HW10: DL November 3

Nelder-Mead Method

Idea:

Demo: Nelder-Mead Method [cleared]

Newton's method (n D)

What does Newton's method look like in n dimensions?

$$f(\vec{x}+\vec{s}) = f(\vec{x}) + \nabla f(x)^T s + \frac{1}{2} s^T H_f(\vec{x})\,\vec{s} + O(\|s\|^3)$$

$$\approx \tilde{f}(\vec{s})$$

$$\nabla \tilde{f}(\vec{s}) = 0 \iff H_f(\vec{x})\,\vec{s} = -\nabla f(x)$$

$$\vec{x}_0 = (\text{starting guess})$$

$$\vec{x}_{k+1} = x_k - H_f(\vec{x}_k)^{-1} \nabla f(x_k)$$

Newton's method (n D): Observations

Drawbacks?

- locally conv.
- need two derivatives
- If Hessian is ill-conditioned, ⇒ problems

Demo: Newton's method in n dimensions [cleared]

Expect quadratic conv.

Quasi-Newton Methods

Secant/Broyden-type ideas carry over to optimization. How?

$$\vec{x}_{k+1} = \vec{x}_k - \alpha_k \underline{B_k^{-1}} \nabla f(x_k)$$

Broyden

BFGS: Secant-type method, similar to Broyden:

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{B}_k}{\boldsymbol{s}_k^T \boldsymbol{B}_k \boldsymbol{s}_k}$$

where

- $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$
- $\boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$

$B_{k+1} s_k = y_k$ "secant condition"

$B_{k+1} = B_k + \alpha \vec{u}\vec{u}^T + \beta \vec{v}\vec{v}^T$

update B_k^{-1} via Sherman-Morrison

In-Class Activity: Optimization Methods

In-class activity: Optimization Methods

Nonlinear Least Squares: Setup

What if the f to be minimized is actually a 2-norm?

$$f(\boldsymbol{x}) = \|\underline{\boldsymbol{r}(\boldsymbol{x})}\|_2, \qquad \boldsymbol{r}(\boldsymbol{x}) = \underline{\boldsymbol{y} - \boldsymbol{a}(\boldsymbol{x})}$$

Define 'helper function':

$$\varphi(\vec{x}) = \frac{1}{2}\vec{r}(x)^T\vec{r}(x)$$

and minimize that instead

$$\frac{\partial}{\partial x_i}\varphi = \frac{1}{2}\sum_{j=1}^{n}\frac{\partial}{\partial x_i}[r_j(x)]^2 = \sum_{j=1}^{n}\left(\frac{\partial}{\partial x_i}r_j(\vec{x})\right)r_j(\vec{x})$$

$$\nabla\varphi = J_{\vec{r}}(x)^T\vec{r}(x)$$

Gauss-Newton

For brevity: $J := J_r(\boldsymbol{x})$.

$$H_\varphi = J^T J + \sum_{i=1}^{n} r_i(x)\, H_{r_i}(x)$$

Newton step: $\vec{x}_{k+1} = \vec{x}_k - H_\varphi^{-1} \nabla\varphi$

$$\tilde{H}_\varphi = J^T J$$

Gauss-Newton step: $\vec{x}_{k+1} = \vec{x}_k \underbrace{- \tilde{H}_\varphi^{-1} \nabla\varphi}_{\vec{s}}$

$$J^T J \vec{s} = J^T \vec{r}$$

normal equations for $J\vec{s} \cong \vec{r}$

Gauss-Newton: Observations?

Demo: Gauss-Newton [cleared]

Observations?

- still locally convergent
- possibly even more sensitive to choice of starting guess
- $J^T J$ ill-conditioned $\Rightarrow$ problems

Levenberg-Marquardt

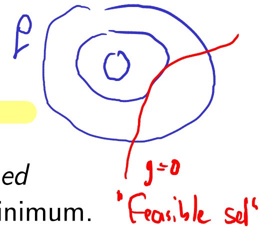
If Gauss-Newton on its own is poorly, conditioned, can try Levenberg-Marquardt:

$$\left(J_{\vec r}(\vec x_k)^T J_{\vec r}(\vec x_k) + \mu_k I\right)\vec s_k = -J_{\vec r}(x_k)^T \vec r_k(x_k)$$

equivalent to the LSQ

$$\begin{bmatrix} J_{\vec r}(r_k) \\ \sqrt{\mu_k}\, I \end{bmatrix} s_k \cong \begin{bmatrix} -\vec r(x_k) \\ 0 \end{bmatrix}$$

Constrained Optimization: Problem Setup



Want $\boldsymbol{x}^*$ so that

$$f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = 0$$

No inequality constraints just yet. This is *equality-constrained optimization*. Develop a (local) necessary condition for a minimum.

Assume $g(\vec{x}) = 0$.

Necessary cond: "no feasible descent direction"

Unconstrained: $\nabla f(x) = 0$

$\vec{s}$ is a feasible direction iff $\vec{x} + \alpha\vec{s}$ is feasible for $\alpha \in [0, r]$

Constrained Optimization: Necessary Condition

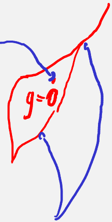
Need: $\nabla f(x) \cdot \vec{s} \geq 0$ for any feasible direction

"uphill that way"

- 'Not at the boundary'

  $\rightsquigarrow$ locally unconstrained

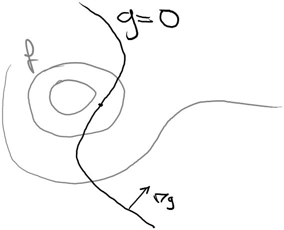
  $\nabla f(\vec{x}) = 0$



- 'At the boundary'

  $-\nabla f \in \operatorname{rowspan}(J_g)$

Lagrange Multipliers



Seen: Need $-\nabla f(\boldsymbol{x}) = J_{\boldsymbol{g}}^T \boldsymbol{\lambda}$ at the (constrained) optimum.

Idea: Turn constrained optimization problem for $\boldsymbol{x}$ into an *unconstrained* optimization problem for $(\boldsymbol{x}, \boldsymbol{\lambda})$. How?