November 11, 2024

Announcements

- HW8
- Exam 4 (content cutoff today)

Goals

- Nonlinear LSQ
- Interpolation

Review

Newton's method (n D)

$f : \mathbb{R}^n \to \mathbb{R}$

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

$$\underset{\substack{\uparrow \\ \vec{x}_k}}{f(\vec{x}+\vec{h})} = \underbrace{f(\vec{x}) + \nabla f(\vec{x}) \cdot \vec{h} + \frac{1}{2}\vec{h}^T H(\vec{x})\vec{h}}_{\hat{f}(\vec{h})} + O(\|h\|^3)$$

$$\nabla \hat{f}(h) = \nabla f(\vec{x}) + H_f(\vec{x})\,\vec{h} = \vec{0}$$

$$H_f(\vec{x})\,\vec{h} = -\nabla f \iff \vec{h} = -H_f^{-1}(\vec{x})\nabla f$$

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

Newton's method (n D): Observations

Drawbacks?

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

Newton's method (n D): Observations

Drawbacks?

- Need second (!) derivatives
  (addressed by Conjugate Gradients, later in the class)
- local convergence
- Works poorly when H_f is nearly indefinite

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

Quasi-Newton Methods

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

Come up with a way to update to update the approximate Hessian.

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

$\alpha_k \to$ line search / damping parameter

$s_k = x_{k+1} - x_k$

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

$B_k s_k = y_k \leftarrow n$ conditions

BFGS: Secant-type method, similar to Broyden:

$\uparrow$ Broyden

$$B_{k+1} = B_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{B_k \mathbf{s}_k \mathbf{s}_k^T B_k}{\mathbf{s}_k^T B_k \mathbf{s}_k}$$

Nonlinear Least Squares: Setup

$y = a(\vec{x})$

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

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

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

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

$\vec{r}: \mathbb{R}^n \to \mathbb{R}^n$

$$\nabla\varphi = J_r^T\vec{r}$$

$$J_r = \begin{pmatrix} \partial_1 r_1 & & \partial_n r_1 \\ \vdots & & \\ \partial_1 r_n & & \end{pmatrix}$$

Gauss-Newton

$$x_{k+1} = x_k - H_\varphi^{-1} \nabla \varphi$$

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

$$H_\varphi(\vec{x}) = J_r^T J_r + \cancel{\sum_i r_i H_{r_i}(\vec{x})}$$

"small because residual is (hopefully) small"

$$\text{"}H_\varphi\text{"}\, \vec{h} = -\nabla \varphi$$

$$J_r^T J_r \vec{h} = -J_r^T \vec{r} \iff J \vec{h} \cong -\vec{r}$$

Gauss-Newton: Observations?

Demo: Gauss-Newton [cleared]

Observations?

Gauss-Newton: Observations?

Demo: Gauss-Newton [cleared]

Observations?

- Newton on its own is still only locally convergent
- Gauss-Newton is clearly similar
- It's worse because the step is only approximate
  $\rightarrow$ Much depends on the starting guess.

Levenberg-Marquardt

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

$x_{k+1} = x_k + \tilde{h}_k$

$$(J^T J + \mu_k I)\tilde{h} = -J^T r$$

$$\Leftrightarrow \begin{bmatrix} J \\ \sqrt{\mu_k} I \end{bmatrix} \tilde{h} \cong \begin{bmatrix} -r \\ 0 \end{bmatrix} \qquad \sqrt{\mu_k}\|h\|_2 \to \min$$

Constrained Optimization: Problem Setup

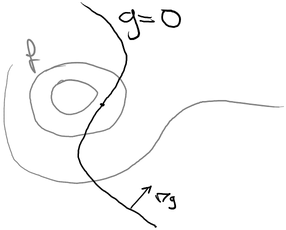
Want $\boldsymbol{x}^*$ so that

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

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

Constrained Optimization: Necessary Condition

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?

Lagrange Multipliers: Development

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) := f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}).$$

Demo: Sequential Quadratic Programming [cleared]

Inequality-Constrained Optimization

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

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

Develop a necessary condition for a minimum.

Lagrangian, Active/Inactive

Put together the overall Lagrangian.

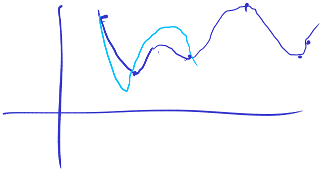
What are active and inactive constraints?

Karush-Kuhn-Tucker (KKT) Conditions

Develop a set of necessary conditions for a minimum.

Outline

Interpolation: Setup



Given: $(x_i)_{i=1}^N$, $(y_i)_{i=1}^N$
Wanted: Function f so that $f(x_i) = y_i$

How is this not the same as function fitting? (from least squares)

$$\max_{x\in[a,b]} |f_{\text{interp}}(x) - f(x)|$$

smooth, "true" function

Interpolation: Setup

Given: $(x_i)_{i=1}^N$, $(y_i)_{i=1}^N$
Wanted: Function f so that $f(x_i) = y_i$

How is this not the same as function fitting? (from least squares)

It's very similar–but the key difference is that we are asking for *exact equality*, not just minimization of a residual norm.
$\rightarrow$ Better error control, error not dominated by residual

Idea: There is an *underlying function* that we are approximating from the known point values.

Error here: Distance from that underlying function

Interpolation: Setup (II)

Given: $(x_i)_{i=1}^N$, $(y_i)_{i=1}^N$
Wanted: Function f so that $f(x_i) = y_i$

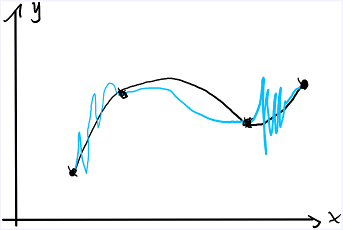
Does this problem have a unique answer?

Interpolation: Setup (II)

Given: $(x_i)_{i=1}^N$, $(y_i)_{i=1}^N$
Wanted: Function f so that $f(x_i) = y_i$

Does this problem have a unique answer?

No–there are infinitely many functions that satisfy the problem as stated:



Interpolation: Importance

Why is interpolation important?

Interpolation: Importance

Why is interpolation important?

It brings all of calculus within range of numerical operations.

- Why?

  Because calculus works on functions.
- How?
  1. Interpolate (go from discrete to continuous)
  2. Apply calculus
  3. Re-discretize (evaluate at points)

Making the Interpolation Problem Unique

$\varphi_1, \ldots, \varphi_n$ given basis of functions, often polynomials

$$p_{n-1}(x) = \sum_{j=1}^{n} \alpha_j \varphi_j(x)$$

α_j: unknown

$$y_1 = p_{n-1}(x_1) = \sum_{j=1}^{n} \alpha_j \varphi_j(x_1)$$

$$V\vec{\alpha} = \vec{y}$$

$$\Rightarrow \underbrace{\begin{pmatrix} \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & & \\ \varphi_1(x_n) & & \varphi_n(x_n) \end{pmatrix}}_{\text{generalized Vandermonde}} \vec{\alpha} = \vec{y} = \begin{pmatrix} \\ \\ y_n \end{pmatrix}$$

If $\varphi_j = x^j$, "actual" Vandermonde matrix

Existence/Sensitivity

Solution to the interpolation problem: Existence? Uniqueness?

Sensitivity?