10-725/36-725: Convex Optimization **Fall 2019**

Lecture 17: October 23rd Quasi Newton Method

Lecturer: Ryan Tibshirani *Scribes: Danlei Zhu, John Fang, Swaminathan Gurumurthy*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

This lecture's notes illustrate some uses of various LaTeX macros. Take a look at this and imitate.

17.1 Motivation for Quasi-Newton Method

Recall that for unconstrained smooth convex optimization problem

$$\min_x f(x)$$

Gradient descent has global linear convergence while Newton's method has (local) quadratic convergence. But gradient descent iteration costs $\mathcal{O}(n)$ while Newton iteration costs $\mathcal{O}(n^3)$.

Quasi-Newton has superlinear convergence but each iteration costs $\mathcal{O}(n^2)$. In theory, n Quasi-Newtonian steps is as expensive as one Newton step. However, Quasi-Newton methods are typically far cheaper in practice.

17.2 Quasi-Newton Template

Let $x^{(0)} \in \mathbb{R}^n$, $B^{(0)}$ positive definite. For $k = 1, 2, 3, \ldots$ repeat

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

Requirements for $B^{(k)}$

- **Secant Equation** Motivated by the one dimensional case:

$$F'(x^+) = \frac{F(x^+) - F(x)}{x^+ - x}$$

we want $B^{(k)}$ to satisfy:

$$\nabla f(x^{(k)}) - \nabla f(x^{(k-1)} = B^{(k)}s^{(k-1)}$$

Let $y = \nabla f(x^+) - \nabla f(x)$ then equivalently we can write the condition as

$$y = B^+ s$$

- B^+ is symmetric
- B^+ is "close" to B
- B positive definite $\implies$ B^+ positive definite

17.3 Updating B^+

17.3.1 Symmetric Rank One Update (SR1): Failed Attempt

Rank one update is of the form

$$B^+ = B + auu^T$$

And the secant equation $B^+s = y$ yields

$$(au^Ts)u = y - Bs$$

This only holds if u is parallel to $y - Bs$. Put $u = y - Bs$, then the above gives $a = \frac{1}{(y-Bs)^Ts}$

$$B^+ = B + \frac{(y - Bs)(y - Bs)^T}{(y - Bs)^Ts} \tag{17.1}$$

To solve $B^+s^+ = -\nabla f(x^+)$ efficiently, we need to propagate inverses: $C = B^-1$ to $C^+ = (B^+)^{-1}$ using **Sherman-Morrison Formula**

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

hence the inverse is updated by

$$C^+ = C + \frac{(S - Cy)(S - Cy)^T}{(S - Cy)^Ty} \tag{17.2}$$

The shortcoming of SR1 is that **it does not preserve positive definiteness**.

17.3.2 Broyden-Fletcher-Goldfarb-Shanno Update (Rank Two)

Try rank two update of the form

$$B^+ = B + auu^T + bvv^T$$

By secant equation $y = B^+s$ we have

$$y - Bs = (au^Ts)u + (bv^Ts)v$$

Then putting $u = y, v = Bs$ we solve for a, b and get

$$B^+ = B - \frac{Vss^TB}{s^TBs} + \frac{yy^T}{y^Ts} \tag{17.3}$$

This is called Broyden-Fletcher-Goldfarb-Shanno (**BFGS**) update.

Same as in rank one update, we hope to propagate the inverse, by **Woodbury Formula**

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)VA^{-1}$$

Then in our case we have the update for the inverse C:

$$C^+ = (I - \frac{sy^T}{y^Ts})C(I - \frac{ys^T}{y^Ts}) + \frac{ss^T}{y^Ts} \tag{17.4}$$

Cost of BFGS update is quite cheap. We are multiplying a matrix with a rank 1 matrix, which can be re-written as multiplying a matrix and a vector. Hence, the cost is only $\mathcal{O}(n^2)$, as opposed to the usual $\mathcal{O}(n^3)$ for matrix multiplication. And BFGS update **preserves positive definiteness** i.e. B positive definite $\implies$ B^+ positive definite. Equivalently C positive definite $\implies$ C^+ positive definite. To see this we check

$$x^TC^+x = (x - \frac{sx}{y^Ts}y)^TC(x - \frac{s^Tx}{y^Ts}y) + \frac{(s^Tx)2}{y^Ts}$$

The first term on right hand side is positive since C is positive definite, and the second term is positive by monotonicity of convex function

$$y^Ts = (\nabla f(x^+) - \nabla f(x))^T(x^+ - x) \geq 0$$

Hence we have $x^TC^+x > 0$. So C^+ is positive definite.

17.3.3 Davidon-Fletcher-Powell (Alternate Rank Two)

In the BFGS method described earlier, we described a rank-two update for B, then applied the Woodbury formula to find how to apply a rank-two update to the inverse C. However, one could also imagine that we instead applied the rank-two update to C first.

$$C^+ = C + auu^T + bvv^T$$

Then the by multiplying by y and using the secant equation $s = C^+y$, we get

$$C^+ = C - \frac{Cyy^TC}{y^TCy} + \frac{ss^T}{y^Ts}$$

and applying Woodbury shows the update for B is

$$B^+ = \left(I - \frac{ys^T}{y^Ts}\right)B\left(I - \frac{sy^T}{y^Ts}\right) + \frac{yy^T}{y^Ts}$$

We can see that these equations are mirrors of equations 17.3 and 17.4. This update is called **David-Fletcher-Powell (DFP)**. Just like in BFGS, the updates are $\mathcal{O}(n^2)$ and preserve positive definiteness. However, DFP is not as popular as BFGS.

17.3.4 Alternate Motivation for DFP

Note that $B^+ \succ 0$ and $B^+s = y$ imply

$$y^Ts = s^Tb^+s > 0$$

This is called the curvature condition, and it implies taht there exists $M \succ 0$ s.t. $Ms = y$.

There is another way to derive the DFP update. The motivation is the question "what is the minimal amount that you can move from B while maintaining certain conditions such as the secant condition?". Formally,

$$\begin{aligned} \min_{B^+} \quad & \|W^{-1}(B^+ - B)W^{-T}\|_F \\ \text{s.t.} \quad & B^+ = (B^+)^T \\ & B^+ s = y \end{aligned}$$

where W is non-singular and $WW^T s = y$. Solving this is actually the same as DFP.

17.4 Other Quasi-Newtonian Updates

So far, we have discussed SR1, DFP and BFGS. However, these are only a few of the many possible quasi-Newtonian updates.

17.4.1 Broyden Class

One particular set of quasi-Newtonian updates is called the **Broyden Class**. These can be thought of as updates that are some linear interpolation of BFGS and DFP. Formally, the Broyden class of updates is defined by

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

By putting $v = \frac{y}{y^T s} - \frac{Bs}{s^T Bs}$, we can re-write above as

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

Note that the three methods that we have described all belong to the Broyden class

- $\phi = 0$: BFGS
- $\phi = 1$: DFP
- $\phi = \frac{y^T s}{y^T s - s^T Bs}$: SR1

17.5 Convergence Analysis

We make the same assumptions as we did for Newton's method for the convergence analysis of DFP and BFGS, namely

- f is convex, twice differentiable and has $\text{dom}(f) = \mathbb{R}^n$
- ∇f is Lipschitz with parameter L
- f is strongly convex with parameter m
- $\nabla^2 f$ is Lipschitz with parameter M

Under these assumptions, both DFP and BFGS converge globally with backtracking line search. Furthermore for all $k \geq k_0$,

$$\|x^{(k)} - x^*\|_2 \leq c_k \|x^{(k-1)} - x^*\|_2$$

where $c_k \to 0$ and $k \to \infty$. Here k, c_k depend on L, m, M. This is called local superlinear convergence.

17.6 Implicit-Form Quasi-Newton

Motivation: Quasi-Newton methods are significantly cheaper to compute than the original newton updates, but still take $O(n^2)$ space and time. If n is large, storing or even forming the matrix C is intractable. **Idea:** Instead of storing and computing C explicitly, we could use an implicit version by maintiaining all (y,s) pairs (difference in gradient and difference in iterates pairs). This is especially useful if the number of updates $k << n$.

Reminder that,

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

Thus, we should be able to compute C^+, if we have all (y,s) pairs. But we don't want to compute it. We instead want to compute C^+g directly at each iteration.

This can be done using two loops of length k using the algorithm shown below:

17.6.1 IFQN Algorithm

1. Let $q = -\nabla f(x^k)$
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 $\beta = (y^{(i)})^T p/((y^{(i)})^T s^{(i)})$
   (b) Update $p = p + (\alpha_i - \beta)s^{(i)}$
5. return p

Complexity Analysis
Explicit form requires n^2 memory and time. Thus after k steps, it take $O(kn^2)$ time.
Implicit form (IFQN algo) takes $O(kn)$ memory, since each (y, s) pairs each of size n need to be stored for k iterations. Similarly, it takes total $O(k^2n)$ time after k steps. Thus, there are gains if $k < n$.

17.7 LBFGS

Motivation:
We saw that the IFQN algorithm has better computational and space complexity for $k < n$, however becomes inefficient as k approaches n. In fact, in a lot of practical problems we observe that $k > n$. **Idea:**

Instead of using all k steps to compute C, we can use only the last m steps and assume an identity value of C before that. This provides a total computational complexity of $O(knm)$ and memory $O(nm)$. For a constant m, this effectively approaches the complexity of gradient descent. This method is considered the 'state of the art' for many task (e.g. large scale least squares).

The algorithm is provided below:

17.7.1 LBFGS Algorithm

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

$\hat{C}^{(k-m)}$ is not stored but is a guess using some prior knowledge. A popular choice is $\hat{C}^{(k-m)} = I$

17.8 Stochastic quasi-Newton methods

An obvious extension to the previous formulation is to consider a stochastic approximation of C.

$$x^{(k)} = x^{(k-1)} - t_k C^{(k-1)} \nabla f(x^{(k-1)}, \zeta_k)$$

where ζ_k is the noise variable introducing the stochasticity.

But it's unclear if this is useful,

1. Can have at best sublinear convergence. Is the additional overhead of quasi-Newton worth it, over plain SGD.
2. The computed C depends on the gradient estimates which are themselves noisy

Due to the above considerations, it is unclear if stochastic LBFGS would be useful in practical scenarios. This is still an open research problem and is being investigated with much interest in the community

References

[T19] R. Tibshiranih , "Quasi Newton Method," *Lecture Slides for 10-725 Convex Optimization*, 2019.