10-725/36-725: Convex Optimization **Fall 2018**

Lecture 17: October 29

Lecturer: Lecturer: Ryan Tibshirani *Scribes: Matthew Levine, Junxian He*

17.1 Quasi-Newton Methods

Quasi-Newton methods are motivated to mitigate the expensive computation in vanilla Newton method (regarding Hessian matrix), and instead follow the procedure as:

- Solve $B^{(k-1)}\Delta x^{(k-1)} = -\nabla f(x^{(k-1)})$
- Update $x^{(k)} = x^{(k-1)} + t_k \Delta x^{(k-1)}$
- Compute $B^{(k)}$ from $B^{(k-1)}$

Since $B^{(k-1)}$ already contains info about the Hessian, usually we use suitable matrix update to form $B^{(k)} = g(B^{(k-1)})$, different methods will implement this g function differently, and commonly we can also compute $(B^{(k)})^{-1}$ from $(B^{(k-1)})^{-1}$. Therefore, we can maintain $B^{(k)}$ (or $(B^{(k)})^{-1}$) in memory during training, and obtain new $B^{(k)}$ (or $(B^{(k)})^{-1}$) from the last step, which results in a much cheaper procedure than original Newton methods.

17.1.1 Requirements of B^+

- $B^+ s = y$, where $y = \nabla f(x^+) - \nabla f(x)$, $s = x^+ - x$ (**secant equation**)
- B^+ is symmetric
- B^+ is "close" to B
- $B \succeq 0 \Rightarrow B^+ \succeq 0$

Next we will introduce several different Quasi-Newton methods.

17.1.2 Symmetric Rank-One (SR1) Update

- Rank-one update form: $B^+ = B + auu^T$
- Secant equation $B^+ s = y \Rightarrow (au^T s)u = y - Bs$

- Solve the above we obtain $B^+ = B + \frac{(y-Bs)(y-Bs)^T}{(y-Bs)^T s}$
- Given Sherman-Morrison formula:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u},$$

  we can derive the update form for inverse of B which we define as $C = B^{-1}$:

$$C^+ = C + \frac{(s - Cy)(s - Cy)^T}{(s - Cy)^T y}$$

- SR1 is simple and cheap, but does **NOT** preserve positive definiteness

17.1.3 Broyden-Fletcher-Goldfarb-Shanno (BFGS) Update

- Rank-two update form: $B^+ = B + auu^T + bvv^T$
- Secant equation $B^+ s = y \Rightarrow y - Bs = (au^T s)u + (bv^T s)v$
- Set $u = y, v = Bs$ and solve above to obtain $B^+ = B - \frac{Bss^T B}{s^T Bs} + \frac{yy^T}{y^T s}$
- Given Woodbury formula:

$$(A + UDV)^{-1} = A^{-1} - A^{-1}U(D^{-1} + VA^{-1}U)^{-1}VA^{-1},$$

  we can derive the update form for inverse of B which we define as $C = B^{-1}$:

$$C^+ = (I - \frac{sy^T}{y^T s})C(I - \frac{ys^T}{y^T s}) + \frac{ss^T}{y^T s}$$

- BFGS update is still quite cheap, $O(n^2)$ per update
- BFGS preserves positive definiteness under appropriate conditions:

  $y^T s = (\nabla f(x^+) - \nabla f(x))^T (x^+ - x) > 0$ (A sufficient condition could be strict convexity)
  $C \succeq 0$,

  Given the above conditions we consider

$$x^T C^+ x = (x - \frac{s^T x}{y^T s} y)^T C (x - \frac{s^T x}{y^T s} y) + \frac{(s^T x)^2}{y^T s},$$

  both terms are nonnegative, second term is zero when $s^T x = 0$, and in that case the first term is only zero when $x = 0$.

17.1.4 Davidon-Fletcher-Powell (DFP) Update

- Rank-two update directly on inverse C: $C^+ = C + auu^T + bvv^T$
- Secant equation $s = C^+ y$, setting $u = s, v = Cy$ and solving for a, b gives

$$C^+ = C - \frac{Cyy^T C}{y^T Cy} + \frac{ss^T}{y^T s}$$

- Preserves positive definiteness of Hessain, $O(n^2)$ computation, but not often used anymore.

17.2 Broyden Class

All of the methods discussed so far are special cases of the ”Broyden Class”

Theorem 17.1 *The* ***Broyden class*** *is the convex combination of the previous classes, BFGS, and DFP. Exactly, it is*

$$B^+ = (1-\phi)B^+_{BFGS} + \phi B^+_{DFP}, \phi \in \mathbb{R}$$

for an interpolating factor ϕ

The Broyden class therefore generalizes both BFGS and DFP trivially.

While you could always (non-interestingly) create a more general collection this way, the Broyden class also generates SR1 for a particular value of ϕ, namely

$$\phi = y^T s/(y^T s - s^T Bs)$$

Although not discussed, the slides also mention that with

$v = y/(y^T s) - Bs/(S^T Bs)$, we can rewrite as

$$B^+ = B - (Bss^T B)/(s^T Bs) + (yy^T)/(y^T s) + \phi(s^T Bs)vv^T$$

The slides note that BFGS corresponds to $\phi = 0$ and DFS corresponds to $\phi = 1$, which can be trivially read off of the equation.

17.2.1 Student Question 1

Student question (a little hard to hear): how to pick a ”next point”, which seemed to correspond to picking a value of ϕ. Professor answered that it was more of a generalization step than an extrapolation tool. Notes that you can sort of think of the class as a ”rank 3 update” that reduces to rank 2 when terms cleverly cancel.

Student clarified asking about intuition between picking the next term as in the previous algorithms, professor answers that the intuition is centered around picking particular rank updates, as motivated the previous slides.

17.2.2 Student Question 2

Student question: is strict convex assumption a strong one? Professor says is is, and should be expected to be. Explains Newton’s method works with strict convexity so approximations to Newton’s method should co-opt that assumption

17.3 Convergence Analysis

No proofs offered.

Methods attain **superlinear convergence**. Seen linear for gradient descent and quadratic for Newton's method.

17.3.1 Review

Reminder (written in Professor's notes):

Linear convergence:

$$\|x^k - x^*\| <= c\|x^{k-1} - x^*\|$$

Quadratic convergence

$$\|x^k - x^*\| <= c\|x^{k-1} - x^*\|^2$$

Superlinear

$$\|x^k - x^*\| <= c_k\|x^{k-1} - x^*\|$$

where c_k goes to 0 as k tends to infinity. Not *better* than quadratic, but does tend to 0.

Analysis would look at iterates similar to the argument in Newton's method.

Note: Limited memory quasi newton's method **do not** have this property.

Slides add (though are only briefly shown in class) that we are assuming ∇f is Lipschitz, that f is strongly convex, and that $\nabla^2 f$ is Lipschitz. These are the same conditions as the analysis of Newton's method.

17.3.2 Example

Example shows two graphs where quasi newton does better than Newton's in toy example. The function minimized is

$$c^T x - \sum_{i=1}^{m} \ln(b_i - a_i^T x)$$

the dimensionality of x is 100, 500 terms in sum. Newton's method converges in 12 iterations to 0; BFGS takes 150. Graphs have similar shape.

But, each newton is n^3 (100^3) but each BFGS is n^2. So BFGS used 10x fewer updates for the same convergence.

Although this is a toy example, professor claims that this is a pattern known in data analysis.

17.4 Implicit-Form Quasi-Newton

Motivation here is leading to limited memory quasi newton. "Workhorse" of modern optimization.

Basic idea if n is large (e.g., millions), just forming matrix C is intractable, even for BFGS.

Strategy: iterate form of BFGS updates. After some number of updates, we don't have to evaluate C directly; we can look at what C was originally, and apply all of those rank 2 updates from the past.

If the number of steps is much less than the dimensionality, we shouldn't have to form C.

So, maintain all the previous y and s (difference in gradient and difference in iterates) and compute solutions to quasi newton recursively.

Observation:

$$C^{+} = (1 - sy^T/y^Ts)C(I - ys^T/y^Ts) + ss^Ty^Ts$$

this falls from the definition.

Derivation left from slides.

17.4.1 IFQN Algorithm

Algorithm is, noting that C^{+} is computed in two loops of length k where C^{+} is the approximation to the inverse Hessian after k iterations,

```
1 Let $q = -\nabla fx(k)$
2 For $i = k-1, ...0$
    a Compute $\alpha_i = (s^{(i)^T} q/((y^{(i)^T} s^{(i)})$
    b Update $q = q - \alpha y^{(i)}$
3 Let $p = C^{(0)}q$
4 For $i = 0, ..., k-1$
    a Compute $\beta = (y^{(i)} p/y^{(i)^T} s^{(i)})$
    b Update $p = p + (\alpha_i - \beta)s^{(i)}$
5 return $p$
```

17.4.2 Summary (from Prof. Notes)

Explicit form requires n^2 memory (store whole C) and time. After k steps, $O(kn^2)$ time

Implicit (Some derivation very lightly sketched in professors notes) In total, $O(kn)$ memory, $O(k^2n)$ (total running over k iterations). If $k < n$, gain here

17.4.3 Student question

Note that on the graph before, this probably would be a bad idea since we don't observe that $k < n$. Professor notes, it often isn't great in practice; we often need more than n steps so algorithmic is more pedagogical than practical

17.5 LMBFGS

Note that first two letter are acronym; latter 4 are people's names. Big idea: **only look back m steps**. Time complexity $O(knm)$ for an effectively constant (chosen) m; same complexity as gradient descent. Despite losing some formal convergence guarantees, is considered "state of the art" for many tasks (least-squares at a large scale, e.g.). Memory $O(nm)$

Algorithm is over-viewed on slide 21, but (probably because it does not differ substantially from previous algorithm in main steps) is not discussed step-by-step

17.5.1 Out of time

Stochastic Newton's methods exist. Recent papers (referenced on slides as Byrd 2015) attempt them, but "jury is still out".