Proximal Newton Method

Zico Kolter
(notes by Ryan Tibshirani)
Convex Optimization 10-725

Last time: quasi-Newton methods

Consider the problem

$$\min_x \ f(x)$$

with f convex, twice differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$. Generic form of quasi-Newton method: start with $x^{(0)} \in \mathbb{R}^n$, and repeat:

$$x^{(k)} = x^{(k-1)} - t_k C^{(k-1)} x^{(k-1)}, \quad k = 1, 2, 3, \ldots$$

where $C^{(k-1)} \approx (\nabla^2 f(x^{(k-1)}))^{-1}$, an approximation to the inverse Hessian at $x^{(k-1)}$. Step sizes chosen by backtracking. Key: $C^{(0)}$ is easily computed, and $C^{(k-1)}$ is easily updated from $C^{(k-2)}$, $k \geq 2$

- SR1: rank 1 update for Hessian, use SM for inverse Hessian
- DFP: rank 2 update for inverse Hessian, use SM for Hessian
- BFGS: reverse roles of Hessian and inverse Hessian in DFP
- LBFGS: limited memory version of BFGS, very popular

Outline

Today:

- Proximal Newton method
- Backtracking line search
- Convergence analysis
- Notable examples
- Projected Newton method

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 repeat

$$x^{(k)} = \text{prox}_{t_k}\big(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\big), \quad k = 1, 2, 3, \ldots$$

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

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

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

Recall motivation for prox gradient: iteratively minimize quadratic expansion in g, plus original h

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

Quadratic approximation uses $\frac{1}{t}I$ (spherical curvature), as in pure gradient descent when $h = 0$

A fundamental difference between gradient descent and Newton's method: latter also iteratively minimizes quadratic approximations, but these use 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 proximal Newton method: we repeat

$$
\begin{aligned}
v^{(k)} &= \operatorname*{argmin}_v \ \nabla g(x^{(k-1)})^T v + \frac{1}{2} v^T H^{(k-1)} v + h(x^{(k-1)} + v) \\
x^{(k)} &= x^{(k-1)} + t_k v^{(k)}
\end{aligned}
$$

Here $H^{(k-1)} = \nabla^2 g(x^{(k-1)})$ is the Hessian at $x^{(k-1)}$, and t_k is a step size. Equivalent formulation:

$$
\begin{aligned}
z^{(k)} = \operatorname*{argmin}_z \Big\{ & \nabla g(x^{(k-1)})^T (z - x^{(k-1)}) + \\
& \qquad \frac{1}{2}(z - x^{(k-1)})^T H^{(k-1)} (z - x^{(k-1)}) + h(z) \Big\}
\end{aligned}
$$

$$
x^{(k)} = x^{(k-1)} + t_k (z^{(k)} - x^{(k-1)})
$$

Scaled proximal map

Given $H \succ 0$, define

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

where $\|x\|_H^2 = x^T H x$. This is called a scaled proximal map

With $H = \frac{1}{t}I$, we get back usual (unscaled) definition. In general, the scaled prox shares retains many of the nice properties of usual prox (e.g., uniqueness, nonexpansiveness)

Now consider

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

Thus another equivalent form for proximal Newton update:

$$\begin{aligned} z^{(k)} &= \mathrm{prox}_{H^{(k-1)}}\big(x^{(k-1)} - (H^{(k-1)})^{-1}\nabla g(x^{(k-1)})\big) \\ x^{(k)} &= x^{(k-1)} + t_k(z^{(k)} - x^{(k-1)}) \end{aligned}$$

Notes:

- When $h(z) = 0$, we get back the usual Newton update
- If we replaced $H^{(k-1)}$ by $\frac{1}{r_k}I$, and set $t_k = 1$, we get proximal gradient update, with step size r_k
- Difficulty of prox depends strongly on h. However, now it also depends on the structure of the Hessian of g
- E.g., having a diagonal or banded Hessian generally makes a big difference compared to a dense Hessian

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 apply backtracking line search: fix $0 < \alpha \leq 1/2$, $0 < \beta < 1$, and let

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

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

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

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

When would we use proximal Newton?

High-level picture, for problem: $\min_x \; g(x) + h(x)$

Proximal gradient	Proximal Newton
• Iteratively minimize $\|b - x\|_2^2 + h(x)$	• Iteratively minimize $b^T x + x^T A x + h(x)$
• Often closed-form prox	• Almost never closed-form prox
• Iterations are cheap	• Iterations are very very expensive
• Convergence of gradient descent	• Convergence of Newton's method

So we use proximal Newton when we have an fast inner optimizer for scaled prox (quadratic plus h), expect few iterations

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 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 $k \geq k_0$, to get within $f(x^{(k)}) - f^\star \leq \epsilon$, we need $O(\log\log(1/\epsilon))$ iterations. Note: each iteration uses scaled prox evaluation!

Proof sketch

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

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

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

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

$$\|x^+ - x^\star\|_2 \underset{\substack{\uparrow\\ \text{lowest eigenvalue}\\ \text{bound}}}{\leq} \frac{1}{\sqrt{m}}\|x^+ - x^\star\|_H \underset{\substack{\uparrow\\ \text{nonexpansiveness,}\\ \text{Lipschitzness,}\\ \text{largest eigenvalue}}}{\leq} \frac{M}{2m}\|x - x^\star\|_2^2$$

Glmnet and QUIC

Two notable examples of proximal Newton methods:

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

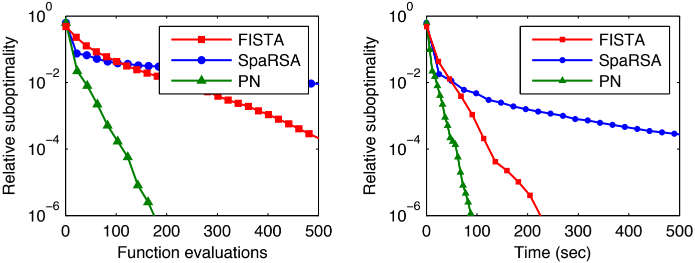
Both of these implementations are very widely used for their own purposes. At the proper scale, these are ≈ state-of-the-art

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

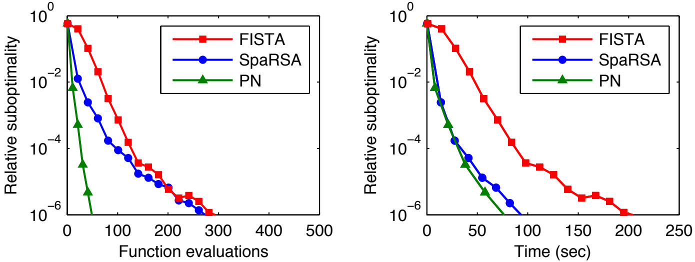
Example from Lee et al. (2012): ℓ_1 regularized logistic regression, FISTA (accelerated prox grad) versus spaRSA (spectral projected gradient method) versus PN (proximal Newton)

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



Here cost is dominated by expensive $g, \nabla g$ $(\exp, \log)$ evaluations

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

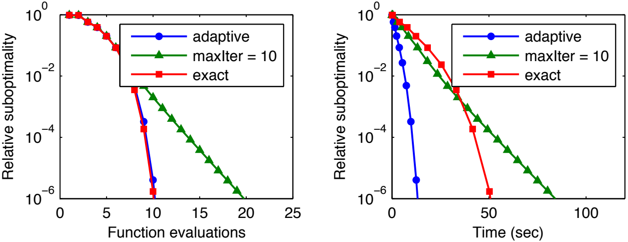


Now evaluating $g, \nabla g$ make up less of total cost, since X is sparse

Inexact prox evaluations

An important point: 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 $\tilde{g}_{k-1}$ quadratic approximation to g about $x^{(k-1)}$. Stop when

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

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

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

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

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

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 avoids forming $H^{(k-1)} = \nabla^2 g(x^{(k-1)})$ at each step

- Lee et al. (2012) propose BFGS-type updating rules. These work very well empirically, local superlinear convergence
- Tseng and Yun (2009) consider smooth plus block separable problems, propose approximating the Hessian in a blockwise fashion. Helpful because only small Hessians are ever needed. Their method has linear convergence

Quasi-Newton can be helpful not only when Hessian is burdensome computationally, but also when it is ill-conditioned: singular or near singular

What's wrong with projected Newton?

When $h = I_C(x)$, indicator function of convex set C, our problem:

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

Proximal gradient descent in this case reduces to projected gradient descent. What about proximal Newton? Updates are based on

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

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

Projected Newton for box constraints

Projected Newton method can be made to work for box constraints (Bertsekas, 1982; Kim et al., 2010; Schmidt et al., 2011). Given

$$\min_x \ g(x) \ \text{ subject to } \ l \le x \le u$$

the projected Newton method specifies an initial point $x^{(0)}$, small constant $\epsilon > 0$, and repeats the following steps for $k = 1, 2, 3, \ldots$

- Define the binding set

$$B_{k-1} = \{i : x_i^{(k-1)} \le l_i + \epsilon \text{ and } \nabla_i g(x^{(k-1)}) > 0\} \cup \\ \{i : x_i^{(k-1)} \ge u_i - \epsilon \text{ and } \nabla_i g(x^{(k-1)}) < 0\}$$

  These are the variables that are at (close to) boundary, and moving them inward would increase the criterion

- Define the free set $F_{k-1} = \{1, \dots n\} \setminus B_{k-1}$
- Define the inverse of the principal submatrix of the Hessian along the free variables

$$S^{(k-1)} = \left[\left(\nabla^2 g(x^{(k-1)}) \right)_{F_{k-1}} \right]^{-1}$$

- Take a Newton step along the free variables only, then project:

$$x^{(k)} = P_{[l,u]} \left(x^{(k-1)} - t_k \left[\begin{array}{cc} S^{(k-1)} & 0 \\ 0 & I \end{array} \right] \left[\begin{array}{c} \nabla_{F_{k-1}} g(x^{(k-1)}) \\ \nabla_{B_{k-1}} g(x^{(k-1)}) \end{array} \right] \right)$$

  where $P_{[l,u]}$ is the projection onto $[l, u] = [l_1, u_1] \times \dots [l_n, u_n]$
- Note that the update leaves binding set effectively untouched

Convergence properties

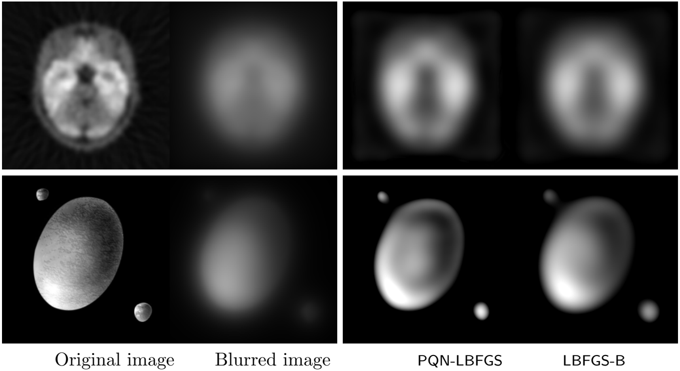
Convergence:

- Bertsekas (1982) shows that, under appropriate assumptions, projected Newton identifies the proper binding constraints in a finite number of iterations. Then it is just the usual Newton's method on the free variables
- Bertsekas (1982) also proves superlinear convergence
- Kim et al. (2010), Schmidt et al. (2011) describe a projected quasi-Newton method, using BFGS-style updates

What kinds of problems have box constraints? Lots, it turns out!

- Nonnegative least squares
- Support vector machine dual
- Graphical lasso dual
- Fused lasso (TV denoising) dual

Example from Kim et al. (2010): image deblurring performed with nonnegative KL divergence minimization



References

Proximal Newton method:

- 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 (2014), "Proximal Newton-type methods for minimizing composite functions"
- P. Tseng and S. Yun (2009), "A coordinate gradient descent method for nonsmooth separable minimization"

Projected Newton method:

- A. Barbero and S. Sra (2011), "Fast Newton-type methods for total variation regularization"
- D. Bertsekas (1982), "Projected Newton methods for optimization problems with simple constraints"
- D. Kim and S. Sra. and I. Dhillon (2010), "Tackling box-constrained optimization via a new projected quasi-Newton approach"
- M. Schmidt and D. Kim and S. Sra (2011), "Projected Newton-type methods in machine learning"