Lecture 19: Optimization Algorithms

goals:
- intro to optimization algorithms
- unconstrained methods
  - gradient descent
  - Newton's method

- so far we've focused on modeling or formulating optimization problems + studying their convexity properties

- convexity (largely) allows decoupling of modeling and algorithm design
  - can be very effective user of convex opt. w/o knowing a lot about underlying algorithms

- however advanced, research-level users should have good knowledge of algorithms to enhance practical effectiveness + enable analysis/design of new algorithms

Optimization algorithms are iterative

- start with a guess of optimal solution
- generate a sequence of iterates that approach a (global, ideally) solution
    - can categorize algorithms based on what information is used at each step to generate new iterates
        - possibly current and past values of objective and constraint functions and their derivatives
    - interpret as algorithms for solving (algebraic) optimality conditions (KKT)
        - ultimately, lowest level steps involve basic numerical linear algebra operations (matrix factorization via LU, Cholesky, LDL^T, ... decompositions - see BV Appendix C)
- good algorithm properties:

  ① Computational efficiency (fast, low memory)

  ② Robustness (works well on wide range of problems w/in class, low sensitivity to numerical errors in data, implementation)

- often see efficiency-robustness tradeoffs
    - eg. fastest method may lack robustness

Oracle Models

- algorithms often studied using oracle models for objective + constraint functions
    - need not know functions explicitly, instead "query an oracle" to find out info at a current iterate x_k (typically function values + some derivatives)

Unconstrained Optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

- assume f convex, twice differentiable, optimal value attained and finite

- goal: find a point $\bar{x}$ where

$$\|\nabla f(\bar{x})\| \leq \varepsilon$$

Descent Methods

$$x_{k+1} = x_k + \underbrace{\alpha_k}_{\text{step size}} \underbrace{\Delta x_k}_{\text{search direction}} \quad \text{with} \quad f(x_{k+1}) < f(x_k)$$

Algorithm Input: starting point $x_0 \in \operatorname{dom} f$, set $k=0$

```
while (stopping criterion not satisfied: $\|\nabla f(x_k)\| > \varepsilon$)
    find descent direction $\Delta x_k$
    line search to choose step size $\alpha_k > 0$
    update $x_{k+1} = x_k + \alpha_k \Delta x_k$
```

Line Search

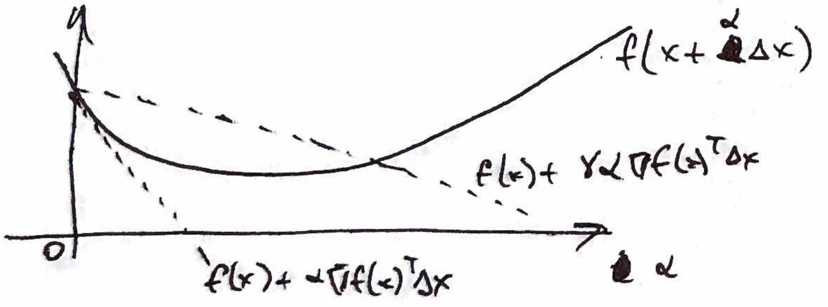
① Exact line search: $\alpha_k = \underset{\alpha > 0}{\operatorname{argmin}} \; f(x_k + \alpha \Delta x_k)$

- a convex, 1D subproblem

② Backtracking line search (parameters γ, β) $\gamma \in (0, \frac{1}{2})$, $\beta \in (0,1)$

- start with $\alpha = 1$, repeat $\alpha := \beta\alpha$ until

$$f(x + \alpha \Delta x) < f(x) + \gamma \alpha \nabla f(x)^T \Delta x$$



Note: Also often have

③ constant step size $\alpha_k = \alpha$

④ slowly decreasing step size

e.g. $\alpha_k = \frac{1}{\sqrt{k}}$ or $\alpha_k = \frac{1}{k}$

Gradient Descent

- descent method with $\Delta x_k = -\nabla f(x_k)$

$\boxed{\text{Algorithm}}$ Input: starting point $x_0 \in \text{dom } f$, set $k=0$

```
while $\|\nabla f(x_k)\| > \varepsilon$
    set $\Delta x_k = -\nabla f(x_k)$
    line search to choose step size $\alpha_k > 0$
    update $x_{k+1} = x_k + \alpha_k \Delta x_k$
```

- global convergence to an optimal solution under some mild assumptions on f (e.g. strong convexity)

$$f(x_k) - f^* \leq c^k \left(f(x_0) - f^* \right)$$

where $c \in (0,1)$ depends on line search type and properties of f (specifically the condition number of $\nabla^2 f(x)$)

- assume $\exists\, m, M > 0$ such that

$$mI \preceq \nabla^2 f(x) \preceq MI \quad \forall x$$

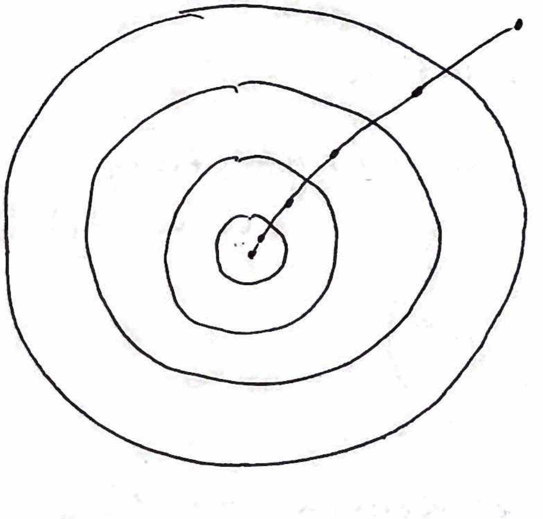
i.e. ① f is strongly convex w/ parameter m

② f has Lipschitz gradient w/ parameter M

$$\Rightarrow \underset{\substack{\uparrow \\ \text{condition number}}}{\text{cond}}\left(\nabla^2 f(x)\right) \le \frac{M}{m}$$

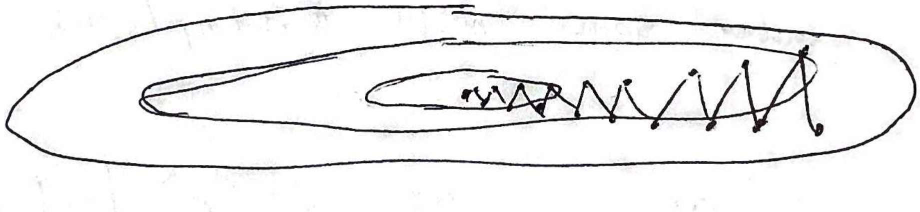
gradient descent performance depends on $\frac{M}{m}$

$\frac{M}{m} \approx 1$



few iterations

$\frac{M}{m} >> 1$



many iterations

- can reduce this by a change of variables (aka "scaling" or "preconditioning")

e.g. $f(x_1,x_2) = x_1^2 + \gamma x_2^2 \xrightarrow[y_2 = \frac{1}{\sqrt{\gamma}} x_2]{\text{let } y_1 = x_1} f(y_1,y_2) = y_1^2 + y_2^2$

$\nabla^2 f(x) = \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix} \qquad \nabla^2 f(y) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

but not always easy to do this in general

Gradient Descent

Advantages:

① Simple to implement, cheap to execute

- widely used in large-scale optimization (e.g. in machine learning) with millions/billions of variables + constraints

② Many useful variations

- e.g. accelerated/robust methods, stochastic gradient descent, projected/proximal, etc.

③ well-developed convergence theory

- especially for convex functions

Disadvantages:

① Convergence can be very slow, especially sensitive to poor conditioning

② Not always easy to precondition, tune step size

Ex GD variation: Heavy-ball Method

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

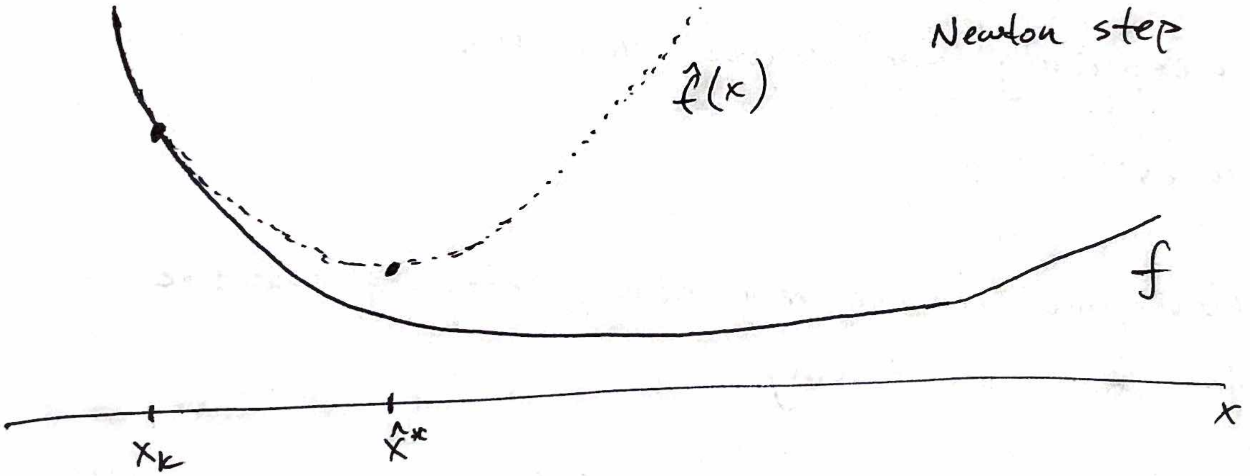
- still first-order, uses past iterates to accelerate convergence

Newton's Method

- a second-order method that uses the Hessian $\nabla^2 f(x)$ in addition to the gradient $\nabla f(x)$
- can be interpreted as sequential quadratic approximation: approximate f by quadratic, solve analytically, repeat
  - near x_k we have (via Taylor expansion)

$$f(x) \approx \hat{f}(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

$$\text{minimizer:} \quad \hat{x}^* = x_k \underbrace{- \nabla^2 f(x_k)^{-1} \nabla f(x_k)}_{\text{Newton step}}$$



- not hard to show that Newton step is independent of affine change of variables (affine invariance)
  - no sensitivity to conditioning of f!

Define Newton Decrement

$$\lambda(x) = \left(\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right)^{\frac{1}{2}}$$

- also affine invariant, useful both for theoretical/ convergence analysis + practical stopping criterion

Algorithm Input: starting point $x_0 \in \text{dom } f$, tolerance $\varepsilon > 0$

set $k = 0$

```
while $\frac{1}{2}\lambda(x_k)^2 > \varepsilon$
    compute Newton step + decrement
        $\Delta x_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$, $\lambda(x) = (\nabla f(x_k)^T \nabla^2 f(x_k)^{-1} \nabla f(x_k))^{\frac{1}{2}}$
    line search to choose step size $\alpha_k > 0$
    update: $x_{k+1} = x_k + \alpha_k \Delta x_k$
```

Advantages:

① often very fast

② Affine invariance (no preconditioning necessary)

Disadvantages:

- computationally expensive
  - even storing the $n \times n$ Hessian matrix can be prohibitive for large n
  - inverting Hessian required at each step, $O(n^3)$

- "Quasi-Newton" methods approximate Hessian + its inverse
  - Conjugate gradient
  - Broyden-Fletcher-Goldfarb-Shanno (BFGS)

Interior Point Methods

- How to handle constraints?

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && f_i(x) \le 0 \qquad i = 1, \dots, m \end{aligned}$$

- Basic idea: augment objective using a barrier function that approaches ∞ as we approach a constraint boundary

$$\text{minimize} \quad f(x) - \mu \sum_{i=1}^{m} \log(-f_i(x)) \qquad \text{"log barrier"}$$

Then alternate between:

① iteration of an unconstrained method (usually Newton)

② shrinking μ toward zero to get better approximation of constraint