Lecture 9, Feb 16, 2024

Unconstrained Optimization

- Given $f(\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^n$ and $f\colon \mathbb{R}^n \mapsto \mathbb{R}$, we want to find $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} f(\boldsymbol{\theta})$
- Let $\boldsymbol{\theta}^*$ be a local minimum of f, then $f(\boldsymbol{\theta}^* + \epsilon\boldsymbol{p}) \geq f(\boldsymbol{\theta}^*)$ for any arbitrary $\boldsymbol{p}$ and small ϵ
  - $f(\boldsymbol{\theta}^* + \epsilon\boldsymbol{p}) = f(\boldsymbol{\theta}^*) + \epsilon\boldsymbol{p}^T\boldsymbol{g}(\boldsymbol{\theta}^*) + \frac{1}{2}\epsilon^2\boldsymbol{p}^T\boldsymbol{H}(\boldsymbol{\theta}^*)\boldsymbol{p} + \mathcal{O}(\epsilon^3)$ where $\boldsymbol{g}(\boldsymbol{\theta})$ is the gradient and $\boldsymbol{H}(\boldsymbol{\theta})$ is the Hessian
  - The middle two terms must be greater than or equal to zero due to the local minimum condition
  - This means that $\boldsymbol{g}(\boldsymbol{\theta}^*) = \boldsymbol{0}$, and $\boldsymbol{H}(\boldsymbol{\theta}^*)$ is symmetric positive definite, since $\boldsymbol{p}$ is arbitrary
- KKT conditions:
  - The first order necessary optimality condition states that the gradient must be zero at the minimum
  - The second order necessary optimality condition states that the Hessian must be positive semi-definite
  - If both the gradient is zero and the Hessian is positive definite, then we have sufficient conditions for a minimum

Gradient-Based Unconstrained Numerical Optimization

- Gradient-based algorithms are based on the following template:
  - Start with $k = 0$ and an initial guess $\boldsymbol{\theta}_0$
  - In each iteration:
    * Test for convergence; if we have converged, stop and take $\boldsymbol{\theta}_k$ as the solution; if not, continue
    * Compute the search direction $\boldsymbol{p}_k$
    * Compute the step length $\alpha_k > 0$ s.t. $f(\boldsymbol{\theta}_k + \alpha_k\boldsymbol{p}_k) < f(\boldsymbol{\theta}_k)$
    * Take $\theta_{k=1} \leftarrow \boldsymbol{\theta}_k + \alpha_k\boldsymbol{p}_k$ and $k \leftarrow k + 1$
- To obtain a valid search direction, we need $\boldsymbol{p}_k^T\boldsymbol{g}_k < 0$, i.e. the step and the gradient have to point in opposite directions
  - Take $\boldsymbol{p}_k = -\boldsymbol{B}\boldsymbol{g}_k$ for some symmetric positive definite $\boldsymbol{B}$
  - Possible choices for $\boldsymbol{B}$:
    * Steepest descent: $\boldsymbol{B} = \boldsymbol{1}$
      - The search direction is directly opposite to the gradient
    * Newton's method: $\boldsymbol{B} = \boldsymbol{H}_k^{-1}$
    * Quasi-Newton methods: $\boldsymbol{B} \approx \boldsymbol{H}_k^{-1}$
      - For computational reasons, these methods take an approximation of the inverse Hessian
- Computing the appropriate α_k is a tradeoff between reducing function evaluations (i.e. getting to the goal with fewer steps) and computational cost at each step
  - One technique is to take a number of α_ks and stop at the first one that meets some condition
  - Armijo sufficient decrease condition: $f(\boldsymbol{\theta}_k + \alpha_k\boldsymbol{p}_k) \leq f(\boldsymbol{\theta}_k) + \mu_1\alpha_k\boldsymbol{g}_k^T\boldsymbol{p}_k$
    * The constant μ_1 is typically chosen in the range of 10^{-4}
  - Backtracking line search:
    * Choose starting step length (between 0 and 1)
    * Check if Armijo condition is satisfied, and if so use the current step length
    * If not, $\alpha \leftarrow \rho\alpha$ (typically $\rho \in [0.1, 0.5]$) and check again
- Steepest descent algorithm:
  - Select initial guess $\boldsymbol{\theta}_0$, gradient tolerance ϵ_g, absolute tolerance ϵ_a, relative tolerance ϵ_r
  - If $\|\boldsymbol{g}(\boldsymbol{\theta}_k)\|_2 \leq \epsilon_g$ then stop
  - Set $\boldsymbol{p}_k = -\dfrac{\boldsymbol{g}(\boldsymbol{\theta}_k)}{\|\boldsymbol{g}(\boldsymbol{\theta}_k)\|_2}$
  - Find α_k such that $f(\boldsymbol{\theta}_k + \alpha\boldsymbol{p}_k)$ satisfies the sufficient decrease conditions
  - Update as $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha_k\boldsymbol{p}_k$
  - Evaluate $f(\boldsymbol{\theta}_{k+1})$; if $|f(\boldsymbol{\theta}_{k+1}) - f(\boldsymbol{\theta}_k)| \leq \epsilon_a + \epsilon_r|f(\boldsymbol{\theta}_k)|$ for two successive iterations, stop
    * In this case our algorithm has gotten stuck
- For steepest descent, for all k, we can show that $\boldsymbol{p}_{k+1}$ is orthogonal to $\boldsymbol{p}_k$

- $\frac{\partial f(\boldsymbol{\theta}_{k+1})}{\partial \alpha} = 0 \implies \vec{\nabla}^T f(\boldsymbol{\theta}_{k+1})\boldsymbol{p}_k = 0 \implies \boldsymbol{g}(\boldsymbol{\theta}_{k-1})^T \boldsymbol{g}(\boldsymbol{\theta}_k) = 0$
- This means we're always zig-zagging at each iteration
- This is inefficient (high number of iterations needed), but it is guaranteed to converge
- Convergence rate is linear: $\lim_{k \to \infty} \frac{|f(\boldsymbol{\theta}_{k-1}) - f(\boldsymbol{\theta}^*)|}{|f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}^*)|} = K$

- Conjugate gradient method (nonlinear, first order, i.e. only uses the gradient):
  - $\boldsymbol{p}_0 = -\frac{\boldsymbol{g}(\boldsymbol{\theta}_0)}{\|\boldsymbol{g}(\boldsymbol{\theta}_0)\|_2}$ for the first step
  - For all other steps $\boldsymbol{p}_k = -\boldsymbol{g}_k + \beta_k \boldsymbol{p}_{k-1}$
    * Fletcher-Reeves: $\beta_k = \frac{\boldsymbol{g}_k^T \boldsymbol{g}_k}{\boldsymbol{g}_{k-1}^T \boldsymbol{g}_{k-1}}$
    * Polak-Ribieve: $\beta_k = \frac{\boldsymbol{g}_k^T (\boldsymbol{g}_k - \boldsymbol{g}_{k-1})}{\boldsymbol{g}_{k-1}^T \boldsymbol{g}_{k-1}}$
  - Since we're using knowledge from previous gradients, this does not zig zag as much as steepest descent
- Newton's method (second order, i.e. also uses the Hessian):
  - Quadratic convergence: $\lim_{k \to \infty} \frac{|f(\boldsymbol{\theta}_{k-1}) - f(\boldsymbol{\theta}^*)|}{|f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}^*)|^2} = K > 0$
  - Use $\boldsymbol{p}_k = \boldsymbol{H}_k^{-1} \boldsymbol{g}_k$
    * Note that this step direction is only valid if $\boldsymbol{H}_k$ is positive definite
    * We can check the dot product of $\boldsymbol{p}_k$ with $\boldsymbol{g}_k$ at each step, and if this is invalid (i.e. greater than 0) we simply go in the opposite direction
- Newton's method requires computation of the inverse Hessian, which can be very inefficient; for computational reasons, we use quasi-Newton methods which approximate the inverse Hessian
  - These don't make use of the Hessian but can still get better than linear convergence
  - Iteratively update $\boldsymbol{B}_{k+1}^{-1} = \boldsymbol{B}_k^{-1} + \Delta \hat{\boldsymbol{B}}_k$
  - $\Delta \hat{\boldsymbol{B}}_k$ depends on the specific method