10-725: Optimization **Fall 2012**

Lecture 12: October 4

Lecturer: Geoff Gordon/Ryan Tibshirani *Scribes: Huan-Kai Peng, Hao-Chih Lee*

This lecture covers the second half of Newton's method. Four topics are covered: handling equality constraints, examples, convergence, and variants.

12.1 Newton's method under equality constrains

The problem we consider in this section is of the form

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) = 0. \end{aligned} \tag{12.1}$$

Here we assume that $f(x):\ R^d \to R$ be strictly convex and at least twice differentiable, and that $h(x):\ R^d \to R^k$ is differentiable. We will also use the following convenient terminologies:

- $g(x):\ R^d \to R^d$ denotes the gradient of $f(x)$.
- $H(x):\ R^d \to R^{d\times d}$ denotes the Hessian of $f(x)$.
- $J(x):\ R^d \to R^{k\times d}$ denotes the Jacobian of $h(x)$.

The following is the key theorem that we'll provide an intuitive explanation immediatly.

Theorem 12.1 *The minimizer x^* of the problem 12.1 satisfies*

$$g(x^*) = J(x^*)^T \lambda$$

for some $\lambda \in R^k$.

12.1.1 Intuition in an 1-D case

Consider the plot in page 4 of the slides, where the contour plot represents $f(x)$ and the green curve represent the surface $h(x) = 0$. By looking at the plot (and a little imagination), we can understand that the local minimizer x^* must happen at places where the contour is parallel to the green curve $h(x)$. Otherwise, we would be able to move x^* along $h(x)$ a little bit and find a better minimizer. This intuition can be interpreted in 3 different ways:

1 The contour of $f(x)$ is tangent to $h(x)$ at x^*.

2 $g(x^*)$ is normal to $h(x)$ at x^*.

3 $g(x^*)$ is parallel to $J(x^*)$.

In either way, we have

$$g(x^*) = \lambda J(x^*)$$

, which is the 1-d version of Theorem 12.1.

12.1.2 Newton's method under equality constraints

In unconstrained cases (from previous lectures), the Newton's method can be summarized as:

$$x^+ = x + \Delta x_{nt} \tag{12.2}$$

where $\Delta x_{nt} = -H(x)^{-1}g(x)$ is called the Newton's step. A nice interpretation of the Newton's step is to set it by minimizing the second-order approximation of f at x:

$$\Delta x_{nt} = \arg\min_{\Delta x \in R^d} \hat{f}(x + \Delta x) = \arg\min_{\Delta x \in R^d} f(x) + g(x)^T \Delta x + \frac{1}{2}\Delta x^T H(x) \Delta x \tag{12.3}$$

By setting the derivative of $\hat{f}$ to zero, we will obtain $\Delta x_{nt} = -H(x)^{-1}g(x)$.

In the equality-constrained case, the Newton's step need to be calculated differently. In particular, Equation 12.3 needs to incorporate the constraint, that is, Δx_{nt} equals to:

$$\begin{aligned} \arg\min_{\Delta x \in R^d} \quad & f(x) + g(x)^T \Delta x + \frac{1}{2}\Delta x^T H(x) \Delta x \\ \text{s.t.} \quad & h(x) = 0. \end{aligned} \tag{12.4}$$

To solve Δx_{nt}, we claim that it suffices to solve the linear system:

$$\begin{bmatrix} H(x) & J(x)^T \\ J(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \lambda \end{bmatrix} = \begin{bmatrix} -g(x) \\ -h(x) \end{bmatrix}. \tag{12.5}$$

To explain why, let's expand the matrix form into two equations:

$$H(x)\Delta x_{nt} + J(x)^T \lambda = -g(x) \tag{12.6}$$

$$J(x)\Delta x_{nt} = -h(x). \tag{12.7}$$

Now we can see that Equation 12.6 comes directly from Theorem 12.1, whereas Equation 12.7 comes from Newton's update for solving $h(x) = 0$, which is $\Delta x_{nt} = x^+ - x = -J(x)^{-1}h(x)$.

The good thing about Equation 12.5 is that it enable us to solve Δx_{nt} much more easily than Equation 12.4. Assuming that f is strictly convex, we know H has full-rank; assuming independent constraints, we also know J has full-row-rank. Thus the matrix in the LHS is invertible. The Newton's step can thus be calculated by direct matrix inversion, and be plugged into Equation 12.2 for the update rule. In cases when H is not stable to inverse (e.g., when H is sparse), we can still use Gaussian Elimination to solve the linear system without inversion. The later bundle adjustment problem is one such example.

12.2 Examples

12.2.1 Newton's step in linear equality constraints

When $h(x) = Ax$ where $A \in R^{k \times d}$ and assume that the current solution is feasible (h(x) = 0), then linear system in Equation 12.5 becomes:

$$\begin{bmatrix} H(x_k) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -g \\ 0 \end{bmatrix}.$$

Suppose that $H(x)$ is invertible (e.g., strictly convex $f(x)$) and A is full row-rank, then the linear system is solvable. Hence

$$\begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} H(x_k) & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} -g \\ 0 \end{bmatrix}.$$

12.2.2 Exponential Family

We'll derive the MLE for exponential family using Newton's method. Let $\{x_i\}_{i=1}^N$ be i.i.d samples with a density in exponential family, i.e.

$$p(x_i|\theta) = exp(\theta^T x - A(\theta))$$

Where

$$A(\theta) = \ln \int_X exp(\theta^T x) dx$$

To estimate $\hat{\theta}_{mle}$, we define the negative log-likelihood as

$$L(\theta) = -\ln \Pi_{i=1}^N p(x_i|\theta) = -\sum_i (\theta^T x_i - A(\theta))$$

Let's calculate its gradient, Hessian matrix and the Newton's step.

$$dL(\theta) = -N \cdot \frac{1}{N} d(\sum_i (x^T \theta - A(\theta)))$$

$$= -N(\frac{1}{N} \sum_i x^T \theta - A'(\theta)) d\theta$$

$$= N(-\bar{x}^T + A'(\theta)) d\theta$$

$$\Rightarrow \nabla L(\theta) = N(\nabla A(\theta) - \bar{x})$$

$$\Rightarrow \nabla^2 L(\theta) = N \nabla^2 A(\theta)$$

Now we further figure out $\nabla A(\theta)$ and $\nabla^2 A(\theta)$.

$$A(\theta) = \ln \int_X exp(\theta^T x) dx$$

$$A'(\theta) = \frac{1}{\int_X exp(\theta^T x) dx} \int_X x \cdot exp(\theta^T x) dx$$

$$= \int_X x p(x|\theta) dx$$

$$= E[x|\theta]$$

$$A''(\theta) = \int_X xp(x|\theta)dx$$

$$= \int_X (x \cdot exp(\theta^T x - A(\theta)))'$$

$$\int_X x \cdot exp(\theta^T x - A(\theta))(x^T - A'(\theta))$$

$$= \int_X p(x|\theta) \cdot (xx^T - x \cdot E(x|\theta))dx$$

$$= E(xx^T|\theta)) - E(x|\theta))^2$$

$$= var[x|\theta]$$

Therefore the Newton's step is

$$\Delta x = -H^{-1}g = var[x|\theta]^{-1}(\bar{x} - E(x|\theta))$$

12.2.3 Bundle adjustment for autonomous robots

In this example, we visit a case where the Hessian is sparse and therefore cannot be reliabliy inverted. In page 8 of the slides, we consider a Bundle adjustment problem as follows. Suppose we have a robot wondering in a room where K landmarks present. At each time step t, 3 types of (noisy) sensor readings are available to the robot:

- v_t: forward and sideway displacements of the robot, 2-dimensional.
- w_t: angular displacement of the robot, 1-dimensional.
- $d_t(k)$: distance from the robot to the k-th landmark, 1-dimensional.

The problem for the robot is therefore to infer the following 3 sets of variables from the above noisy sensors:

- x_t: the true location of the robot at each time step t, 2-dimensional.
- θ_t: the true orientation of the robot at each time step, 1-dimensional.
- y_k: the true location of the k-th landmark, 2-dimensional.

where θ_t is further encoded using $u_t = (\cos\theta_t, \sin\theta_t)$ for convenience. Consider the following assumptions:

- $v_t = R(u_t)(x_{t+1} - x_t) + \epsilon_{v,t}$.
- $w_t = \theta_{t+1} - \theta_t + \epsilon_{w,t}$.
- $d_t(k) = R(u_t)(y_k - x_t) + \epsilon_{k,t}$

where $R(u_t)$ is the rotation matrix:

$$\begin{bmatrix} \cos\theta_t & \sin\theta_t \\ -\sin\theta_t & \cos\theta_t \end{bmatrix} \tag{12.8}$$

and ϵ's are the noise resulted from sensors. The optimization in page 9 of the slides can then be interpreted as finding the true robot location/orientation and the true landmark locations by minimizing the sum of squared noise.

We now proceed by observing that the Hessian of the objective is sparse. Note that in the objective, each variable doesn't interact with one another in non-adjacent timesteps. That is to say, picking any two variables more than 1 timestep away, the corresponding second derivative in the Hessian will be zero. This Hessian is going to be extremely sparse, making it super unstable to inverse. This is a good case where we want to calculate the Newton step by solving a linear system instead of inverting the Hessian.

12.3 Convergence results

Assuming a strictly convex f where its Hessian is Lipschitz, the convergence of Newton's method is "quadratic" in number of iterations. That is,

- Given an error ϵ, the number of iterations needed to achieve that error is $k = O(\ln \epsilon^{-2})$.
- Given k iterations, the error shrinks as quickly as $\epsilon = O(e^{-\frac{k}{2}})$.

More specifically, the convergence of Newton's method consists of two phases. During the "damp phase", the optimization will take $O(1)$ iterations; during the "quadratic phase", the optimization takes $O(\ln \epsilon^{-2})$ iterations, which is essentially less than 6. More details about the convergence analysis of Newton's method can be found in Boyed's book on page 488.

As amazing as its quick convergence, it is important to keep in mind that the convergence result is in terms of number of iterations. Although few iterations are needed for convergence, each iteration needs to calculate and inverse the Hessian, which is typically $O(n^3)$ where n is the dimensionality. For high-dimensional problems, this is inhibitively expensive to run even one iteration. Therefore, first-order methods are still attractive in many settings.

12.4 Variants

12.4.1 Trust region

Although Newton method has fast convergence rate, it may also diverge quickly when Hessian matrix is ill-conditioned. One way to avoid "bad" steps is by introducing the trust region. For example, we may instead solve

$$(H(x) + tI)\Delta x = -g(x)$$

to find the unconstrained Newton step. Here the I is the identify matrix. The intuition here is when t is large, the Newton method would reduce to gradient descent method since the updating step is almost $-I^{-1}g = -g$. On the contrary, if t is small, we get the Newton method. So the strategy would be using large t initially to go along the gradient and small t later on to speed up the convergence. Another example of trust region is

$$(H(x) + t(I \circ H))\Delta x = -g(x).$$

The advantage of using this trust region is that it make the algorithm invariant to scaling.

12.4.2 Quasi-Newton

Quasi-Newton methods use only the information of gradients to successively estimate the Hessian matrix. For example, finite-differences may be used to approximate the Hessian matrix by gradients at nearby points. An example of quasi-Newton method is the L-BFGS method, which can often get "good enough" estimation with only a few old information.

12.4.3 Gaussian-Newton

Gaussian-Newton method solves the case that $f(x) = \sum_{i=1}^{N} \frac{1}{2}(y_i - f(x_i, \theta))^2$ where the minimization is with respect to θ. In this case we approximate $f(x_i, \theta + \Delta\theta) \approx f(x_i, \theta) + \nabla f(x_i, \theta)^T \Delta\theta$ and

$$\begin{aligned}\sum_{i=1}^{N} \frac{1}{2}(y_i - f(x_i, \theta))^2 &\approx \sum_{i=1}^{N} \frac{1}{2}(r_i(\theta) - \nabla f(x_i, \theta)^T \Delta\theta)^2 \\ &= \sum_i (r_i(\theta)^2) + P^T \Delta X + \frac{1}{2}\Delta X^T Q \Delta X.\end{aligned}$$

where $r_i(\theta) = y_i - f(x_i, \theta)$, $P = -\sum_{i=1}^{N}(y_i - f(x_i, \theta)\nabla f(x_i, \theta)$ and $Q = \sum_{i=1}^{N} \nabla f(x_i, \theta)\nabla f(x_i, \theta)^T$. Therefore the unconstrained Newton step is

$$\Delta\theta = (\sum_{i=1}^{N} \nabla f(x_i, \theta)\nabla f(x_i, \theta)^T)^{-1}(\sum_{i=1}^{N}(y_i - f(x_i, \theta)\nabla f(x_i, \theta)).$$

References

[CW87] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 1–6.