10-725/36-725: Convex Optimization **Spring 2015**

Lecture 19: March 30

Lecturer: Ryan Tibshirani *Scribes: Hao Zhang (hzhang2)*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

In this section, we conduct a case study on the **Generalized Lasso Problems**.

19.1 Review

So far we have learned many optimization approaches, including the first-order methods, the Newton/quasi-Newton methods and the interior point methods. Given the number of available optimization tools, it may seem overwhelming to choose a method in practice. A fair question naturally emerges: how to know what to use when?

However, it's not possible to give a complete answer for this question. To answer this, we need to investigate the following parts of the whole optimization problem:

- Assumptions on criterion function
- Assumptions on constraint functions/set
- Ease of implementation
- Cost of each iteration
- Number of iterations needed

Also, in other settings, we need to consider issues such as parallelization, data storage and statistical interplay. In this lecture, we take the generalized lasso problems as an example, and walk through some of the high-level reasoning.

The generalized lasso problem is defined as follows:

$$\min_{\beta} f(\beta) + \lambda \|D\beta\|_1 \tag{19.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth, convex function and $D \in \mathbb{R}^{m \times n}$ is a penalty matrix. The usual lasso, $D = I$, encodes sparsity in solution $\hat{\beta}$, while the generalized lasso encodes sparsity in

$$D\hat{\beta} = \begin{bmatrix} D_1\hat{\beta} \\ \vdots \\ D_m\hat{\beta} \end{bmatrix} \tag{19.2}$$

where $D_i, i = 1, \ldots, m$ are the rows of D.

19.2 Notable examples

19.2.1 Fused lasso or total variation denoising, 1d

A special case of generalized lasso problems is the fused lasso or total variation denoising in $1d$, where

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \tag{19.3}$$

Therefore, the lasso term can be rewritten as

$$\|D\beta\|_1 = \sum_{i=1}^{n-1} |\beta_i - \beta_{i+1}| \tag{19.4}$$

Hence, we obtain sparsity in adjacent differences $\hat{\beta}_i - \hat{\beta}_i$, i.e., we obtain $\hat{\beta}_i = \hat{\beta}_i$ at many locations i. Specifically, the solution $\hat{\beta}$ appears *piecewise constant*.

Higher order polynomials fits are also possible, which leads to the *trend filtering* methods in $1d$. For example, when we choose

$$D = \begin{bmatrix} 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{19.5}$$

We will get:

$$\|D\beta\|_1 = \sum_{i=1}^{n-2} |\beta_i - 2\beta_{i+1} + \beta_{i+2}| \tag{19.6}$$

which gives *piecewise linear* solution $\hat{\beta}$. When we choose:

$$D = \begin{bmatrix} -1 & 3 & -3 & 1 & \cdots & 0 \\ 0 & 1 & -3 & 3 & \cdots & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{19.7}$$

we will get:

$$\|D\beta\|_1 = \sum_{i=1}^{n-3} |\beta_i - 3\beta_{i+1} + 3\beta_{i+2} - \beta_{i+3}| \tag{19.8}$$

which gives us a *piecewise quadratic* solution $\hat{\beta}$.

19.2.2 Fused lasso or total variation denoising, graphs

Another special case of generalized lasso is the 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 ℓ-th row with all zeros except the i-th column as -1 and the j-th column as 1, which consequently leads to

$$\|D\beta\|_1 = \sum_{(i,j)\in E} |\beta_i - \beta_j| \tag{19.9}$$

At the solution $\hat{\beta}$, 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.

19.2.3 Problems with a big dense D

In some problems we encounter a big dense operator D, whose structure might as well be considered arbitrary. For example, we might have collected measurements that we know should lie mostly orthogonal to the desired estimate $\hat{\beta}$, and we stack these along the rows of D, known as analyzing operator in this setup.

The equality-constrained lasso problems follows this case:

$$\min_{\beta} f(\beta) + \lambda\|\beta\|_1 \text{ subject to } A\beta = 0 \tag{19.10}$$

Note that the above problem can be reparametrized by letting $D \in \mathbb{R}^{n\times r}$ have columns to span $null(A)$. Then $A\beta = 0 \Leftrightarrow \beta = D\theta$ for some $\theta \in \mathbb{R}^r$, and the above is equivalent to

$$\min_{\theta} f(D\theta) + \lambda\|D\theta\|_1 \tag{19.11}$$

which follows the form of generalized lasso problem.

19.3 Algorithmic considerations

Now, let's go through all methods we have learned, to figure out how to solve the generalized lasso problem:

$$\min_{\beta} +\lambda\|D\beta\|_1 \tag{19.12}$$

19.3.1 Subgraident method

The subgradient of the above criterion is

$$g = \nabla f(\beta) + \lambda D^T\gamma \tag{19.13}$$

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

$$\gamma_i \in \begin{cases} \{\text{sign}((D\beta)_i)\} & \text{if } (D\beta)_i \neq 0, i = 1,\ldots,m \\ [-1,1] & \text{if } (D\beta)_i = 0, i = 1,\ldots,m \end{cases} \tag{19.14}$$

The advantage of subgradient method is that g is easy to compute: provided ∇f, if $S = supp(D\beta)$, then we let

$$g = \nabla f(\beta) + \lambda \sum_{i\in S} \text{sign}((D\beta)_i) D_i \tag{19.15}$$

However, the shortcoming of subgradient method is that convergence is slow.

19.3.2 Proximal gradient descent

The prox operator of the generalized lasso problem is:

$$\text{prox}_t(\beta) = \arg\min_z \frac{1}{2t}\|\beta - z\|_2^2 + \lambda\|Dz\|_1 \tag{19.16}$$

However, this is not easy for a generic D. Actually, this is a highly nontrivial optimization problem, even when D is structured.

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

Let's investigate the dual problem.

19.3.3 Generalized lasso dual

Our primal and dual problems are:

$$\begin{aligned} \text{Primal: } & \min_{\beta} f(\beta) + \lambda \|D\beta\|_1 \\ \text{Dual: } & \min_{u} f^*(-D^T u) \text{ subject to } \|u\|_\infty \leq \lambda \end{aligned} \tag{19.17}$$

where f^* is the conjugate of f. The primal and dual solutions $\hat{\beta}$ and $\hat{u}$ are linked by KKT conditions:

$$\nabla f(\hat{\beta}) + D^T \hat{u} = 0 \tag{19.18}$$

and

$$\hat{u}_i = \begin{cases} \{\lambda\} & \text{if } (D\hat{\beta})_i > 0 \\ \{-\lambda\} & \text{if } (D\hat{\beta})_i < 0, i = 1, \ldots, m \\ [-\lambda, \lambda] & \text{if } (D\hat{\beta})_i = 0 \end{cases} \tag{19.19}$$

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

Now let's think about solving dual problem

$$\min_{u} f^*(-D^T u) \text{ subject to } \|u\|_\infty \leq \lambda \tag{19.20}$$

19.3.3.1 Proximal gradient descent

The prox operator for the dual problem is

$$\text{prox}_t(u) = \arg\min_{z} \frac{1}{2t}\|u - z\|_2^2 \text{ subject to } \|z\|_\infty \leq \lambda \tag{19.21}$$

which seems to be much easy to compute. Actually, this is projection onto a box $[-\lambda, \lambda]^m$, i.e., prox returns $\hat{z}$ with

$$\hat{z}_i = \begin{cases} \lambda & \text{if } u_i > \lambda \\ -\lambda & \text{if } u_i < -\lambda, i = 1, \ldots, m \\ u_i & \text{if } u_i \in [-\lambda, \lambda] \end{cases} \tag{19.22}$$

19.3.3.2 Interior point method

We can rewrite the dual problem as:

$$\min_{u} f^*(-D^T u) \text{ subject to } -\lambda \leq u_i \leq \lambda, i = 1, \ldots, m. \tag{19.23}$$

Note that these are just linear constraints, so we can easily form log barrier as in

$$\min_{u} t f^*(-D^T u) + \phi(u) \tag{19.24}$$

with

$$\phi(u) = -\sum_{i=1}^{m}(\log(\lambda - u_i) + \log(u_i + \lambda)) \tag{19.25}$$

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

Now, let consider the efficiency of Newton Updates. Define the barrier-smoothed dual criterion function

$$F(u) = tf^*(-D^T u) + \phi(u) \tag{19.26}$$

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

$$\begin{aligned} g &= \nabla F(u) = -tD(\nabla f^*(-D^T u)) + \nabla\phi(u) \\ H &= \nabla^2 F(u) = tD(\nabla^2 f^*(-D^T u))D^T + \nabla^2\phi(u) \end{aligned} \tag{19.27}$$

Inspecting the second equation: for the first term, if Hessian off the loss term $\nabla^2 f^(v)$ is structured, and D is structured, then often $D\nabla^2 f^*(v)D^T$ is structured. For the second term, Hessian of log barrier term $\nabla^2\phi(u)$ is diagonal.

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

19.3.4 Conclusion

Putting all of above together, we conclude:

- Primal subgradient method: iterations are cheap (we sum up rows of D over active set S), but convergence is slow
- Primal proximal gradient: the evaluation of the prox operator is very expensive, while convergence is medium
- Dual proximal graident: iterations involve projecting onto a box, so very cheap, convergence is medium
- Dual interior point method: iterations involve a solving linear $Hx = g$ system which may or may not be expensive, convergence is rapid

19.4 Back to examples

19.4.1 Linear tread filtering

Suppose that we are studying the linear trend filtering problem, so

$$D = \begin{bmatrix} 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{19.28}$$

The loss is either Gaussian or logistic. Suppose further that we desire solution at a hight level of accuracy. What algorithm should we use? Primal subgradient and primal proximal gradient are out because they are 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.

Dual proximal gradient descent admits very efficient iterations, as it just projects $u + tD\nabla f^*(-D^T u)$ onto a box, repeatedly. However, it takes far too long to converge to high accuracy: even more so than usual, because it suffers from poor conditioning of D. Importantly, $\nabla^2 f^*(v)$ is a diagonal matrix in both the Gaussian and logistic cases. Therefore, the Newton steps in a dual interior point method involve solving a linear system $Hx = g$ in

$$H = DA(u)D^T + B(u) \tag{19.29}$$

where $A(u), B(u)$ are both diagonal. This is a banded matrix, and so these systems can be solved very efficiently, in $O(n)$ 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.

19.4.2 Graph fused lasso

The same story holds when D is the fused lasso operator on an arbitrary graph:

- Primal subgradient is slow, primal prox is intractable
- Dual prox is cheap to iterate, but slow to converge
- Dual interior point method solves structured linear systems, so its iterations are efficient, and is preferred

Dual interior point method repeatedly solves $Hx = g$, where

$$H = DA(u)D^T + B(u) \tag{19.30}$$

and $A(u), B(u)$ are both diagonal. This is no longer banded, but it is highly structured: D is the edge incidence matrix of the graph, and $L = D^T D$ is the graph Laplacian.

19.4.3 Regression problems

Consider the same D, and the same Gaussian and logistic losses, but with regressors or predictors $x_i \in \mathbb{R}^p, i = 1, \ldots, n$. For instance, if the predictors are connected over a graph, and D is the graph fused lasso matrix, then the estimated coefficients will be constant over regions of the graph.

Assume that the predictors values are arbitrary, everything in the dual is more complicated now. Denote $X \in \mathbb{R}^{n \times p}$ as the predictor matrix (rows $x_i, i = 1, \ldots, n$), and $f(\beta) = h(X\beta)$ as the loss. Our problems are:

$$\begin{aligned} \text{Primal: } & \min_\beta h(X\beta) + \lambda \|D\beta\|_1 \\ \text{Dual: } & \min_{u,v} h^*(v), \text{ subject to } X^T v + D^T u = 0, \|u\|_\infty \leq \lambda \end{aligned} \tag{19.31}$$

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

$$\begin{aligned} & \nabla h(X\hat{\beta}) - \hat{v} = 0 \text{ or equivalently} \\ & X^T \nabla h(X\hat{\beta}) + D^T \hat{u} = 0 \end{aligned} \tag{19.32}$$

Computing $\hat{\beta}$ from the dual requires solving a linear system in X, very expensive for generic X.

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

$$\text{prox}_t(u, v) = \underset{X^T w + D^T z = 0}{\arg\min} \frac{1}{2t}\|u - z\|_2^2 + \frac{1}{2t}\|v - w\|_2^2 + \|u\|_\infty \tag{19.33}$$

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

$$X^T v + D^T u = 0 \tag{19.34}$$

must be maintained. To do 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. However, 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:

$$\text{prox}_t(\beta) = \arg\min_z \frac{1}{2t}\|\beta - z\|_2^2 + \lambda\|Dz\|_1 \tag{19.35}$$

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.

19.4.4 1d fused lasso

Now we turn to the special case of the fused lasso in 1d, recall:

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \tag{19.36}$$

The prox function in the primal is

$$\text{prox}_t(\beta) = \arg\min_z \frac{1}{2t}\|\beta - z\|_2^2 + \lambda \sum_{i=1}^n |z_i - z_{i+1}| \tag{19.37}$$

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

Let's write the primal and dual problem as follows:

$$\begin{aligned} \text{Primal: } & \min_\beta f(\beta) + \lambda\|D\beta\|_1 \\ \text{Dual: } & \min_u f^*(-D^T u) \text{ subject to } \|u\|_\infty \leq \lambda \end{aligned} \tag{19.38}$$

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

Hence, generally, for a larger value λ, it will easier for primal algorithms while for a small λ, dual algorithms will be preferred.

19.4.5 Big dense D

Consider a problem with a dense, generic D, e.g., as an observed analyzing operator, or stemming from an equality-constrained lasso problem.

Primal prox is intractable, and a dual interior point method has extremely 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 and structured, 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 so massive that even fitting it in memory is a burden. Depending on f and its gradient, primal subgradient method might be the only feasible algorithm; recall the subgradient calculation:

$$g = \nabla f(\beta) + \lambda \sum_{i \in S} \text{sign}((D\beta)_i) D_i \tag{19.39}$$

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.

19.4.6 Conclusion

For generalized lasso study,

- 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

19.5 Implementation tips

Implementation details are critical for optimization methods. Generally, we need to consider the following four issues: 1) speed 2) Robustness 3) Simplicity 4) Portability.

Speed doesn't need to be explained. Robustness refers to the stability of implementation across various use cases. Simplicity and portability are often ignored. An implementation with $20K$ lines of code may run fast, but it's hard to debug or be reused by others.

The following are some tips for a good implementation:

- 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.