Indian Institute of Technology Kanpur
CS774 Optimization Techniques

Scribe: Hrusikesha Pradhan
Instructor: Purushottam Kar
Date: October 3, 2016

LECTURE 16

Quasi-Newton Methods

1 Introduction

In the previous lecture we discussed about why we shifted to Quasi-Newton methods because of evaluation of Hessian is impractical and costly. We discussed about some variants of Quasi-newton methods which included Secant method, Broyden's methods and we also discussed about Sherman-Morrison-Woodbury formula which helps in computing the inverse of the sum of an invertible matrix and outer product of two vectors.

In this lecture we will discuss about Davidon-Fletcher-Powell (DFP) formula which was the first quasi-newton methodQN to generalize the secant method to a multidimensional problem and we will move to BroydenFletcherGoldfarbShanno (BFGS) algorithm. We will also look at L-BFGS, which is a limited-memory version of BFGS that is particularly suited to problems with very large numbers of variables. At last we will just introduce the basic concepts which are required before moving to our next lecture, i.e. Stochastic Optimization.

2 Davidon-Fletcher-Powell (DFP) Rule

The first quasi-Newton method is DFP, named after Davidson, who discovered it in 1959, and Fletcher and Powell, who explored its mathematical properties over the next few years. Instead of computing the true Hessian as in Newtons method, we will use an approximation that is based on the change in gradient between iterations. The primary advantage is that we dont have to compute the exact Hessian at each point, which may be computationally expensive. The idea is to characterize the approximate Hessian **H** via several properties, and then derive an expression for the unique **H** that satisfies the properties. The properties are:

- H_{t+1} must be symmetric.
- When we form a quadratic model using H_{t+1}, as above, the gradient of the model must equal the functions gradient at the points x_t and x_{t+1}. Given a function $f(x)$, its gradient ∇f , and positive definite Hessian matrix **H** , the Secant rule can be stated as,
  $H_{t+1}(x_{t+1} - x_t) = \nabla f(x_{t+1}) - \nabla f(x_t)$

The DFP formula finds a solution that is symmetric, positive definite and closest to the current approximate value of $\mathbf{H}_t$. For a quadratic objective, it simultaneously generates the directions of the conjugate gradient method while constructing the inverse Hessian. At each step the inverse Hessian is updated by the sum of two symmetric rank one matrices.

$$H_{t+1} = \Big(I - \beta_t \Delta g^t (\Delta x^t)^T\Big) H_t \Big(I - \beta_t \Delta x^t (\Delta g^t)^T\Big) + \beta_t \Delta g^t (\Delta g^t)^T \tag{1}$$

where

$$\Delta x^t = x_{t+1} - x_t$$
$$\Delta g^t = \Delta f(x_{t+1}) - \Delta f(x_t)$$
$$\beta_t = \frac{1}{< \Delta x^t, \Delta g^t >}$$

$\mathbf{H}_t$ is symmetric and positive definite matrix Using Sherman Morrison Woodbury formula, the corresponding update to the inverse of $\mathbf{H}_t + 1$ which is denoted by $\mathbf{S}_t + 1$ is given by,

$$S_{t+1} = S_t - \frac{(S_t \Delta x^t)(S_t \Delta x^t)^T}{< \Delta x^t, S_t \Delta x^t >} + \beta_t \Delta g^t (\Delta g^t)^T \tag{2}$$

The final issue to address for DFP is how to choose the initial Hessian H_0. One option is to use the true Hessian at the initial point (so we still have to compute it once), or a diagonal approximation to the true Hessian. An easier option that works fine in practice is to use a scalar multiple of the identity matrix, where the scaling factor is chosen to be in the range of the eigenvalues of the true Hessian.

3 Broyden-Fletcher-Goldfarb-Shanno (BFGS)

DFP was the first quasi-Newton method, but it was soon superceded by BFGS, which is considered to be the most effective quasi-Newton method. BFGS is named for the four people who (independently!) discovered it in 1970: Broyden, Fletcher, Goldfarb and Shanno. It is actually the same as DFP with a single, very elegant modification: instead of approximating the Hessian, H_t, we approximate its inverse S_t. This has the advantage that we dont need to solve a linear system to get the search direction, but only do a matrix/vector multiply. It is also more numerically stable, and has very effective self-correcting properties not shared by DFP, which may account for its superior performance in practice.

Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula update of $\mathbf{H}_t$ is obtained by taking the complimentary formula of the DFP formula, thus:

$$H_{t+1} = H_t - \frac{(H_t \Delta x^t)(H_t \Delta x^t)^T}{< \Delta x^t, H_t \Delta x^t >} + \beta_t \Delta x^t (\Delta x^t)^T \tag{3}$$

By taking the inverse, the BFGS update formula for $\mathbf{S}_{t+1}$ $(i.e., \mathbf{H^{-1}_{t+1}})$ is obtained using Sherman-Morrison formula as,

$$S_{t+1} = \Big(I - \beta_t \Delta x^t (\Delta g^t)^T\Big) S_t \Big(I - \beta_t \Delta g^t (\Delta x^t)^T\Big) + \beta_t \Delta x^t (\Delta x^t)^T \tag{4}$$

4 L-BFGS

BFGS is a very effective optimization algorithm that does not require computing the exact Hessian, or finding any matrix inverses. However, it is not possible to use BFGS on problems with a very high number n of variables (millions, say), because in that case it is impossible to store or manipulate the approximate inverse Hessian S, which is of size n^2. L-BFGS (limited-memory BFGS) solves this problem by storing the approximate Hessian in a compressed form that requires storing only a constant multiple of vectors of length n. In particular, L-BFGS only remembers updates from the last m iterations, so information about iterates before that is lost. The way this is accomplished is by unrolling the Hessian update from BFGS as follows.

Defining $V_t = \left(I - \beta_t \Delta g^t \Delta x^{t^T}\right)$, the BFGS update can now be written as,

$$S_{t+1} = V_t^T S_t V_t + \beta_t \left(\Delta x^t\right)\left(\Delta x^t\right)^T \tag{5}$$

Rolling back one step gives,

$$S_{t+1} = V_t^T \left(V_{t-1}^T S_{t-1} V_{t-1} + \beta_{t-1}\left(\Delta x^{t-1}\right)\left(\Delta x^{t-1}\right)^T\right) V_t + \beta_t \left(\Delta x^t\right)\left(\Delta x^t\right)^T$$

$$= V_t^T V_{t-1}^T S_{t-1} V_{t-1} V_t + V_t \beta_{t-1}\left(\Delta x^{t-1}\right)\left(\Delta x^{t-1}\right)^T V_t + \beta_t \left(\Delta x^t\right)\left(\Delta x^t\right)^T$$

Lets take $X_t = \Delta x^t \bigotimes \Delta x^t$ (where $\bigotimes$ denotes outer product), then S_{t+1} can be written as,

$$S_{t+1} = V_t^T V_{t-1}^T S_{t-1} V_{t-1} V_t + \beta_{t-1} V_t^T X_{t-1} V_t + \beta_t X_t$$

Further expanding S_{t+1}, we get

$$S_{t+1} = V_t^T V_{t-1}^T V_{t-2}^T S_{t-2} V_{t-2} V_{t-1} V_t + \beta_{t-2} V_t^T V_{t-1}^T X_{t-2} V_{t-1} V_t + \beta_{t-1} V_t^T X_{t-1} V_t + \beta_t X_t$$

In BFGS S_t is unrolled upto S_{t-1}, S_{t-2},.....upto ∞ whereas in L-BFGS S_t is unrolled upto S_{t-1}, S_{t-2}...upto 5 or 6 steps. The influence of previous Hessian is upto 5 or 6 steps and it is a time series analysis. In L-BFGS, S_{t-m} plays the role of S_0 in BFGS where m denotes the number of steps we are unrolling. It is exactly as if we started at the point x_{t-m} and ran m iterations of BFGS, starting with the initial approximation S_{t-m} . If S_{t-m} has a simple form, like a multiple of the identity matrix, then we can easily compute the search direction. Note that unlike the S_0 of BFGS, S_{t-m} is allowed to vary from iteration to iteration. For example, we might use a multiple of the identity based on the most recent change in the gradient.

Since BFGS (and hence L-BFGS) is designed to minimize smooth functions without constraints, the L-BFGS algorithm must be modified to handle functions that include non-differentiable components or constraints. A popular class of modifications are called active-set methods, based on the concept of the active set. The idea is that when restricted to a small neighborhood of the current iterate, the function and constraints can be simplified. These variants of L-BFGS are explained belowL-BFGS:

- **L-BFGS-B:** The L-BFGS-B algorithm extends L-BFGS to handle simple box constraints (aka bound constraints) on variables; that is, constraints of the form $L_i \leq X_i \leq U_i$ where L_i and U_i are per-variable constant lower and upper bounds, respectively. The method works by identifying fixed and free variables at every step (using a simple gradient method), and then using the L-BFGS method on the free variables only to get higher accuracy, and then repeating the process.

- **OWL-QN:** Orthant-wise limited-memory quasi-Newton (OWL-QN) is an L-BFGS variant for fitting l_1 −regularized models, exploiting the inherent sparsity of such models. It minimizes functions of the form

  $f(x) = g(x) + \lambda \left\|X\right\|_1$

  where g is a differentiable convex loss function. The method is an active-set type method: at each iterate, it estimates the sign of each component of the variable, and restricts the subsequent step to have the same sign. Once the sign is fixed, the non-differentiable $\left\|X\right\|_1$ term becomes a smooth linear term which can be handled by L-BFGS. After an L-BFGS step, the method allows some variables to change sign, and repeats the process.

- **O-LBFGS:** It is an online approximation to both BFGS and L-BFGS. Similar to stochastic gradient descent, this can be used to reduce the computational complexity by evaluating the error function and gradient on a randomly drawn subset of the overall dataset in each iteration. It has been shown that O-LBFGS has a global almost sure convergence while the online approximation of BFGS (O-BFGS) is not necessarily convergent.

5 Stochastic Optimization

A stochastic optimization problem can be formulated as,

$$\underset{X \in \mathcal{C}}{min} f(X) where f(X) = \underset{w}{\mathbf{E}} f(X) \left[F(X, w)\right]$$

But before going into the details of Stochastic optimization we will introduce some basic concepts related to probability theory. There are lots of phenomena in nature, like tossing a coin or tossing a die, whose outcomes cannot be predicted with certainty in advance, but the set of all the possible outcomes is known. These are what we call random phenomena or random experiments. Probability theory is concerned with such random phenomena or random experiments.

- **Sample Space (Universe):** The set of all the possible outcomes of a random experiment is called the sample space or Universe and is denoted by Ω.
- **Event:** Any subset E of the sample space Ω is called an event, i.e.$E \subset \Omega$.
- **$\sigma-$algebra/Event Space:** A $\sigma-$algebra on Ω is a collection F of subsets of satisfying:
    1. $\Omega \in F$
    2. $E \in F \Rightarrow E^c \in F$, where E^c=$\Omega \backslash E$
    3. E_1, E_1,.....$\in F$, $\bigcup E_i \in F$
- **Probability Mass Function:** Suppose that P is a Probability mass function denoted as P:$F \rightarrow R_+$. Then it satisfies the following properties.
    1. $P\left(\Omega\right) = 1$
    2. If A_1,A_2,...... are pairwise disjoint events then,

    $$P\left(\underset{i}{\cup} A_i\right) = \underset{i}{\sum} P\left(A_i\right)$$
- **Random variable:** A random variable X:$\Omega \rightarrow R$ is a measurable function from the set of possible outcomes Ω to some set E.
- **Measurability:** For real observation space, the measurability of a random variable X:$\Omega \rightarrow R$ verified by $\{w, X\left(w\right) \leq t\} \in F$, $\forall\ t \in R$
- **Expectation of a random variable:** The expected value of a discrete random variable is the probability-weighted average of all possible values. In other words, each possible value the random variable can assume is multiplied by its probability of occurring, and the resulting products are summed to produce the expected value. The same principle applies to a continuous random variable, except that an integral of the variable with respect to its probability density replaces the sum.

    $$\begin{aligned}\mathbf{E}X &= \underset{t}{\sum} P\left(X = t\right)\left(RiemannIntegral\right)\\ &= \underset{t}{\sum} P\left(X \leq t\right)\left(LebesgueIntegral\right)\end{aligned}$$

    Some properties of expectation areExpectation:

    1. **Constants:**The expected value of a constant is equal to the constant itself; i.e., if c is a constant, then **E**C=C.

2. **Linearity:** The expected value operator (or expectation operator) is linear in the sense that $\mathbf{E}\big(X + Y\big) = \mathbf{E}X + \mathbf{E}Y$
3. **Monotonicity:** If X and Y are random variables such that $X \leq Y$ almost surely, then $\mathbf{E}X \leq \mathbf{E}Y$.

- **Conditional Expectation:** The conditional expectation of a random variable is another random variable equal to the average of the former over each possible "condition". Mathematically it can be written as,

$$\mathbf{E}[X \mid Y = y] = \sum_{w \in \Omega} X\big(w\big) P\big(w \mid Y = y\big)$$

References

Quasi-newton methods. http://www.stat.cmu.edu/ ryantibs/convexopt-F13/lectures/11-QuasiNewton.pdf. Accessed: 2010-09-30.

Expectation. Expected value — wikipedia, the free encyclopedia. URL `https://en.wikipedia.org/w/index.php?title=Expected_value&oldid=744272760`. [Online; accessed 14-October-2016].

L-BFGS. Limited-memory bfgs — wikipedia, the free encyclopedia. URL `https://en.wikipedia.org/w/index.php?title=Limited-memory_BFGS&oldid=744137838`. [Online; accessed 13-October-2016].