Avoiding Derivatives

Part II Roadmap

- Part I – Linear Algebra (units 1-12) $Ac = b$

linearize

line search

- Part II – Optimization (units 13-20)
  - (units 13-16) Optimization -> Nonlinear Equations -> 1D roots/minima ← Theory
  - (units 17-18) Computing/Avoiding Derivatives
  - (unit 19) Hack 1.0: "I give up" $H = I$ and J is mostly 0 (descent methods)
  - (unit 20) Hack 2.0: "It's an ODE!?" (adaptive learning rate and momentum)

Methods

1D Root Finding (see Unit 15)

- Newton's method requires g', as do mixed methods using Newton
- Secant method replaces g' with a secant line though two prior iterates
- Finite differencing (unit 17) may be used to approximate this derivative as well, although one needs to determine the size of the perturbation h
- Automatic differentiation (unit 17) may be used to find the value of g' at a particular point, if/when "backprop" code exists, even when g and g' are not known in closed form
- Convergence is only guaranteed under certain conditions, emphasizing the importance of safe set methods (such as mixed methods with bisection)
- Safe set methods (such as mixed methods with bisection) also help to guard against errors in derivative approximations

1D Optimization (see Unit 16)

- Root finding approaches search for critical points as the roots of g'
  - All root finding methods use the function itself (g' here)
  - Newton (and mixed methods using Newton) require the derivative of the function (g'' here)
- Can use <u>secant</u> lines for g' and <u>interpolating parabolas</u> for g'', using either prior iterates (unit 16) or finite differences (unit 17)
- <u>Automatic differentiation</u> (unit 17) may be leveraged as well
  - Although, not (typically) for approaches that require g''
- Safe set methods (such as mixed methods with bisection or golden section search) help to guard against errors in the approximation of various derivatives

Nonlinear Systems (see Unit 14)

- $J_F(c^q)\Delta c^q = (\beta - 1)F(c^q)$ is solved to find the search direction Δc^q
  - Then, line search utilizes various 1D approaches (unit 15/16)
- The Jacobian matrix of first derivatives $J_F(c^q)$ needs to be evaluated (given c^q)
- Each entry $\frac{\partial F_i}{\partial c_k}(c^q)$ can be approximated via finite differences (unit 17) or automatic differentiation (unit 17)
- Making various approximations to the Jacobian $J_F(c^q)$ perturbs the search direction, so robust/safe set approaches to the 1D line search are important for making "progress" towards solutions

Quasi-Newton Methods

- $J_F(c^q)\Delta c^q = (\beta - 1)F(c^q)$ is solved to find the search direction Δc^q
- The Jacobian matrix of first derivatives $J_F(c^q)$ needs to be evaluated (given c^q)
- Quasi-Newton approaches make various aggressive approximations to the Jacobian $J_F(c^q)$
- Quasi-Newton can wildly perturb the search direction
  - So, robust/safe set approaches to the 1D line search become quite important for making "progress" towards solutions

Broyden's Method

- An initial guess for the Jacobian is repeatedly corrected with rank one updates, similar in spirit to a secant approach
- Let $J^0 = I$
- Solve $J^q \Delta c^q = -F(c^q)$ to find search direction Δc^q
  - Use 1D line search to find c^{q+1} and thus $F(c^{q+1})$; then, update $\Delta c^q = c^{q+1} - c^q$ overwrite Δc^q
- Update $J^{q+1} = J^q + \frac{1}{(\Delta c^q)^T \Delta c^q}\left(F(c^{q+1}) - F(c^q) - J^q \Delta c^q\right)(\Delta c^q)^T$

- Note: $J^{q+1}(c^{q+1} - c^q) = F(c^{q+1}) - F(c^q)$
- That is, J^{q+1} satisfies a secant type equation $J\Delta c = \Delta F$

Optimization (see Unit 13)

- Scalar cost function $\hat{f}(c)$ has critical points where $J_{\hat{f}}^{T}(c) = 0$ (unit 13)
- $H_{\hat{f}}^{T}(c^q)\Delta c^q = (\beta - 1)J_{\hat{f}}^{T}(c^q)$ is solved to find a search direction Δc^q (unit 14)
- Then, line search utilizes various 1D approaches (unit 15/16)
- The Hessian matrix of second derivatives $H_{\hat{f}}^{T}(c^q)$ and the Jacobian vector of first derivatives $J_{\hat{f}}^{T}(c^q)$ both need to be evaluated (given c^q)
- The various entries can be evaluated via <u>finite differences</u> (unit 17) or <u>automatic differentiation</u> (unit 17)
- These approaches can struggle on the Hessian matrix of second partial derivatives

Quasi-Newton Methods (for optimization)

- $H_{\hat{f}}^T(c^q)\Delta c^q = (\beta - 1)J_{\hat{f}}^T(c^q)$ is solved to find a search direction Δc^q
- The Hessian matrix of second derivatives $H_{\hat{f}}^T(c^q)$ and the Jacobian vector of first derivatives $J_{\hat{f}}^T(c^q)$ both need to be evaluated (given c^q)
- Second derivatives pose even more issues than first derivatives
- This makes Quasi-Newton approaches quite popular for optimization
- When c is large, the $O(n^2)$ Hessian $H_{\hat{f}}^T$ is unwieldy/intractable, so some approaches instead approximate the action of $H_{\hat{f}}^{-T}$ on a vector
  - i.e. the action of $H_{\hat{f}}^{-T}$ on the right hand side

Broyden's Method (for Optimization)

- Same formulation as for nonlinear systems (3 slides prior)

- Solve for the search direction, and use 1D line search to find c^{q+1} and $J_{\hat{f}}^T(c^{q+1})$
- Overwrite $\Delta c^q = c^{q+1} - c^q$ and compute $\Delta J_{\hat{f}}^T = J_{\hat{f}}^T(c^{q+1}) - J_{\hat{f}}^T(c^q)$

- Update $(H_{\hat{f}}^T)^{q+1} = (H_{\hat{f}}^T)^q + \frac{1}{(\Delta c^q)^T \Delta c^q}\left(\Delta J_{\hat{f}}^T - (H_{\hat{f}}^T)^q \Delta c^q\right)(\Delta c^q)^T$
- So that $(H_{\hat{f}}^T)^{q+1}\Delta c^q = \Delta J_{\hat{f}}^T$ is satisfied (a secant type equation)

Broyden's Method (for Optimization)

- For the <u>inverse</u>, using $\Delta c^q = c^{q+1} - c^q$ and $\Delta J_{\hat{f}}^T = J_{\hat{f}}^T(c^{q+1}) - J_{\hat{f}}^T(c^q)$
- Update $(H_{\hat{f}}^{-T})^{q+1} = (H_{\hat{f}}^{-T})^q + \frac{\left(\Delta c^q - \left(H_{\hat{f}}^{-T}\right)^q \Delta J_{\hat{f}}^T\right)(\Delta c^q)^T \left(H_{\hat{f}}^{-T}\right)^q}{(\Delta c^q)^T (H_{\hat{f}}^{-T})^q \Delta J_{\hat{f}}^T}$
- So that $(H_{\hat{f}}^{-T})^{q+1} \Delta J_{\hat{f}}^T = \Delta c^q$

- Solving $H_{\hat{f}}^T(c^{q+1}) \Delta c^{q+1} = -J_{\hat{f}}^T(c^{q+1})$ is replaced with defining the search direction by $\Delta c^{q+1} = -(H_{\hat{f}}^{-T})^{q+1} J_{\hat{f}}^T(c^{q+1})$

SR1 (Symmetric Rank 1)

- For the <u>inverse</u>, using $\Delta c^q = c^{q+1} - c^q$ and $\Delta J_{\hat{f}}^T = J_{\hat{f}}^T(c^{q+1}) - J_{\hat{f}}^T(c^q)$
- Update $(H_{\hat{f}}^{-T})^{q+1} = (H_{\hat{f}}^{-T})^q + \dfrac{\left(\Delta c^q - \left(H_{\hat{f}}^{-T}\right)^q \Delta J_{\hat{f}}^T\right)\left(\Delta c^q - \left(H_{\hat{f}}^{-T}\right)^q \Delta J_{\hat{f}}^T\right)^T}{\left(\Delta c^q - \left(H_{\hat{f}}^{-T}\right)^q \Delta J_{\hat{f}}^T\right)^{\mathrm{T}} \Delta J_{\hat{f}}^T}$
- So that $(H_{\hat{f}}^{-T})^{q+1} \Delta J_{\hat{f}}^T = \Delta c^q$

- Solving $H_{\hat{f}}^T(c^{q+1}) \Delta c^{q+1} = -J_{\hat{f}}^T(c^{q+1})$ is replaced with defining the search direction by $\Delta c^{q+1} = -(H_{\hat{f}}^{-T})^{q+1} J_{\hat{f}}^T(c^{q+1})$

DFP (Davidon-Fletcher-Powell)

- For the <u>inverse</u>, using $\Delta c^q = c^{q+1} - c^q$ and $\Delta J_{\hat{f}}^T = J_{\hat{f}}^T(c^{q+1}) - J_{\hat{f}}^T(c^q)$
- Update $(H_{\hat{f}}^{-T})^{q+1} = (H_{\hat{f}}^{-T})^q - \frac{\left(H_{\hat{f}}^{-T}\right)^q \Delta J_{\hat{f}}^T \Delta J_{\hat{f}} \left(H_{\hat{f}}^{-T}\right)^q}{\Delta J_{\hat{f}} \left(H_{\hat{f}}^{-T}\right)^q \Delta J_{\hat{f}}^T} + \frac{\Delta c^q (\Delta c^q)^T}{(\Delta c^q)^T \Delta J_{\hat{f}}^T}$
- So that $(H_{\hat{f}}^{-T})^{q+1} \Delta J_{\hat{f}}^T = \Delta c^q$
- Solving $H_{\hat{f}}^T(c^{q+1}) \Delta c^{q+1} = -J_{\hat{f}}^T(c^{q+1})$ is replaced with defining the search direction by $\Delta c^{q+1} = -(H_{\hat{f}}^{-T})^{q+1} J_{\hat{f}}^T(c^{q+1})$

BFGS (Broyden-Fletcher-Goldfarb-Shanno)

- For the <u>inverse</u>, using $\Delta c^q = c^{q+1} - c^q$ and $\Delta J_{\hat{f}}^T = J_{\hat{f}}^T(c^{q+1}) - J_{\hat{f}}^T(c^q)$
- Update $(H_{\hat{f}}^{-T})^{q+1} = \left(I - \frac{\Delta c^q \Delta J_{\hat{f}}}{(\Delta c^q)^T \Delta J_{\hat{f}}^T}\right)\left(H_{\hat{f}}^{-T}\right)^q \left(I - \frac{\Delta J_{\hat{f}}^T (\Delta c^q)^T}{(\Delta c^q)^T \Delta J_{\hat{f}}^T}\right) + \frac{\Delta c^q (\Delta c^q)^T}{(\Delta c^q)^T \Delta J_{\hat{f}}^T}$
- So that $(H_{\hat{f}}^{-T})^{q+1} \Delta J_{\hat{f}}^T = \Delta c^q$
- Solving $H_{\hat{f}}^T(c^{q+1})\Delta c^{q+1} = -J_{\hat{f}}^T(c^{q+1})$ is replaced with defining the search direction by $\Delta c^{q+1} = -(H_{\hat{f}}^{-T})^{q+1} J_{\hat{f}}^T(c^{q+1})$

L-BFGS (Limited Memory BFGS)

- Storing an nxn approximation to the inverse Hessian can become unwieldy for large problems
- More efficient to instead store the vectors that describe the outer products; however, the number of vectors grows with q
- L-BFGS estimates the inverse Hessian using only a few of the prior vectors
  - often less than 10 vectors (vectors, vector spaces, not matrices)
- This makes it quite popular for machine learning

On optimization methods for deep learning, Andrew Ng et al., ICML 2011

- "we show that more sophisticated off-the-shelf optimization methods such as Limited memory BFGS (L-BFGS) and Conjugate gradient (CG) with line search can significantly simplify and speed up the process of pretraining deep algorithms"

Gradient/Steepest Descent

- Approximate $H_{\hat{f}}^{T}$ very crudely with the identity matrix
  - which is the first step of all the aforementioned methods
- That is, $H_{\hat{f}}^{T}(c^q)\Delta c^q = -J_{\hat{f}}^{T}(c^q)$ becomes $I\Delta c^q = -J_{\hat{f}}^{T}(c^q)$
- So, the search direction is $\Delta c^q = -J_{\hat{f}}^{T}(c^q) = -\nabla\hat{f}(c^q)$
  - This is the steepest descent direction

- See unit 19

Coordinate Descent

- Coordinate Descent ignores $H_{\hat{f}}^{T}(c^q)\Delta c^q = -J_{\hat{f}}^{T}(c^q)$ completely
- Instead, Δc^q is set to the various coordinate directions $\hat{e}_k$

Nonlinear Least Squares

- Recall from Unit 13:
  - Determine parameters c that make $f(x, y, c) = 0$ best fit the training data, i.e. that make $\|f(x_i, y_i, c)\|_2^2 = f(x_i, y_i, c)^T f(x_i, y_i, c)$ close to zero for all i
  - Combining all (x_i, y_i), minimize $\hat{f}(c) = \frac{1}{2}\sum_i f(x_i, y_i, c)^T f(x_i, y_i, c)$

- Let m be the number of data points and $\widehat{m}$ be the output size of $f(x, y, c)$
- Define $\tilde{f}(c)$ by stacking the $\widehat{m}$ outputs of $f(x, y, c)$ consecutively m times, so that the vector valued output of $\tilde{f}(c)$ is length $m * \widehat{m}$
- Then, $\hat{f}(c) = \frac{1}{2}\sum_i f(x_i, y_i, c)^T f(x_i, y_i, c) = \frac{1}{2}\tilde{f}^T(c)\tilde{f}(c)$

Nonlinear Least Squares

- Minimize $\hat{f}(c) = \frac{1}{2}\tilde{f}^T(c)\tilde{f}(c)$

- Jacobian matrix of $\tilde{f}$ is $J_{\tilde{f}}(c) = \begin{pmatrix} \frac{\partial \tilde{f}}{\partial c_1}(c) & \frac{\partial \tilde{f}}{\partial c_2}(c) & \cdots & \frac{\partial \tilde{f}}{\partial c_n}(c) \end{pmatrix}$

- Critical points of $\hat{f}(c)$ have $J_{\hat{f}}^T(c) = \begin{pmatrix} \tilde{f}^T(c)\frac{\partial \tilde{f}}{\partial c_1}(c) \\ \tilde{f}^T(c)\frac{\partial \tilde{f}}{\partial c_2}(c) \\ \vdots \\ \tilde{f}^T(c)\frac{\partial \tilde{f}}{\partial c_n}(c) \end{pmatrix} = J_{\tilde{f}}^T(c)\tilde{f}(c) = 0$

Gauss Newton

- $J_{\tilde{f}}^T(c)\tilde{f}(c) = 0$ becomes $J_{\tilde{f}}^T(c)\left(\tilde{f}(c^q) + J_{\tilde{f}}(c^q)\Delta c^q + \cdots\right) = 0$
  - Using the Taylor series: $\tilde{f}(c) = \tilde{f}(c^q) + J_{\tilde{f}}(c^q)\Delta c^q + \cdots$
- Eliminating high order terms: $J_{\tilde{f}}^T(c)\left(\tilde{f}(c^q) + J_{\tilde{f}}(c^q)\Delta c^q\right) \approx 0$
- Evaluating $J_{\tilde{f}}^T$ at c^q gives $J_{\tilde{f}}^T(c^q)J_{\tilde{f}}(c^q)\Delta c^q \approx -J_{\tilde{f}}^T(c^q)\tilde{f}(c^q)$

- Compare to $H_{\hat{f}}^T(c^q)\Delta c^q = -J_{\hat{f}}^T(c^q)$ and note that $J_{\hat{f}}^T(c) = J_{\tilde{f}}^T(c)\tilde{f}(c)$
- Thus, Gauss Newton uses the estimate: $H_{\hat{f}}^T(c^q) \approx J_{\tilde{f}}^T(c^q)J_{\tilde{f}}(c^q)$
  - Removes the second derivatives!

Gauss Newton (QR approach)

- Gauss Newton equations $J_{\tilde{f}}^{T}(c^q)J_{\tilde{f}}(c^q)\Delta c^q = -J_{\tilde{f}}^{T}(c^q)\tilde{f}(c^q)$ are the normal equations for $J_{\tilde{f}}(c^q)\Delta c^q = -\tilde{f}(c^q)$
- So, (instead) solve $J_{\tilde{f}}(c^q)\Delta c^q = -\tilde{f}(c^q)$ via any least squares (QR) and minimum norm approach

- Note: setting the second factor in $J_{\tilde{f}}^{T}(c)\left(\tilde{f}(c^q) + J_{\tilde{f}}(c^q)\Delta c^q\right) \approx 0$ to zero also leads to $J_{\tilde{f}}(c^q)\Delta c^q = -\tilde{f}(c^q)$
- This is a linearization of the nonlinear system $\tilde{f}(c) = 0$, aiming to minimize $\hat{f}(c) = \frac{1}{2}\tilde{f}^{T}(c)\tilde{f}(c)$

Weighted Gauss Newton

- Given a diagonal matrix D indicating the importance of various equations:

$$DJ_{\tilde{f}}(c^q)\Delta c^q = -D\tilde{f}(c^q)$$

$$J_{\tilde{f}}^T(c^q)D^2 J_{\tilde{f}}(c^q)\Delta c^q = -J_{\tilde{f}}^T(c^q)D^2\tilde{f}(c^q)$$

- Recall: Row scaling changes the importance of the equations
  - It also changes the (unique) least squares solution for any overdetermined degrees of freedom

Regularized Gauss Newton

- When concerned about small singular values in $J_{\tilde{f}}(c^q)\Delta c^q = -\tilde{f}(c^q)$, one can add $\epsilon I = 0$ as extra equations (unit 12 regularization)
- This results in $\left(J_{\tilde{f}}^T(c^q)J_{\tilde{f}}(c^q) + \epsilon^2 I\right)\Delta c^q = -J_{\tilde{f}}^T(c^q)\tilde{f}(c^q)$
- This is often called Levenberg-Marquardt or Damped (Nonlinear) Least Squares