10-725/36-725: Convex Optimization **Fall 2016**

Lecture 19: November 7

Lecturer: Ryan Tibshirani *Scribes: Otilia Stretcu, Akash Umakantha*

19.1 Review of Quasi-Newton Methods

Consider the following problem:

$$\min_x f(x)$$

with f convex, twice differentiable, and dom(f)=$\mathbb{R}^n$.

The quasi-Newton method in generic form is the following:

1. Start with $x^{(0)} \in \mathbb{R}^n$
2. Repeat for $k = 1, 2, 3,...$

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

   where $M^{(k-1)} \approx (\nabla^2 f(x^{(k-1)}))^{-1}$ is an approximation to the inverse Hessian at $x^{(k-1)}$.

The step sizes t_k are chosen by backtracking.

The key idea of quasi-Newton methods is that $M^{(0)}$ should be easily computed, and $M^{(k)}$ should be easily updated from $M^{(k-1)}$ for $k \geq 1$. The most popular methods for computing M are the following:

- SR1: computes a rank 1 update for the Hessian, and uses the Sherman-Morrision-Woodbury (SMW) formula to compute the inverse Hessian.
- DFP: computes a rank 2 update for the inverse Hessian, and uses SMW for the Hessian.
- BFGS: reverses the role of the Hessian and the inverse Hessian in DFP.
- LBFGS: a very popular version of BFGS for limited memory.

19.2 Review of Proximal Gradient Descent

Proximal gradient descent operates on problems of the form:

$$\min_x g(x) + f(x)$$

where g is convex and smooth, and h is convex and "simple".

The main steps of the proximal gradient descent algorithm are the following:

1. Choose an initial $x^{(0)}$.

2. Repeat for k = 1, 2, 3, ...
$$x^{(k)} = \text{prox}_{t_k}(x^{(k-1)} - t_k \nabla g(x^{(k-1)})$$
where $\text{prox}_t(.)$ is the proximal operator associated with h:
$$\text{prox}_t(x) = \arg\min_x \frac{1}{2t}||x - z||_2^2 + h(z)$$

If we can compute ∇g, then the difficulty of the iterations depends only on the form of h.

Proximal gradient descent has the same convergence rate as its fully smooth version. However, since each iteration applies the prox operator, proximal gradient descent is useful only when the prox computation is efficient.

The motivation for proximal gradient descent is to deal with the non-smooth term h by iteratively minimizing a quadratic expansion in g, plus the original h. We can rearrange the update step for x as:

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

The update is equivalent to applying the prox operator to a local quadratic approximation, where the curvature of the quadratic is given by $\frac{1}{t}I$ (spherical curvature). Similar to this method, Newton's method also iteratively minimizes quadratic approximations, but here the curvature of the quadratic is given by the local Hessian of the function in question, $\nabla^2 f(x)$. Combining these ideas, we can replace the curvature $\frac{1}{t}I$ in the prox method, with the Hessian $\nabla^2 g(x)$. This leads us to the proximal Newton method, described in the next section.

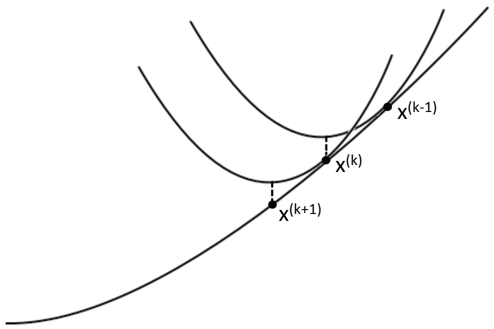

Figure 19.1: Proximal gradient descent updates.

19.3 Proximal Newton Method

19.3.1 Formulation

The proximal Newton method can be formulated as follows:

1. Choose a starting point $x^{(0)}$.
2. Find an update direction:
$$v^{(k)} = \arg\min_v \nabla g(x^{(k-1)})^T v + \frac{1}{2} v^T H^{(k-1)} v + h(x^{(k-1)} + v)$$
where $H^{(k-1)} = \nabla^2 g(x^{(k-1)})$ is the Hessian at $x^{(k-1)}$.
3. Make a step in the direction of $v^{(k)}$:
$$x^{(k)} = x^{(k-1)} + t_k v^{(k)}$$
where t_k is a step size.

Note that in step 2, we find a direction $v^{(k)}$ that minimizes the sum of a quadratic approximation of g (using the Hessian) and h (evaluated at the new point in the direction v). For the step size t_k, in Newton's method we would pick $t_k = 1$, but here we can also apply step size optimization methods.

Another equivalent formulation of proximal Newton is the following:

$$\begin{aligned} z^{(k)} &= \arg\min_x \nabla g(x^{(k-1)})^T (z - x^{(k-1)}) + \frac{1}{2}(z - x^{(k-1)})^T H^{(k-1)} (z - x^{(k-1)}) + h(z) \\ x^{(k)} &= x^{(k-1)} + t_k (z^{(k)} - x^{(k-1)}) \end{aligned}$$

Intuitively, in the first step, we find a surrogate point z that minimizes a quadratic approximation to g, plus h, and in the second step we move from $x^{(k-1)}$ towards z, but not all the way.

19.3.2 Scaled proximal map

We can rewrite the proximal Newton steps in a form that resembles the proximal gradient descent update. To this end, we introduce the following notation. Given $H \succ 0$, let us define a **scaled proximal map** as

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

where $||x||_H^2 = x^T H x$ is the H-norm.

Note that for $H = \frac{1}{t} I$, we get back the usual (unscaled) definition.

In general, the scaled prox retains many of the properties of the usual prox, such as:

- uniqueness: when $H \succ 0$, the prox map is well defined, because we have a strictly convex optimization problem and the minimizer is thus unique.
- non-expansiveness: the prox map generalizes a projection operator, which is non-expansive. That is, for two points x and y, and a projection operator to a convex set C, P_c, the non-expansiveness of P_C means $||P_c(x) - P_c(y)||_2 \le ||x - y||_2$. In other words, P_c is 1-Lipschitz. This property holds only when C is convex.

Now consider:

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

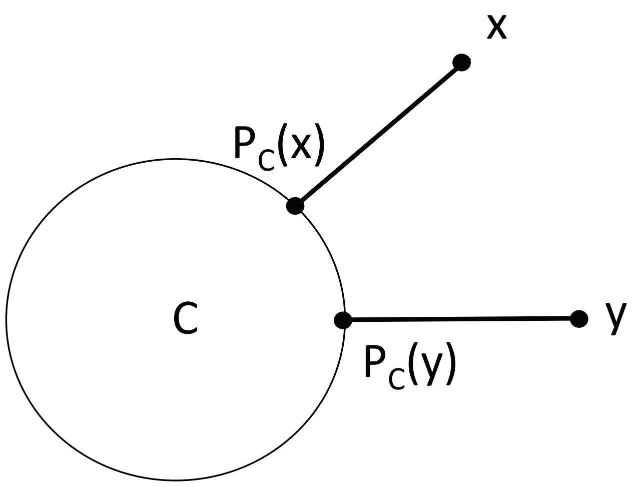

Figure 19.2: Projection onto a convex set C.

Thus another equivalent form for the proximal Newton update is:

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

Intuitively, this means we take a Newton step with respect to g, we apply the scaled prox operator with respect to $H^{(k-1)}$, and then we move in that direction.

Some useful remarks are the following:

- When $h(z) = 0$, the prox is then identity, and we get back the usual Newton update.
- If we replaced $H^{(k-1)}$ by $\frac{1}{r_k}I$, and set $t_k = 1$ (as in pure Newton's method), we get the proximal gradient update with step size r_k. Thus we can see proximal Newton generalizes both proximal gradient descent and Newton's method.
- The difficulty of the prox depends both on h, and on the structure of the Hessian of g. For example, if H is diagonal or banded, the problem is much easier than compared to a dense H.

19.3.3 Backtracking line search for Proximal Newton Method

We would like to select a step size t_k such that we ensure convergence, and we evaluate the prox operator as few times as possible. One possible method for setting t_k is the following backtracking line search approach:

- Fix $0 \le \alpha \le \frac{1}{2}$, and $0 < \beta < 1$.
- Compute $v = \text{prox}_H(x - H^{-1}\nabla g(x)) - x$, the proximal Newton direction at a given iteration.
- Set $t = 1$.
- While
$$f(x + tv) > f(x) + \alpha t \nabla g(x)^T v + \alpha(h(x + tv) - h(x))$$
shrink $t = \beta t$.

Here, $f = g + h$. Intuitively, we seek a step t to move along the direction v, such that we are in a point within a factor α of a linear approximation around point x of our function. However, since the h part of f is not smooth, we use its discrete derivative, given by $h(x + tv) - h(x)$.

Note that there are many ways to perform backtracking line search. In the one presented here, however, we only need to evaluate the prox operator once, when calculating v.

19.3.4 When is proximal Newton useful?

In order to understand when it is useful to use Newton's method, let us compare the properties of proximal Newton versus proximal gradient descent, for the problem $\min_x g(x) + h(x)$ introduced earlier.

Proximal gradient	Proximal Newton
Iteratively minimize $\|\|b - x\|\|_2^2 + h(x)$.	Iteratively minimize $b^T x + \frac{1}{2}x^T Ax + h(x)$.
The prox operator is often known in closed form.	The prox operator is almost never known in closed form, and the problem is treated as an inner optimization problem.
Iterations are cheap.	Iterations are very expensive (more expensive than Newton's method iterations).
Has the same convergence as gradient descent in terms of progress per iteration.	Has the same convergence as Newton's method in terms of progress per iteration.

Therefore, proximal Newton is useful when he have a **fast inner optimizer for the scaled prox map** (for the quadratic plus h), and we expect only few iterations. A commonly used inner optimizer is coordinate descent, when h is a separable function.

19.3.5 Convergence Analysis

Here we will follow the proof in Lee (2014) [1]. Assume:

- $f = g + h$ where g and h convex and g is twice differentiable (smooth)
- $mI \preceq \nabla^2 g(x) \preceq LI$
- $\nabla^2 g(x)$ Lipshitz with constant M
- $prox_H(.)$ can be evaluated exactly

Theorem 19.1 *Proximal Newton with backtracking line search converges globally. Additionally, $\forall k \geq k_0$, proximal Newton has a* ***local quadratic convergence*** *rate:*

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

Local quadratic convergence means that after $k \geq k_0$, we need $O(loglog(1/\epsilon)$ iterations in order to achieve $f(x^{(k)} - f^* \leq \epsilon$. In other words, when we are close enough to the solution we get very fast convergence.

Proof:

1. It is possible to show that the backtracking exit condition satisfies (for step size t):

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

2. This can be used to show that the update direction converges to 0, which can only happen at the global minimum.
3. Now for local quadratic converge, it is possible to show that after some number of iteration k_0, the pure Newton step $t = 1$ satisfies backtracking exit conditions.
4. This implies

$$\begin{aligned}\|x^+ - x^*\|_2 \leq & \frac{1}{\sqrt{m}}\|x^+ - x^*\|_H \\ = & \frac{1}{\sqrt{m}}\|prox_H(x - H^{-1}\nabla g(x)) - prox_H(x^* - H^{-1}\nabla g(x^*))\|_H \\ \leq & \frac{M}{2m}\|x^{(k-1)} - x^*\|_2^2\end{aligned}$$

   The first inequality holds by the lowest eigenvalue bound, the equality holds by the definition of x^+ and the fact that $prox_H(.)$ is identity at the global minimum x^*, and the final inequality holds by nonexpasiveness of the proximal operator, the Lipshitz assumption, and the largest eigenvalue bound.

■

19.3.6 Notable examples

Some very popular packages that use proximal Newton are:

- glmnet (Friedman et al., 2009) for l_1 penalized linear models
- QUIC (Hsieh et al., 2011) for graphical lasso

Both software packages use proximal Newton in their outer loops, and coordinate descent in their inner loops. Both are state of the art for problems of "not-too-large" scale.

In general, proximal Newton evaluates the gradient $\nabla g(x)$ far fewer times than proximal gradient. So when $\nabla g(x)$ is very expensive, proximal Newton may be a better choice. This is the case for lasso logistic regression, where proximal Newton does better than other methods like FISTA (prox gradient) or spaRSA (spectral gradient).

19.3.7 Inexact proximal Newton

In proximal Newton, evaluating the proximal operation is almost always done inexactly, but needs to be done to a high degree of accuracy to have nice properties. Lee (2014) [1] propose a stopping rule for this inner problem that maintains global convergence and local superlinear convergence.

In regular Newton method, the inner problem minimizes $\tilde{g}$, the quadratic approximation to g about $x^{(k-1)}$ and stops when

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

for some forcing sequence η_k.

For inexact proximal Newton, Lee (2014) propose using generalized gradients,

$$\|\nabla \tilde{G}_{\tilde{f}_{k-2}/M}(x^{(k)})\|_2 \leq \eta_k \|\nabla G_{f/M}(x^{(k-1)})\|_2$$

where $\tilde{f}_{k-1} = \tilde{g}_{k-1} + h$ and $mI \preceq \nabla^2 g(x) \preceq MI$. To get superlinear convergence, the forcing sequence is defined as

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

19.3.8 Proximal quasi-Newton

When the Hessian is computationally expensive or when it is ill-conditioned, proximal quasi-Newton methods may be useful. These methods avoid forming the hessian $H^{(k-1)} = \nabla g(x^{(k-1)})$ at each step.

- Lee (2014) [1] propose using BFGS-style updates for the Hessian. This method tends to work well and has local superlinear convergence.
- Tseng and Yun (2009) [5] propose approximating the Hessian blockwise. This only works for problems where h (from $f = g + h$) can be separated into parts that rely only on some of the optimization variables. Evaluating the Hessian blockwise makes computation much faster. This method has linear convergence.

19.4 Projected Newton Method

19.4.1 Relationships between projected and proximal methods

Proximal gradient descent reduces to projected gradient descent when $h(x) = I_C(x)$. In other words, projected gradient descent is a special case of proximal gradient descent.

However, with proximal Newton method, we have

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

which is not a projection in general because the norm is with respect to H (the scaled proximal map). If instead, the norm were the 2 norm (i.e., $H = I$), we would have a projection. Thus, projected Newton method is not a special case of proximal Newton method.

19.4.2 Projected Newton for box constraints

Projected Newton works well in some cases, for example when there are box constraints, $l \leq x \leq u$. Examples of problems that have box constraints include:

- SVM dual
- fused lasso dual
- graphical lasso dual
- non-negative least squares

In these problems, we can use projected Newton method. For some $\epsilon > 0$:

1. Define binding set ($B^{(k-1)}$): $x_i^{(k-1)}$ within ϵ of l_i with $\nabla_i g(x^{(k-1)}) > 0$ or $x_i^{(k-1)}$ within ϵ of u_i with $\nabla_i g(x^{(k-1)}) < 0$. In words, taking optimization steps in these variables would push further towards the box constraint boundaries.
$$B^{(k-1)} = \{i : x_i^{(k-1)} < l_i + \epsilon \text{ and } \nabla_i g(x^{(k-1)}) > 0\} \cup \{i : x_i^{(k-1)} > u_i - \epsilon \text{ and } \nabla_i g(x^{(k-1)}) < 0\}$$
2. Define free set ($F^{(k-1)}$): complement of the binding set $\{1, ..., n\} \backslash B^{(k-1)}$.
3. Compute inverse of the hessian among the variables in the free set: $S^{(k-1)} = \left[\nabla^2 g(x^{(k-1)})_{F^{(k-1)}}\right]^{-1}$.
4. Take a Newton step along the direction of the free variables and a gradient step along the direction of the binding variables. Then project onto the box constraint set $[l, u] = [l_1, u_1] \times ... \times [l_n, u_n]$.
$$x^{(k)} = P_{[l,u]}\left(x^{(k-1)} - t_k \begin{bmatrix} S^{(k-1)} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \nabla_{F^{(k-1)}} g(x^{(k-1)}) \\ \nabla_{B^{(k-1)}} g(x^{(k-1)}) \end{bmatrix}\right)$$
Projection onto the box constraint set is simple. Check if $l_i \leq x_i \leq u_i$, and if not, set x_i to the appropriate lower bound l_i or upper bound u_i.

19.4.3 Convergence of projected Newton for box constraints

Bertsekas (1982) [2] discusses the convergence properties of projected gradient method. Bersekas shows:

1. The binding constraints can be identified in a finite number of iterations.
2. Projected Newton has a superlinear convergence rate.

More recent papers [3,4] use BFGS-like updates to develop projected quasi-Newton methods.

19.5 glmnet details

$$\arg\min_\beta g(\beta) + \lambda\|\beta\|_1 = \arg\min_\beta L(X\beta) + \lambda\|\beta\|_1$$

glmnet solves this problem via pathwise optimization over $\lambda_1 > ... > \lambda_m \geq 0$. Pathwise optimization is often faster even if you simply want the solution at a single λ_i. The coordinate-wise KKT conditions (for loss L and predictive function f) are:
$$x_i^T(y - f_i(\beta)) = \lambda s_i$$

Then the active set is $A = \{j : |x_j^T(y - f_j(\beta^{(0)}))| \geq \lambda\}$. If this is not the case, then the corresponding dual variable s_j must be smaller than 1, and thus $\beta_j = 0$. Now do proximal Newton on only the variables in set A. After convergence, check the KKT conditions, and if they are not met for some coordinate i, then simply add i to the active set and repeat the above procedure.

Once the above is complete for a particular λ_i, then we move to the next $\lambda_j < \lambda_i$ for $j = i + 1$ and repeat. Now, we already have a good approximation for the active set A, and the problem is quite fast and easy to solve.

References

[1] J. Lee and Y. Sun and M. Saunders (2014), Proximal Newton-type methods for minimizing composite functions

[2] D. Bertsekas (1982), Projected Newton methods for optimization problems with simple constraints.

[3] D. Kim and S. Sra. and I. Dhillon (2010), Tackling box-constrained optimization via a new projected quasi-Newton approach

[4] M. Schmidt and D. Kim and S. Sra (2011), Projected Newton-type methods in machine learning

[5] P. Tseng and S. Yun (2009), A coordinate gradient descent method for nonsmooth separable minimization