Proximal Newton-type methods for minimizing composite functions

Jason D. Lee
Joint work with Yuekai Sun, Michael A. Saunders

Institute for Computational and Mathematical Engineering, Stanford University

June 12, 2014

Minimizing composite functions

Proximal Newton-type methods

Inexact search directions

Computational experiments

Minimizing composite functions

$$\underset{x}{\text{minimize}}\ f(x) := g(x) + h(x)$$

- g and h are convex functions
- g is continuously differentiable, and its gradient ∇g is Lipschitz continuous
- h is not necessarily everywhere differentiable, but its *proximal mapping* can be evaluated efficiently

Minimizing composite functions: Examples

ℓ_1-regularized logistic regression:

$$\min_{w \in \mathbf{R}^p} \; \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) + \lambda \|w\|_1.$$

Sparse inverse covariance:

$$\min_{\Theta} \; -\text{logdet}(\Theta) + \mathbf{tr}(S\Theta) + \lambda \|\Theta\|_1$$

Minimizing composite functions: Examples

Graphical Model Structure Learning

$$\min_{\theta} \; - \sum_{(r,j)\in E} \theta_{rj}(x_r, x_j) + \log Z(\theta) + \lambda \sum_{(r,j)\in E} \|\theta_{rj}\|_F.$$

Multiclass Classification:

$$\min_{W} \sum_{i=1}^{n} - \log \left(\frac{e^{w_{y_i}^T x_i}}{\sum_k e^{w_k^T x_i}} \right) + \|W\|_*$$

Minimizing composite functions: Examples

Arbitrary convex program

$$\min_x \ g(x) + \mathbf{1}_C(x)$$

Equivalent to solving

$$\min_{x \in C} \ g(x)$$

The proximal mapping

The proximal mapping of a convex function h is

$$\operatorname{prox}_h(x) = \arg\min_y \; h(y) + \frac{1}{2}\|y - x\|_2^2 .$$

- $\operatorname{prox}_h(x)$ exists and is unique for all $x \in \operatorname{dom} h$
- proximal mappings generalize projections onto convex sets

Example: soft-thresholding: Let $h(x) = \|x\|_1$. Then

$$\operatorname{prox}_{t\|\cdot\|_1}(x) = \operatorname{sign}(x) \cdot \max\{|x| - t, 0\}.$$

The proximal gradient step

$$
\begin{aligned}
x_{k+1} &= \text{prox}_{t_k h}\left(x_k - t_k \nabla g(x_k)\right) \\
&= \arg\min_y h(y) + \frac{1}{2t_k}\left\|y - (x_k - t_k \nabla g(x_k))\right\|^2 \\
&= x_k - t_k G_{t_k f}(x_k)
\end{aligned}
$$

- $G_{t_k f}(x_k)$ minimizes a simple quadratic model of f:

$$
-t_k G_{t_k f}(x_k) = \arg\min_d \; \nabla g(x_k)^T d + \underbrace{\frac{1}{2t_k}\|d\|_2^2}_{\text{simple quadratic}} + h(x_k + d).
$$

- $G_f(x)$ can be thought of as a generalized gradient of $f(x)$. Simplifies to the gradient descent on $g(x)$ when $h = 0$.

The proximal gradient method

Algorithm 1 The proximal gradient method

Require: starting point $x_0 \in \operatorname{dom} f$

1: **repeat**

2: Compute a *proximal gradient step*:
$$G_{t_k f}(x_k) = \frac{1}{t_k}\left(x_k - \operatorname{prox}_{t_k h}(x_k - t_k \nabla g(x_k))\right).$$

3: Update: $x_{k+1} \leftarrow x_k - t_k G_{t_k f}(x_k)$.

4: **until** stopping conditions are satisfied.

Proximal Newton-type methods

Main idea: use a local quadratic model (in lieu of a simple quadratic model) to account for the curvature of g:

$$\Delta x_k := \arg\min_d \nabla g(x_k)^T d + \underbrace{\frac{1}{2} d^T H_k d}_{\text{local quadratic}} + h(x_k + d).$$

Solve the above subproblem and update

$$x_{k+1} = x_k + t_k \Delta x_k.$$

A generic proximal Newton-type method

Algorithm 2 A generic proximal Newton-type method

Require: starting point $x_0 \in \operatorname{dom} f$

repeat

    Choose an approximation to the Hessian H_k.

    Solve the subproblem for a search direction:

    $$\Delta x_k \leftarrow \arg\min_d \nabla g(x_k)^T d + \frac{1}{2} d^T H_k d + h(x_k + d).$$

    Select t_k with a backtracking line search.

    Update: $x_{k+1} \leftarrow x_k + t_k \Delta x_k$.

until stopping conditions are satisfied.

Why are these proximal?

Definition (Scaled proximal mappings)

Let h be a convex function and H, a positive definite matrix. Then the scaled proximal mapping of h at x is defined to be

$$\text{prox}_h^H(x) = \arg\min_y \; h(y) + \frac{1}{2}\|y - x\|_H^2.$$

The proximal Newton update is

$$x_{k+1} = \text{prox}_h^{H_k}\left(x_k - H_k^{-1}\nabla g(x_k)\right)$$

and analogous to the proximal gradient update

$$x_{k+1} = \text{prox}_{h/L}\left(x_k - \frac{1}{L}\nabla g(x_k)\right)$$

$\Delta x = 0$ if and only if x minimizes $f = g + h$.

A classical idea

Traces back to:

- Projected Newton-type methods
- Generalized proximal point methods

Popular methods tailored to specific problems:

- `glmnet`: lasso and elastic-net regularized generalized linear models
- LIBLINEAR: ℓ_1-regularized logistic regression
- QUIC: sparse inverse covariance estimation

Choosing an approximation to the Hessian

1. **Proximal Newton method:** use Hessian $\nabla^2 g(x_k)$
2. **Proximal quasi-Newton methods:** build an approximation to $\nabla^2 g(x_k)$ using changes in ∇g:

$$H_{k+1}(x_{k+1} - x_k) = \nabla g(x_k) - \nabla g(x_{k+1})$$

3. If problem is large, use limited memory versions of quasi-Newton updates (e.g. L-BFGS)
4. Diagonal+rank 1 approximation to the Hessian.

Bottom line: Most strategies for choosing Hessian approximations Newton-type methods also work for proximal Newton-type methods

Theoretical results

Take home message:

The convergence of proximal Newton methods parallel those of the regular Newton Method.

Global convergence:

- smallest eigenvalue of H_k's bounded away from zero

Quadratic convergence (prox-Newton method):

- Quadratic convergence: $\|x_k - x^\star\|^2 \le c^{2^k}$ or $\log\log\frac{1}{\epsilon}$ iterations to achieve ϵ accuracy.
- Assumptions: g is strongly convex, and $\nabla^2 g$ is Lipschitz continuous

Superlinear convergence (prox-quasi-Newton methods):

- BFGS, SR1, and many other hessian approximations. Dennis-More condition $\frac{\left\|\left(H_k - \nabla^2 g(x^\star)\right)(x_{k+1} - x_k)\right\|_2}{\|x_{k+1} - x_k\|_2} \to 0.$
- Superlinear convergence means it is faster than any linear rate. E.g. c^{k^2} converges superlinearly to 0.

Questions so far?

Any Questions?

Minimizing composite functions

Proximal Newton-type methods

Inexact search directions

Computational experiments

Solving the subproblem

$$\Delta x_k = \arg\min_d \; \nabla g(x_k)^T d + \frac{1}{2} d^T H_k d + h(x_k + d)$$
$$= \arg\min_d \; \hat{g}_k(x_k + d) + h(x_k + d)$$

Usually, we must use an iterative method to solve this subproblem.

- Use proximal gradient or coordinate descent on the subproblem.
- A gradient/coordinate descent iteration on the subproblem is much cheaper than a gradient iteration on the original function f, since it does not require a pass over the data. By solving the subproblem, we are more efficiently using a gradient evaluation than gradient descent.
- H_k is commonly a L-BFGS approximation, so computing a gradient takes $O(Lp)$. A gradient of the original function takes $O(np)$. The subproblem is independent of n.

Inexact Newton-type methods

Main idea: no need to solve the subproblem exactly only need a good enough search direction.

- We solve the subproblem approximately with an iterative method, terminating (sometimes very) early
- number of iterations may increase, but computational expense per iteration is smaller
- many practical implementations use inexact search directions

What makes a stopping condition good?

We should solve the subproblem more precisely when:

1. x_k is close to $x^\star$, since Newton's method converges quadratically in this regime.
2. $\hat{g}_k + h$ is a good approximation to f in the vicinity of x_k (meaning H_k has captured the curvature in g), since minimizing the subproblem also minimizes f.

Early stopping conditions

For regular Newton's method the most common stopping condition is

$$\|\nabla \hat{g}_k(x_k + \Delta x_k)\| \leq \eta_k \|\nabla g(x_k)\| .$$

Analogously,

$$\underbrace{\left\|G_{(\hat{g}_k+h)/M}(x_k + \Delta x_k)\right\|}_{\text{optimality of subproblem solution}} \leq \eta_k \underbrace{\left\|G_{f/M}(x_k)\right\|}_{\text{optimality of } x_k}$$

Choose η_k based on how well $G_{\hat{g}_k+h}$ approximates G_f:

$$\eta_k \sim \frac{\left\|G_{(\hat{g}_{k-1}+h)/M}(x_k) - G_{f/M}(x_k)\right\|}{\left\|G_{f/M}(x_{k-1})\right\|}$$

Reflects the Intuition: solve the subproblem more precisely when

- $G_{f/M}$ is small, so x_k is close to optimum.
- $G_{\hat{g}+h} - G_f \approx 0$, means that H_k is accurately capturing the curvature of g.

Convergence of the inexact prox-Newton method

- Inexact proximal Newton method converges superlinearly for the previous choice of stopping criterion and η_k.
- In practice, the stopping criterion works extremely well. It uses approximately the same number of iterations as solving the subproblem exactly, but spends much less time on each subproblem.

Minimizing composite functions

Proximal Newton-type methods

Inexact search directions

Computational experiments

Sparse inverse covariance (Graphical Lasso)

Sparse inverse covariance:

$$\min_{\Theta} -\text{logdet}(\Theta) + \mathbf{tr}(S\Theta) + \lambda\|\Theta\|_1$$

- S is a sample covariance, and estimates Σ the population covariance.

$$S = \sum_{i=1}^{p}(x_i - \mu)(x_i - \mu)^T$$

- S is not of full rank since $n < p$, so S^{-1} doesn't exist.
- Graphical lasso is a good estimator of Σ^{-1}

Sparse inverse covariance estimation

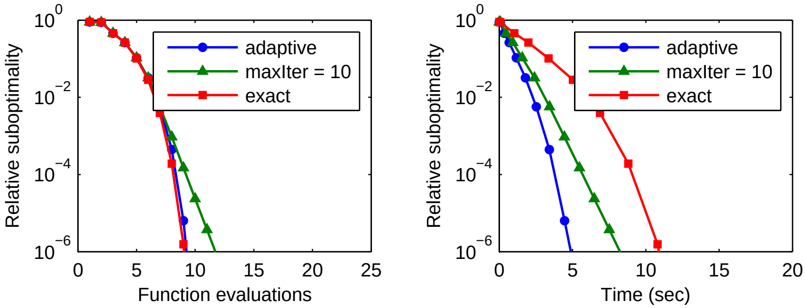
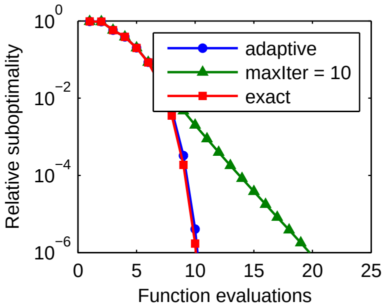


Figure: Proximal BFGS method with three subproblem stopping conditions (Estrogen dataset $p = 682$)

Sparse inverse covariance estimation



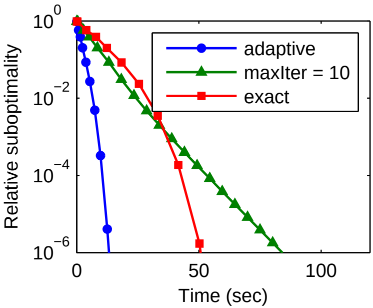


Figure: Leukemia dataset $p = 1255$

Another example

Sparse logistic regression

- training data: $x^{(1)}, \ldots, x^{(n)}$ with labels $y^{(1)}, \ldots, y^{(n)} \in \{0, 1\}$
- We fit a sparse logistic model to this data:

$$\underset{w}{\text{minimize}} \ \frac{1}{n} \sum_{i=1}^{n} -\log(1 + \exp(-y_i w^T x_i)) + \lambda \|w\|_1$$

Sparse logistic regression

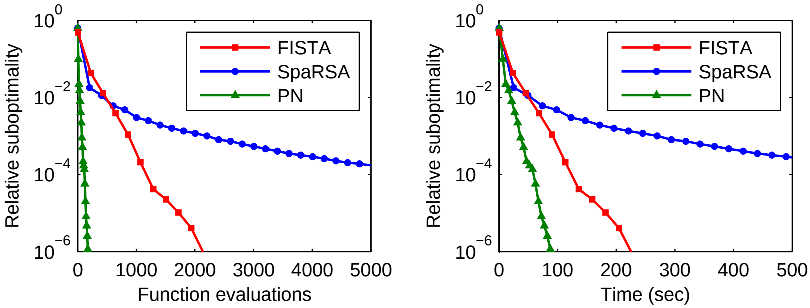
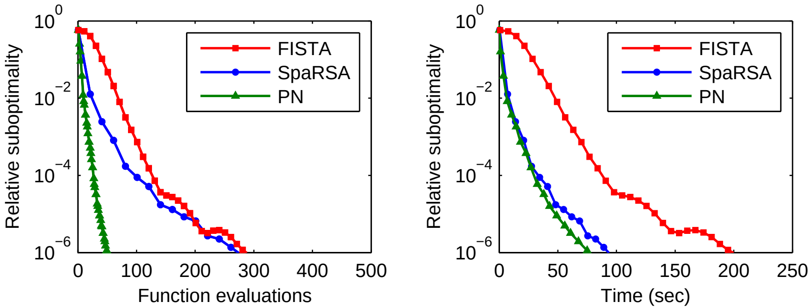


Figure: Proximal L-BFGS method vs. FISTA and SpaRSA (`gisette` dataset, $n = 5000$, $p = 6000$ and dense)

Sparse logistic regression

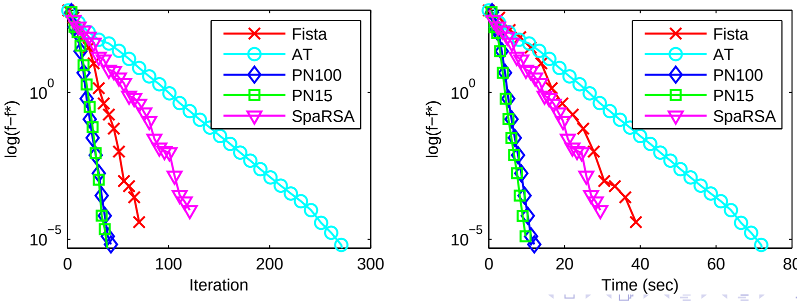
Figure: `rcv1` dataset, $n = 47,000$, $p = 542,000$ and 40 million nonzeros



Markov random field structure learning

$$\underset{\theta}{\text{minimize}} \; -\sum_{(r,j)\in E} \theta_{rj}(x_r, x_j) + \log Z(\theta) + \sum_{(r,j)\in E} \left(\lambda_1 \|\theta_{rj}\|_2 + \lambda_F \|\theta_{rj}\|_F^2\right).$$

Figure: Markov random field structure learning



Summary

Proximal Newton-type methods

- converge rapidly near the optimal solution, and can produce a solution of high accuracy
- are insensitive to the choice of coordinate system and to the condition number of the level sets of the objective
- are suited to problems where g, ∇g is expensive to evaluate compared to h, prox_h. This is the case when $g(x)$ is a loss function and computing the gradient requires a pass over the data.
- "more efficiently uses" a gradient evaluation of $g(x)$.

Thank you for your attention. Any questions?