discuss soon, with ADMM) provides a general approach for problems of the form

$$f(x) + g(x)$$

possibly subject to constraints, by reducing them to evaluations of the prox operators for f and g. This is very flexible, and applicable to fused lasso, group lasso, and group fused lasso problems

What did we learn from this?

From generalized lasso study (really, these are general principles):

- There is no single best method: performance depends greatly structure of penalty, conjugate of loss, desired accuracy level, sought regularization level
- Duality is your friend: dual approaches offer complementary strengths, move linear transformation from nonsmooth penalty into smooth loss, and strive in different regularization regime
- Regressors complicate duality: presence of predictor variables in the loss complicate dual relationship, but proximal gradient will reduce this to a problem without predictors
- Recognizing easy subproblems: if there is a subproblem that is specialized and efficiently solvable, then work around it
- Limited memory at scale: for large problems, active set and/or stochastic methods may be only option

Your toolbox will only get bigger

There are still many algorithms to learn. E.g., for the problems we considered, depending on the setting, we might instead use:

- Alternating direction method of multipliers
- (Block) coordinate descent methods
- Projected Newton methods
- Exact path-following methods

Remember, you don't have to find/design the perfect optimization algorithm, just one that will work well for your problem!

For completeness, recall tools like `cvx`[4] and `tfocs`[5], if performance is not a concern, or you don't want to expend programming effort

[4] Grant and Boyd (2008), "Graph implementations for nonsmooth convex problems", `http://cvxr.com/cvx/`

[5] Beckter et al. (2011), "Templates for convex cone problems with applications to sparse signal recovery", `http://cvxr.com/tfocs/`

Implementation tips

Implementation details are not typically the focus of optimization courses, because in a sense, implementation skills are under-valued

Still an extremely important part of optimization. Considerations:

- Speed
- Robustness
- Simplicity
- Portability

First point doesn't need to be explained. Robustness refers to the stability of implementation across various use cases. E.g., suppose our graph fused lasso solver supported edge weights. It performs well when weights are all close to uniform, but what happens under highly nonuniform weights? Huge and small weights, mixed?

Simplicity and portability are often ignored. An implementation with 20K lines of code may run fast, but what happens when a bug pops up? What happens when you pass it on to a friend? Tips:

- A constant-factor speedup is probably not worth a much more complicated implementation, especially if the latter is hard to maintain, hard to extend
- Speed of convergence to higher accuracy may be worth a loss of simplicity
- Write the code bulk in a low-level language (like C or C++), so that it can port to R, Matlab, Python, Julia, etc.
- Don't re-implement standard routines, this is often not worth your time, and prone to bugs. Especially true for numerical linear algebra routines!