MAT 653: Statistical Simulation

Instructor: Dr. Wei Li

2023-09-07

Back **Updated:** 2023-09-07 Statistical Simulation, Wei Li

Numerical method for optimization

For some smooth function f, consider min $f(x) \to x^*$.

We'll consider the gradient method, and Newton's method (and its variants). Both methods guarantee that a stationary point of f can be found (i.e. $\nabla f(x^*) = 0$). The basic idea is

1. start the process with some initial point x_0;
2. iterate steps $x_k \to x_{k+1}$ by going downhill.
3. Repeat step 2 until the sequence of points converge to a stationary point.

For a general (non-convex) function, we run the procedure several times with different initial values x_0.

Gradient descent

Choose a direction p_k and search along the direction from the current iterate x_k for a new iterate with lower function value $f(x_{k+1}) < f(x_k)$. Consider $x_k + \alpha P_k$ (α = "step length", "learning rate", $\alpha > 0$ scalar) as the next iterate.

Fix α: Taylor approximation of f at x_k is given by

$$f(x_k + \alpha p) \approx f(x_k) + \alpha p^T \nabla f(x_k) := T(x_k, \alpha p)$$

To solve for the direction p, $\min_{||p||=1} p^T \nabla f(x_k)$, whose solution gives us the the unit direction that has the most rapid decrease in f.

$p^T \nabla f(x_k) = ||p|| \cdot ||\nabla f(x_k)|| cos(\theta)$, $0 \le \theta \le \pi \Rightarrow cos(\theta) = -1 \Rightarrow p$ is the exact opposite direction of $\nabla f(x_k)$, $p = \frac{-\nabla f(x_k)}{||\nabla f(x_k)||}$.

Note: Any p_k such that $p_k^T \nabla f(x_k) < 0$ would work. It's called "descent direction".

Algorithm

Gradient descent (fix α), set $k = 0$, given x_0.

Repeat

- $x_{k+1} \leftarrow x_k - \alpha \nabla f(x_k)$
- $k \leftarrow k + 1$

Until stopping condition is met (e.g., $||\nabla f(x_k)|| = 0$).

Newton's Method

Take a second order Taylor expansion of $T(x_k, \alpha p)$ in the direction given by αp:

$$T(x_k, \alpha p) := f(x_k) + \alpha p^T \nabla f(x_k) + \alpha^2 p^T \nabla^2 f(x_k) p/2.$$

we travel to a stationary point of this quadratic approximation by solving for a minimizer p (if f is convex) and move $x_{k+1} \leftarrow x_k + \alpha p$.

To see this, set $\alpha \equiv 1$; set the gradient of $T(x_k, \alpha p)$ to zero and solve for p:

$$\begin{aligned} FOC : &\nabla_p T(x_k, \alpha p) \overset{set}{=} 0 \\ &\nabla f(x_k) + \nabla^2 f(x_k) p = 0 \\ &\Rightarrow p = p_k = -\nabla f(x_k)^{-1} \cdot \nabla f(x_k), \text{ if } \nabla^2 f(x_k) \text{ is invertible} \\ &x_{k+1} \leftarrow x_k - \left[\nabla^2 f(x_k)\right]^{-1} \cdot \nabla f(x_k) \end{aligned}$$

If $\nabla^2 f(x_k)$ is not invertible, you can use $\left[\nabla^2 f(x_k)\right]^{\dagger}$. An alternative way is to solve:

$$\begin{aligned} &\nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0. \\ \Leftrightarrow &\nabla^2 f(x_k) x_{k+1} = \nabla^2 f(x_k) x_k - \nabla f(x_k). \\ \Rightarrow &\text{solve for } x_{k+1} \end{aligned}$$

Remark: Newton method produces a sequence of points $x_1, x_2, ...$ that minimizes a sequence of the function by repeatedly creating a quadratic approximation to the function f centered at x_k. With a quadratic approximation more closely mimicking the object function. Newton method is often more effective than the gradient method. However, the reliance on the quadratic approximation makes Newton's method more difficult to use, especially for non-convex function. $\nabla^2 f(x_k)$ may not be invertible, so can use $\left[\nabla^2 f(x_k)\right]^{+}$ or solving for x_{k+1} from this system as above. However, both approaches do not guarantee that $p_k^T \nabla^2 f(x_k) < 0$.

Quasi Newton Method

$$\begin{aligned} &p_k = -\left[\nabla^2 f(x_k)\right]^{-1} \cdot \nabla f(x_k) \qquad \text{Newton Method} \\ &p_k = -B_k^{-1} \nabla f(x_k) \qquad \text{Quasi -Newton Method} \end{aligned}$$

here, B_k is some approximation to $\nabla^2 f(x_k)$. Note that

$$\begin{aligned} &\nabla f(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) + o(||x_{k+1} - x_k||) \\ \Leftrightarrow\ &\nabla^2 f(x_k)(x_{k+1} - x_k) \approx \nabla f(x_{k+1}) - \nabla f(x_k) \end{aligned}$$

B_k is required to satisfy: (1). B_k is symmetric; (2). $B_k(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$ (secant equation).

Let $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. If $B_0 > 0$ and $s_k^T y_k > 0$ (curvature condition), the BFGS update produces positive definite approximation to Hessian. The curvature condition is satisfied if the Wolfe conditions are imposed on the line search for the step length α_k (see below).

BFGS: The idea of BFGS method is to update B_k using $B_{k+1} = B_k + \alpha uu^T + \beta vv^T$. Imposing the secant condition, $B_k s_k = y_k$. Choosing $u = y_k$ and $v = B_k s_k$, we can solve for α, β:

$$\alpha = \frac{1}{y_k^T s_k}, \qquad \beta = -\frac{1}{s_k^T B_k s_k}.$$

Then substitute α and β back into $B_{k+1} = B_k + \alpha uu^T + \beta vv^T$ and get the update equation of B_{k+1} :

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

Alternative version of BFGS

Direct update the inverse of "Hessians" B_k: $p_k = -\tilde{B}_k \nabla f(x_k)$ where $\tilde{B}_k = B_k^{-1}$:

$$\tilde{B}_{k+1} = \left(I - \frac{s_t y_k^T}{y_k^T s_k}\right) \tilde{B}_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}.$$

Maximization Problem

Note: $\max f(x) = \min(-f(x))$

Newton method

$$x_{k+1} \leftarrow x_k - \left[\nabla^2 f(x_k)\right]^{-1} \cdot \nabla f(x_k) \quad \text{ note: } \left[\nabla^2 f(x_k)\right]^{-1} \text{is n.d.}$$

gradient descend method

$$x_{k+1} \leftarrow x_k - (-\nabla f(x_k))$$

Line Search

We now turn to the problem of how to find the step length: choose α assume p_k has been found (say $p_k^T \nabla f_k < 0$). An ideal choice of α is find a global minimizer of

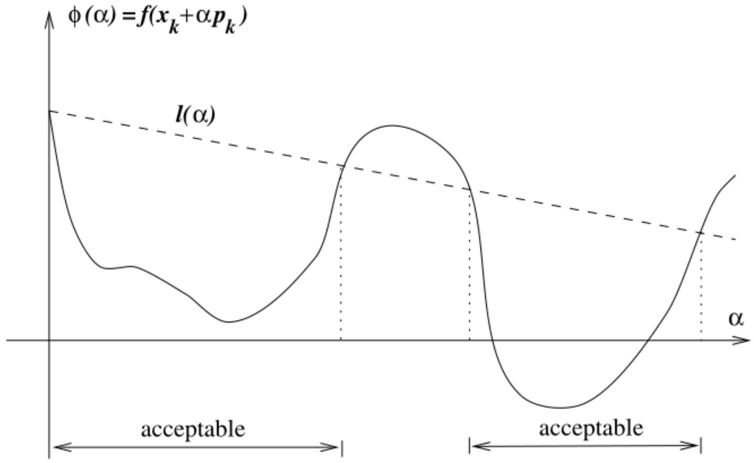
$$\phi(\alpha) \equiv f(x_k + \alpha p_k), \quad \alpha > 0$$

but too costly. Instead, we use some inexact line search method to identify a step length that achieves adequate reduction in f.

Sufficient reduction condition(Armijo condition) Let $l(\alpha)$ be the Taylor approximation of $\phi(\alpha)$ at 0, so $l(\alpha) = \phi(0) + \alpha\phi'(0) = f(x_k) + \alpha \nabla f(x_k)^T p_k$, the idea is to choose some α so that $\phi(\alpha) \leq l(\alpha)$. More precisely, consider

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k,$$

here, c_1 can be some small number, say 10^{-4}. The condition says we need to choose α such that there is sufficient reduction in $\phi(\alpha)$, i.e., $\phi(\alpha) \le l(c_1\alpha)$.



https://optimization.cbe.cornell.edu/index.php?title=File:Armijo.png

Procedure (Backtracking line search) choose $\bar{\alpha} > 0, \rho \in (0,1), c_1 > 0$.

Inside the k-th step of iteration, do

$$\begin{aligned}
&\text{repeat until } f(x_k + \alpha p_k) \le f(x_k) + c_1 \alpha p_k^T \nabla f(x_k) \\
&\qquad\qquad\quad \alpha \leftarrow \rho\alpha \\
&\text{end(repeat)}
\end{aligned}$$

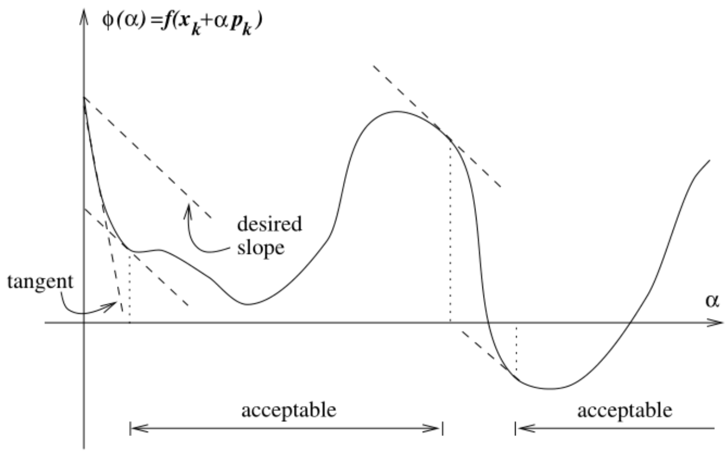
When terminated, $\alpha_k \leftarrow \alpha$.

Curvature condition

While the sufficient reduction condition ensures the step length decreases f “sufficiently”, it is possible that the step length is too small (or no sufficient reduction in the slope). To address this issue, often the curvature condition is imposed requiring α:

$$\phi'(\alpha) \equiv \nabla f(x_k + \alpha p_k)^T p_k \ge c_2 \phi'(0) \equiv c_2 \nabla f(x_k)^T p_k$$

where $c_2 \in (c_1, 1)$, typically on the order of 0.1. Note that the right hand side $f(x_k)^T p_k < 0$.



https://optimization.cbe.cornell.edu/index.php?title=File:Curvature_condition.png

So if $\phi'(\alpha) < c_2\phi'(0)$, ϕ is still decreasing at α, so we can improve the reduction in f by taking a larger α. If instead, $\phi'(\alpha) \geq c_2\phi'(0)$, then either we are close to a stationary point (minimum) where $\phi'(\alpha) = 0$ or $\phi'(\alpha) > 0$ which means we passed the stationary point.

This curvature condition, together with the Armijo condition are called the **Wolfe condition**.

Note: the Wolfe conditions can result in an α value that is not close to the minimizer of $\phi(\alpha)$. The **Strong Wolfe condition**

$$\left|\nabla f\left(x_k + \alpha p_k\right)^T p_k\right| \leq c_2 \left|\nabla f\left(x_k\right)^T p_k\right|$$

prevents the slope of $\phi(\alpha)$ from getting too large, hence excluding unlikely candidates that are far away from the stationary point.

WIKI: "Wolfe's conditions are more complicated than Armijo's condition, and a gradient descent algorithm based on Armijo's condition has a better theoretical guarantee than one based on Wolfe conditions."

Algorithm (BFGS with line search) For the BFGS algorithm, the curvature condition in the line search guarantees the $s_k^T y_k > 0$ (simple check).

Let $x_0, \epsilon > 0, \tilde{B}_0, k = 0$

$$
\begin{aligned}
&\text{while } ||\nabla f(x_k)|| > \epsilon : \\
&\quad \text{compute } p_k = -\tilde{B}_k \nabla f(x_k) \\
&\quad \text{line search with Wolfe condition } for\ \alpha_k \\
&\quad x_{k+1} \leftarrow x_k + \alpha_k p_k \\
&\quad s_k \leftarrow x_{k+1} - x_k \\
&\quad y_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k) \\
&\quad \text{compute } \tilde{B}_{k+1} \\
&\quad k \leftarrow k + 1 \\
&\text{end while}
\end{aligned}
$$

Line Search by interpolation

Goal :Find α satisfies the sufficient reduction condition without being too small.

Let

$$\phi(\alpha) = f(x_k + \alpha p_k), \quad (\alpha > 0),$$

The Armijo condition is

$$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0), \quad \phi'(0) = \nabla f(x_k)^T p_k,$$

Suppose our initial guess for next α is $\alpha_0 > 0$.

If Armijo condition if satisfied, then done.

If Armijo condition is not satisfied, then we know $\phi(\alpha)$ may be minimized further on $[0, \alpha_0]$

The interpolation idea is to approx ϕ by quadratic approx $\phi_q(\cdot)$, such that $\phi_q(\alpha) = a\alpha^2 + b\alpha + c$, satisfying

$$
\begin{aligned}
\phi_q(0) &= \phi(0) \\
\phi_q(\alpha_0) &= \phi(\alpha_0) \\
\phi_q'(\alpha_0) &= \phi'(\alpha_0)
\end{aligned}
$$

Solve for a, b, c in terms of $\phi(0), \phi(\alpha_0)$ and $\phi'(\alpha_0)$.

$$\phi_q(\alpha) = \frac{\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)}{\alpha_0^2} \alpha^2 + \phi'(0)\alpha + \phi(0),$$

The minimizer of $\phi_q(\cdot)$ over α is given by

$$\alpha_{min} = -\frac{b}{2a} = \frac{-\alpha_0^2 \phi'(0)}{2(\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0))}.$$

$\alpha_1 \leftarrow \alpha_{min}$.

If α_1 satisfy Armijo condition, then done.

If not, then we interpolate a cubic function at $\phi(0), \phi'(0), \phi(\alpha_0)$ and $\phi(\alpha_1)$ obtaining $\phi_c(\alpha) = a\alpha^3 + b\alpha^2 + \alpha\phi'(0) + \phi(0)$ where

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_0^2\alpha_1^2(\alpha_1 - \alpha_0)} \begin{bmatrix} \alpha_0^2 & -\alpha_1^2 \\ -\alpha_0^3 & \alpha_1^3 \end{bmatrix} \begin{bmatrix} \phi(\alpha_1) - \phi(0) - \phi'(0)\alpha_1 \\ \phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0 \end{bmatrix}$$

The minimizer α_2 of $\phi_c(\cdot)$ turns out lies in the interval $[0, \alpha_1]$ and is given by $\alpha_2 = \frac{-b+\sqrt{b^2-3a\phi'(0)}}{3a}$.

If α_2 satisfy Armijo condition, then done.

If not, continue the process using a cubic interpolate of $\phi(0), \phi'(0)$ and two most recent values of ϕ, until α is found to satisfy Amijo condition.

If any α_i is either too close to it's predecessor α_{i-1} or too much closer to 0, then we simply set $\alpha_i = \frac{\alpha_{i-1}}{2}$.

Modified Newton Method

By eigen-decomposition, we write for the square symmetric matrix $\nabla^2 f(x_k) = V_k D_k V_k^T$. By definition, $\nabla^2 f(x_k)V_k = V_k D_k$ where ($D_k = \delta_j$diagonal matrix consists of eigenvalues of the $\nabla^2 f(x_k)$).

If $\delta_j < \epsilon$, then set $\delta_j \leftarrow 2\epsilon$, call the new $D_k \to \widetilde{D}_k$; redefine Hessian to be $V_k\widetilde{D}_k V_K^T$. Note that $\nabla^2 f(x_k)^{-1} = V_k\widetilde{D}_k^{-1}V_k^T$.

No linear-LS-Problems

$f(x) = \frac{1}{2}\sum_{j=1}^m r_j^2(x), f : R^n \to R$, each r_j is some residual term, assume $m \geq n$.

$$r = \begin{bmatrix} r_1(x) \\ r_2(x) \\ \vdots \\ r_m(x) \end{bmatrix}$$

then $f(x) = \frac{1}{2}||r(x)||^2$.

Let $J(x)$ denote the Jacobian of $r(x)$:

$$J(x) := \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix}.$$

We can show that

$$\nabla f(x) = \sum_{j=1}^m r_j(x)\nabla r_j(x) = J(x)^T r(x)$$

$$\nabla^2 f(x) = \sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x)$$

On the right hand side, the first term $\sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)^T$ is in general more important than the second term, either because of the near-linearity of the model near the solution, or because of small residuals.

Recall Newton's Method: solve p_k from $\nabla^2 f(x_k) p_k = -\nabla f(x_k)$. Here we solve $J_k^T J_k p_k = -J_k^T r_k$ iteratively. This method is sometimes called **Gauss-Newton method**.

Solve non-linear equatins

$r : R^n \to R^n, r(x) = 0$

$r(x_k + p) \approx r(x_k) + J(x_k)p$ (Taylor expansion for multiple equation and $J(x) = \nabla r(x)$ – a $n \times n$ matrix is the Jacobian of r). We solve for $p_k = -J^{-1}(x_k) r(x_k)$ if $J(x_k)$ is invertible. Then $x_{k-1} \leftarrow x_k + p_k$.

Note: if the solution to $r(x) = 0$ exists, then solving the equation is equivalent to the optimization problem $\min_{x \in \mathbb{R}^n} \|r(x)\|^2$.

L2-regularzation

Example: Linear LS problem

$\min f(x) = ||Ax - b||^2 = x^T A^T Ax - 2b^T Ax + b^T b$

Example: non-linear LS problem (logistic regression):

n data points: $(x_i, y_i)_{i=1}^n, 0 \leq y_i \leq 1, f(x_i) \approx y_i$; logistic function (sigmoid): $\delta(t) = \frac{1}{1+e^{-t}}$; $y_i \approx \delta(a + bx_i)$, or $y_i \approx \delta(\alpha + x_i^T \beta)$ if x is a vector.

LS problem: $min_{\alpha,\beta} \sum_{i=1}^n (\delta(\alpha + x_i^T \beta) - y_i)^2$

$$\widetilde{\beta} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \widetilde{x}_i = \begin{bmatrix} 1 \\ x_i \end{bmatrix}$$

$$\begin{aligned} \nabla RSS(\widetilde{\beta}) &= \frac{d}{d_{\widetilde{\beta}}} \sum_{i=1}^{n} (\delta(\widetilde{x}_i^T \widetilde{\beta}) - y_i)^2 \\ &= 2 \sum_{i=1}^{n} (\delta(\widetilde{x}_i^T \widetilde{\beta}) - y_i) \delta(\widetilde{x}_i^T \widetilde{\beta})(1 - \delta(\widetilde{x}_i^T \widetilde{\beta})) \widetilde{x}_i \end{aligned}$$

Regularized logistic regression: $\sum_{i=1}^n (\delta(\alpha + x_i^T \beta) - y_i)^2 + \lambda ||\beta||^2$

Note: From a pure optimization's perspective, regularize is a simple convex function that is often added to a non-convex objective function slightly convexifying it and helping numerical optimization technique avoid some poor solution in some flat area.