Proximal Newton Method

Lecturer: Ryan Tibshirani
Convex Optimization 10-725/36-725
Last time: quasi-Newton methods

Consider the problem

$$\min_{x} f(x)$$

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

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

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

- SR1: rank 1 update for Hessian, use SMW for inverse Hessian
- DFP: rank 2 update for inverse Hessian, use SMW 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 choose initial $x^{(0)}$ and repeat for $k = 1, 2, 3, \ldots$

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

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

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

- Difficulty of iterations is in applying prox, which depends only on $h$ (assuming that $\nabla g$ 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 $h$

$$x^+ = \arg\min_z \frac{1}{2t} \|x - t \nabla g(x) - z\|^2_2 + h(z)$$

$$= \arg\min_z \nabla g(x)^T(z - x) + \frac{1}{2t} \|z - x\|^2_2 + h(z)$$

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** 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 $\nabla^2 g(x)$?
This leads us to the **proximal Newton method**. Starting with $x^{(0)}$, we repeat for $k = 1, 2, 3, \ldots$

$$v^{(k)} = \arg\min_v \nabla g(x^{(k-1)})^T v + \frac{1}{2} v^T H^{(k-1)} v + h(x^{(k-1)} + v)$$

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

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:

$$z^{(k)} = \arg\min_z \nabla g(x^{(k-1)})^T (z - x^{(k-1)}) + \frac{1}{2} (z - x^{(k-1)})^T H^{(k-1)} (z - x^{(k-1)}) + h(z)$$

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

Given $H \succ 0$, let us define

$$\text{prox}_H(x) = \arg\min_z \frac{1}{2} \|x - z\|^2_H + h(z)$$

where $\|x\|^2_H = x^T H x$. 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

$$z^+ = \arg\min_z \nabla g(x)^T (z - x) + \frac{1}{2} (z - x)^T H (z - x) + h(z)$$

$$= \arg\min_z \frac{1}{2} \|x - H^{-1} \nabla g(x) - z\|^2_H + h(z)$$
Thus another equivalent form for the proximal Newton update is:

\[
\begin{align*}
    z^{(k)} &= \text{prox}_{H^{(k-1)}} \left( x^{(k-1)} - \left( H^{(k-1)} \right)^{-1} \nabla g(x^{(k-1)}) \right) \\
    x^{(k)} &= x^{(k-1)} + t_k \left( z^{(k)} - x^{(k-1)} \right)
\end{align*}
\]

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 need to apply, say, backtracking line search. Fix \( 0 < \alpha \leq 1/2, \ 0 < \beta < 1, \) and let

\[
v = \text{prox}_H(x - H^{-1}\nabla g(x)) - x
\]

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

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

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: \( \min_x g(x) + h(x) \)

<table>
<thead>
<tr>
<th>Proximal gradient</th>
<th>Proximal Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Iteratively minimize ( |b - x|^2_2 + h(x) )</td>
<td>• Iteratively minimize ( b^T x + x^T Ax + h(x) )</td>
</tr>
<tr>
<td>• Often closed-form prox</td>
<td>• Almost never closed-form prox</td>
</tr>
<tr>
<td>• Iterations are cheap</td>
<td>• Iterations are very very expensive</td>
</tr>
<tr>
<td>• Convergence of gradient descent</td>
<td>• Convergence of Newton’s method</td>
</tr>
</tbody>
</table>

So we use proximal Newton when we have a 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 $M$
- $\text{prox}_H(\cdot)$ is exactly evaluable

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

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

Recall that this is called **local quadratic convergence**. After $k \geq k_0$, to get within $f(x^{(k)}) - f^* \leq \epsilon$, 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, they show that for large enough \( k \), the pure step \( t = 1 \) eventually satisfies backtracking exit condition. Therefore

\[ \| x^+ - x^* \|_2 \leq \frac{1}{\sqrt{m}} \| x^+ - x^* \|_H \leq \frac{M}{2m} \| x - x^* \|_2^2 \]

\[ \text{\textup{\uparrow \ updownarrow \ lowest eigenvalue \ bound}} \]  

\[ \text{\textup{\updownarrow \ updownarrow \ nonexpansiveness, }} \]

\[ \text{\textup{\ updownarrow \ Lipschitzness, largest eigenvalue}} \]
Glmnet and QUIC

Two notable examples of proximal Newton methods:

- **glmnet** (Friedman et al., 2009): prox Newton for $\ell_1$ 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 $\approx$ 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): $\ell_1$ 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 \( X \)

![Graph](image)

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 $\eta_k$, $k = 1, 2, 3, \ldots$.

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.
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, and have 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

$$z^+ = \arg\min_{z \in C} \frac{1}{2} \|x - H^{-1}\nabla g(x) - z\|_H^2$$

$$= \arg\min_{z \in C} \nabla g(x)^T(z - x) + \frac{1}{2}(z - x)^TH(z - x)$$

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

In special cases, projected Newton can be made to work, e.g., for box constraints (Bertsekas, 1982; Kim et al., 2010; Schmidt et al., 2011). Given a problem

\[
\min_x g(x) \quad \text{subject to} \quad l \leq x \leq u
\]

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

\[
B_{k-1} = \{i : x_i^{(k-1)} \leq l_i + \epsilon \text{ and } \nabla_i g(x^{(k-1)}) > 0\} \cup \{i : x_i^{(k-1)} \geq u_i - \epsilon \text{ and } \nabla_i g(x^{(k-1)}) < 0\}
\]

These are the variables that are at (close to) boundary, and moving them any further would decrease the criterion.
• Define the free set $F_{k-1} = \{1, \ldots n\} \setminus B_{k-1}$

• Define the inverse of the principal submatrix of the Hessian along the free variables

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

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

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

where $P_{[l,u]}$ is the projection onto $[l, u] = [l_1, u_1] \times \ldots [l_n, u_n]$

• Note that the update leaves binding set effectively untouched
Convergence properties:

• 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 (total variation denoising) dual
Example from Kim et al. (2010): image deblurring performed with nonnegative KL divergence minimization
References

Proximal Newton method:

Projected Newton method: