CS 450: Numerical Anlaysis

Lecture 16

Chapter 6 Numerical Optimization

Unconstrained Optimization Algorithms and Nonlinear Least Squares

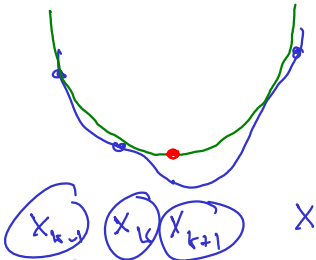
Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

March 9, 2018

Successive Parabolic Interpolation

- Interpolate f with a quadratic function at each step and find its minima:



use last 3 iterates to define interpolant

$$\begin{bmatrix} x_{k-2}, f(x_{k-2}) \\ x_{k-1}, f(x_{k-1}) \\ x_k, f(x_k) \end{bmatrix}$$

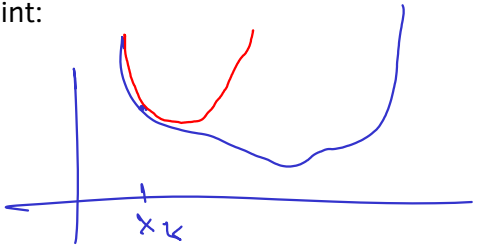
then find its minima

- The convergence rate of the resulting method is roughly 1.324

superlinear
better than GSS (linear, $c \simeq .6$)
worse than Newton (quadratic)

Newton's Method for Optimization

- Instead of interpolating points, match derivatives at current approximate point:



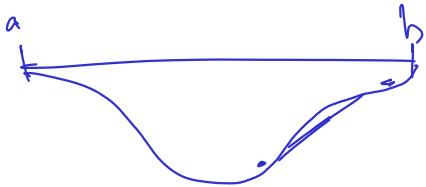
expand Taylor series of f about x_k and keep 3 terms then minimize

- The new approximate guess will be given by $x_{k+1} - x_k = -f'(x_k)/f''(x_k)$:

$$x_{k+1} = x_k - \underbrace{f'(x_k)/f''(x_k)}$$

Safeguarded 1D Optimization

- Safeguarding can be done by bracketing via golden section search:



unimodal → GSS will succeed
strictly decreasing from a until minima, then strictly increasing until b

- Backtracking and step-size control:

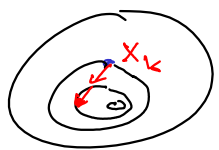
try Newton step
if outside of bracket
perform step of GSS

Multidimensional Optimization

- Direct search methods by simplex (*Nelder-Mead*):

- Steepest descent: find the minimizer in the direction of the negative gradient:

iteration:



$-\nabla f(x_k)$ $\quad x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

find α_k to minimize

$$\alpha_k = \underset{\alpha_k}{\operatorname{argmin}} f(x_k - \alpha_k \nabla f(x_k))$$

Convergence of Steepest Descent

- Steepest descent converges linearly with a constant that can be arbitrarily close to 1:

can always make progress for sufficiently small α_k

but could have to set α_k very small

---

descent direction → any direction in which the objective func. ↓

Newton's Method for Multidimensional Optimization

- Newton's method in n dimensions is given by finding minima of n-dimensional quadratic approximation:

gradient Hessian

$$f(\boldsymbol{x}_k) \approx \hat{f}(\boldsymbol{x}_k + \boldsymbol{s}) = f(\boldsymbol{x}_k) + \boldsymbol{s}^T \nabla f(\boldsymbol{x}_k) + \frac{1}{2}\boldsymbol{s}^T \boldsymbol{H}_f(\boldsymbol{x}_k)\boldsymbol{s}$$

Consider f at x_k

$$\vec{0} = \nabla \hat{f}(x_k + s) = \nabla f(x_k) + H_f(x_k)\, s = 0$$

$$-\nabla f(x_k) = H_f(x_k)\, s$$

Solve

$$x_{k+1} = x_k - H_f(x_k)^{-1} \nabla f(x_k)$$

Convergence of Newton's Method

- Newton's method is quadratically convergent in n-dimensions as well:

$$
\begin{aligned}
e_{k+1} &= x_{k+1} - x^* \\
&= x_k - H_f(x_k)^{-1} \nabla f(x_k) - x^* \\
&= -H_f(x_k)^{-1} \left[\nabla f(x_k) + H_f(x_k)(x^* - x_k) \right] \qquad (x^* - x_k = -e_k) \\
\|e_{k+1}\| &\le \|H_f(x_k)^{-1}\| \, \| \nabla f(x_k) - \nabla f(x^*) + H_f(x_k)(x^* - x_k) \| \qquad (\nabla f(x^*) = 0) \\
&\le C \|e_k\|^2 \qquad \left(\| \nabla f(x_k) - \nabla f(x^*) + H_f(x_k)(x^* - x_k) \| = O(\|x^* - x_k\|^2) \right)
\end{aligned}
$$

Quasi-Newton Methods

- *Quasi-Newton* methods compute approximations to the Hessian at each step:

replace $H_f(x_k)$ with B_k (init. $B_0 = I$)

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k)$$

want descent direction | Newton may not give one | step-size control (line search)

- The *BFGS* method is a secant update method, similar to Broyden's method:

good Quasi-Newton methods are more robust than Newton in finding descent direction

Sherman Morrison $O(n^2)$

BFGS is similar Broyden's method (secant approximation), rank-1 update

Nonlinear Least Squares

- Lets consider a *nonlinear least squares* problem of fitting a nonlinear function $f_{\boldsymbol{x}}(t)$ so that $f_{\boldsymbol{x}}(t_i) \approx y_i$:

$$f_{[x_1, x_2]}(t) = x_1 \sin(x_2 t)$$

- We can cast nonlinear least squares as an optimization problem and solve it by Newton's method:

$$r_i(x) = -f_x(t_i) + y_i = y_i - f_x(t_i)$$

$$\min_x \left[\phi(x) = \frac{1}{2} \|r(x)\|_2^2 \right] \qquad \phi(x) = \frac{1}{2} r^T(x) r(x)$$

$$\nabla \phi(x) = J_r(x) r(x)$$

$$H_\phi(x) = J_r^T(x) J_r(x) + \sum_{i=1}^{m} r_i(x) H_{r_i}(x)$$

Gauss-Newton Method

- The Gauss-Newton method is a simplification of Newton's method for the nonlinear least squares problem:

$$H_\phi(x) \approx \hat{H}_\phi(x) = J_r^T(x) J_r(x)$$

- The method is then given by

Quasi-Newton with approx.

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \hat{\boldsymbol{H}}_\phi(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k) \\ &= \boldsymbol{x}_k - (\boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x}_k)\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k))^{-1} \boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k) \end{aligned}$$

nonlinear LS

↓

iterative comp. linear LS

$x_k - s$

$J_r(x_k)\, s \cong r(x_k)$

Levenberg-Marquardt Method

- The *Levenberg-Marquardt* modifies the Gauss-Newton method to use Tykhonov regularization:

$$x_{k+1} = x_k - \left(J_r(x_k)^T J_r(x_k) + \mu_k I \right) J_r(x_k)^T r(x_k)$$

- The scalar μ controls the step size through the least squares problem:

$$\begin{bmatrix} J_r(x_k) \\ \sqrt{\mu_k}\, I \end{bmatrix} s \cong \begin{bmatrix} r(x_k) \\ 0 \end{bmatrix}$$

$\mu \to$ large
then s is small
(step is small)