Proximal Newton Method

Ryan Tibshirani
Convex Optimization 10-725/36-725

Last time: primal-dual interior-point method

Given the problem

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & h_i(x) \le 0, \ \ i = 1, \dots m \\ & Ax = b \end{aligned}$$

where f, h_i, $i = 1, \dots m$ are convex and smooth, we consider the perturbed KKT conditions

$$\begin{gathered} \nabla f(x) + \sum_{i=1}^m u_i \nabla h_i(x) + A^T v = 0 \\ u_i h_i(x) = -1/t, \quad i = 1, \dots m \\ h_i(x) \le 0, \quad i = 1, \dots m, \quad Ax = b \\ u_i \ge 0, \quad i = 1, \dots m \end{gathered}$$

where we have modified complementary slackness

Let's express these conditions as $r(x, u, v) = 0$, where

$$r(x, u, v) = \begin{pmatrix} \nabla f(x) + Dh(x)^T u + A^T v \\ -\mathrm{diag}(u)h(x) - 1/t \\ Ax - b \end{pmatrix}$$

and

$$h(x) = \begin{pmatrix} h_1(x) \\ \dots \\ h_m(x) \end{pmatrix}, \quad Dh(x) = \begin{bmatrix} \nabla h_1(x)^T \\ \dots \\ \nabla h_m(x)^T \end{bmatrix}$$

In the primal-dual interior-point method, with $y = (x, u, v)$ as the current iterate, and $\Delta y = (\Delta x, \Delta u, \Delta v)$ the update direction, we form a Newton step for the above nonlinear system

$$0 = r(y + \Delta y) \approx r(y) + Dr(y)\Delta y$$

and solve for Δy, to get our update direction. Conclusion: similar properties as the barrier method, but often faster

Outline

Today:

- Proximal gradient recap
- Proximal Newton method
- Backtracking line search
- Convergence analysis
- Notable examples

Reminder: proximal gradient descent

Recall that proximal gradient descent operates on a problem

$$\min_x \; g(x) + h(x)$$

where g is convex, smooth and h is convex, "simple". We choose initial $x^{(0)}$ and repeat for $k = 1, 2, 3, \ldots$

$$x^{(k)} = \mathrm{prox}_{t_k}\big(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\big)$$

where $\mathrm{prox}_t(\cdot)$ is the proximal operator associated with h,

$$\mathrm{prox}_t(x) = \underset{z}{\mathrm{argmin}} \; \frac{1}{2t}\|x - z\|_2^2 + h(z)$$

- Difficulty of iterations is in applying prox, which only depends on h (assuming that ∇g is computable)
- Proximal gradient descent enjoys same convergence rate as its fully smooth version, hence useful when prox is efficient

Recall the motivation for proximal gradient: iteratively minimize a quadratic expansion in g, plus original h

$$\begin{aligned} x^+ &= \operatorname*{argmin}_z \; \frac{1}{2t}\|x - t\nabla g(x) - z\|_2^2 + h(z) \\ &= \operatorname*{argmin}_z \; \nabla g(x)^T(z-x) + \frac{1}{2t}\|z-x\|_2^2 + h(z) \end{aligned}$$

The quadratic approximation here uses Hessian equal to (a scaled version of) the identity $\frac{1}{t}I$

A fundamental difference between gradient descent and Newton's method was that the latter also iteratively minimized quadratic approximations, but these used the local Hessian of the function in question

So what happens if we replace $\frac{1}{t}I$ in the above with $\nabla^2 g(x)$?

Proximal Newton method

This leads us to the proximal Newton method. Now we must define

$$\operatorname{prox}_H(x) = \underset{z}{\operatorname{argmin}}\ \frac{1}{2}\|x - z\|_H^2 + h(z)$$

where $\|x\|_H^2 = x^T H x$ defines a norm, given a matrix $H \succ 0$. This is a scaled proximal mapping. With $H = \frac{1}{t} I$, we get back previous definition

Starting with $x^{(0)}$, we repeat for $k = 1, 2, 3, \ldots$

$$\begin{aligned} y^{(k)} &= \operatorname{prox}_{H_{k-1}}\big(x^{(k-1)} - H_{k-1}^{-1} \nabla g(x^{(k-1)})\big) \\ x^{(k)} &= x^{(k-1)} + t_k (y^{(k)} - x^{(k-1)}) \end{aligned}$$

Here $H_{k-1} = \nabla^2 g(x^{(k-1)})$, and t_k is a step size, which we choose by backtracking line search (as in usual Newton)

Let's check this is indeed minimizing a quadratic approximation of g, plus h:

$$\begin{aligned} y &= \underset{z}{\operatorname{argmin}}\ \frac{1}{2}\|x - H^{-1}\nabla g(x) - z\|_H^2 + h(z) \\ &= \underset{z}{\operatorname{argmin}}\ \nabla g(x)^T(z-x) + \frac{1}{2}(z-x)^T H(z-x) + h(z) \end{aligned}$$

Notes:

- When $h(z) = 0$, we get back the usual Newton update
- For $H \succ 0$, can check that $\operatorname{prox}_H(\cdot)$ retains many of the nice properties of (unscaled) proximal mappings (Lee et al. 2012). E.g., it is well-defined, since the minimizer is unique
- Difficulty of prox has mostly to do with h, however, now the Hessian of g also plays a role—the structure of this Hessian H can make a difference

Backtracking line search

As with Newton's method in fully smooth problems, pure step sizes $t_k = 1$, $k = 1, 2, 3, \ldots$ need not converge. We need to apply, say, backtracking line search. Set parameters $0 < \alpha \le 1/2$, $0 < \beta < 1$, and let

$$v = \mathrm{prox}_H\big(x - H^{-1}\nabla g(x)\big) - x$$

be the proximal Newton direction at a given iteration. Start with $t = 1$, and while

$$f(x + tv) > f(x) + \alpha t \nabla g(x)^T v + \alpha\big(h(x + td) - h(x)\big)$$

we shrink $t = \beta t$. (Here $f = g + h$)

Note: this scheme is actually of a different spirit than the one we studied for proximal gradient descent, as it avoids recomputing the prox at each inner backtracking iteration

Wait ... does this even make sense?

Let's back up. One of the main drivers behind proximal gradient descent is that we can transform the problem

$$\min_x \; g(x) + h(x)$$

into a sequence of problems where $g(x)$ is essentially replaced by $\|b - x\|_2^2$. This can be easy, but it depends on h

Now we have transformed into a sequence of problems where $g(x)$ is essentially replaced by $b^T x + x^T A x$. For dense A, this seems like it would rarely be easy, regardless of h ... That is, evaluating the scaled prox

$$\underset{z}{\operatorname{argmin}} \; \nabla g(x)^T (z - x) + \frac{1}{2}(z - x)^T H (z - x) + h(z)$$

seems to be not an easy subproblem, for a generic Hessian H ...

All this is true, and the prox operator in proximal Newton is usually extremely expensive, and one that we solve with an optimization subroutine (e.g., for $h(x) = \|x\|_1$, prox is standard lasso problem)

What we should hope for: the convergence rate of prox Newton, in terms of the number of iterations (prox evaluations) needed, is like the usual Newton method. This ends up being true

Therefore, if we have a decent inner solver for the prox step, it can be quite efficient to use proximal Newton (e.g., this is true with ℓ_1 regularized generalized linear models). But in general, prox Newton is not to be applied without care

(Well-known implementations using prox Newton: glmnet, QUIC; more on this later)

Convergence analysis

Following Lee et al. (2012), assume that $f = g + h$, where g, h are convex and g is twice smooth. Assume further:

- $mI \preceq \nabla^2 g \preceq LI$, and $\nabla^2 g$ Lipschitz with parameter M
- $\mathrm{prox}_H(\cdot)$ is exactly evaluable

Theorem: Proximal Newton method with backtracking line search satisfies converges globally. Furthermore, for all $k \geq k_0$:

$$\|x^{(k)} - x^\star\|_2 \leq \frac{M}{2m} \|x^{(k-1)} - x^\star\|_2^2$$

Recall that this is called local quadratic convergence. After some point, to get within $f(x^{(k)}) - f^\star \leq \epsilon$, we require $O(\log\log(1/\epsilon))$ iterations. Note: each iteration uses scaled prox evaluation!

Proof sketch

- To prove global convergence, they show that at any step, the backtracking exit condition will be satisfied by

$$t \le \min\left\{1, \frac{2m}{L}(1-\alpha)\right\}$$

Use this to show that the update direction converges to zero, which can only happen at the global minimum

- To prove local quadratic convergence, they show that for large enough k, the pure step $t = 1$ eventually satisfies backtracking exit condition. Therefore

$$\begin{aligned}
&\|x^+ - x^\star\|_2 \le \frac{1}{\sqrt{m}}\|x^+ - x^\star\|_H \\
&\quad \le \left\|\mathrm{prox}_H\big(x - H^{-1}\nabla g(x)\big) - \mathrm{prox}_H\big(x^\star - H^{-1}\nabla g(x^\star)\big)\right\|_H \\
&\quad \le \frac{M}{2m}\|x - x^\star\|_2^2
\end{aligned}$$

Glmnet and QUIC

Two notable examples of proximal Newton methods:

- glmnet (Friedman et al. 2009): applies proximal Newton to ℓ_1 regularized generalized linear models, inner probs solved using coordinate descent
- QUIC (Hsiesh et al. 2011): applies proximal Newton to solve graphical lasso problem, uses factorization tricks, inner probs use coordinate descent

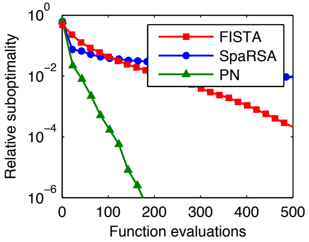
Both of these implementations are very widely used for their own purposes. At the proper scale, these are close to state-of-the-art

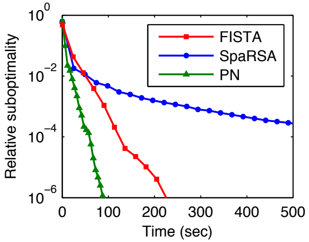
General note: proximal Newton method will use far less evaluations of (gradient of) g than proximal gradient. When these evaluations are expensive, proximal Newton can win

Example: lasso logistic regression

Example from Lee et al. (2012): ℓ_1 regularized logistic regression, FISTA (accelerated prox grad) versus spaRSA (spectral projected gradient method) versus PN (prox Newton)

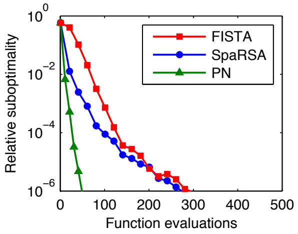
Problem with $n = 5000$, $p = 6000$, and a dense feature matrix X

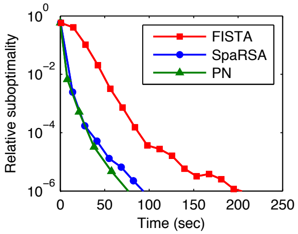




Here g and ∇g require expensive $\exp$ or $\log$ evaluations; dominates computational cost

Now problem with $n = 542,000$, $p = 47,000$, and sparse matrix X



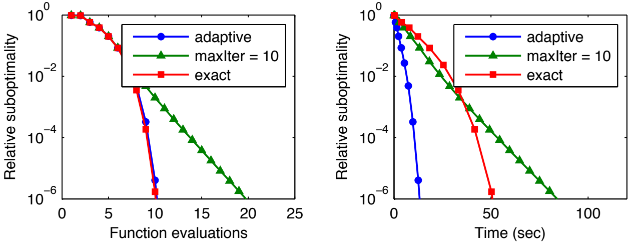


Here g and ∇g require expensive $\exp$ or $\log$ evaluations, but these make up less of total cost, since X is sparse

Inexact prox evaluations

An important note: with proximal Newton, we essentially always perform inexact prox evaluations (not so with proximal gradient)

Example from Lee et al. (2012): graphical lasso estimation, three stopping rules for inner optimizations. Here $n = 72$ and $p = 1255$



Conclusion is that 10 inner iterations is not enough to ensure fast (quadratic convergence), but their adaptive stopping rule is

For usual (smooth) Newton method, inner problem is to minimize $\tilde{g}_{k-1}(z)$ quadratic approximation to g about $x^{(k-1)}$. Stopping rules based on

$$\|\nabla \tilde{g}_{k-1}(z)\|_2 \le \eta_k \|\nabla g(x^{(k-1)})\|_2$$

for a specifically chosen "forcing" sequence η_k, $k = 1, 2, 3, \ldots$

For proximal Newton, Lee et al. (2012) advocate the analogy that uses generalized gradients in place of gradients

$$\|G_{\tilde{f}_{k-1}/M}(z)\|_2 \le \eta_k \|G_{f/M}(x^{(k-1)})\|_2$$

where $\tilde{f}_{k-1} = \tilde{g}_{k-1} + h$, and recall that $m \preceq \nabla^2 g \preceq MI$. Setting

$$\eta_k = \left\{ \frac{m}{2}, \frac{\|G_{\tilde{f}_{k-2}/M}(x^{(k-1)}) - G_{f/M}(x^{(k-1)})\|_2}{\|G_{f/M}(x^{(k-2)})\|_2} \right\}$$

they prove that inexact proximal Newton has local superlinear rate

Proximal quasi-Newton methods

For large problems, computing the Hessian is prohibitive. Proximal quasi-Newton methods avoid exactly forming $H_{k-1} = \nabla g(x^{(k-1)})$ at each step

- Lee et al. (2012) propose iteratively updating H_{k-1} at each step using BFGS-type rules. They show very strong empirical performance, and prove local superlinear convergence
- Tseng and Yun (2009) consider smooth plus block separable problems, and recommend approximating the Hessian in a blockwise fashion, combined with block coordinate descent. This can be very helpful because only small Hessians are ever needed. They prove linear convergence

Note that quasi-Newton methods can not only be helpful when the Hessian is expensive, but also when it is ill-conditioned: singular or close to singular

Proximal Newton versus Tseng and Yun's method

It is interesting to compare Proximal Newton for the problem

$$\min_x \; g(x) + h(x)$$

where $h(x) = \sum_{b=1}^B h_b(x_b)$ separates over B blocks of coordinates, to Tseng and Yun (2009). Their method: block proximal Newton (they call it coordinate gradient descent, bad name!)

The distinction is: perform a quad approximation first, or second?

- Proximal Newton method replaces $g(x+\Delta)$ with $\tilde{g}(x+\Delta) = \nabla g(x)^T \Delta + \frac{1}{2}\Delta^T H \Delta$, and minimizes $\tilde{g}(x+\Delta) + h(x+\Delta)$ to find update Δ. Can find Δ with block coordinate descent
- Tseng and Yun iterate, for each block $b = 1, \ldots B$, replacing smooth part with $\tilde{g}_b(x_b + \Delta_b) = \nabla_b g(x)^T \Delta_b + \frac{1}{2}\Delta_b^T H_b \Delta_b$, and minimize $\tilde{g}_b(x_b+\Delta_b) + h_b(x_b+\Delta_b)$ to find update Δ_b for block b

What's wrong with projected Newton?

Suppose that $h = 1_C(x)$, the indicator function of a convex set C. I.e., consider the problem

$$\min_x \; g(x) \;\; \text{subject to} \;\; C$$

Recall that proximal gradient here reduces to projected gradient. What about proximal Newton? Updates are

$$\begin{aligned} y &= \operatorname*{argmin}_{z \in C} \; \frac{1}{2} \|x - H^{-1}\nabla g(x) - z\|_H^2 \\ &= \operatorname*{argmin}_{z \in C} \; \nabla g(x)^T (z - x) + \frac{1}{2}(z - x)^T H (z - x) \end{aligned}$$

Note when $H = I$ this a projection of $x - \nabla g(x)$ onto C, but this is not a projection in general! In fact, it is much more complicated. Hence, projected Newton does not generally follow from proximal Newton ... we will cover a way to fix this during advanced topics

References

- J. Friedman and T. Hastie and R. Tibshirani (2009), "Regularization paths for generalized linear models via coordinate descent"
- C.J. Hsiesh and M.A. Sustik and I. Dhillon and P. Ravikumar (2011), "Sparse inverse covariance matrix estimation using quadratic approximation"
- M. Patriksson (1998), "Cost approximation: a unified framework of descent algorithms for nonlinear programs"
- J. Lee and Y. Sun and M. Saunders (2012), "Proximal Newton-type methods for minimizing composite functions"
- P. Tseng and S. Yun (2009), "A coordinate gradient descent method for nonsmooth separable minimization"